Cvičenia z geometrických algoritmov a ich zložitosti

(pracovná verzia)

zostavovateľ: Pavel Chalmovianský

Cvičenia riešené študentami z oblasti výpočtovej geometrie a zložitosti geometrických algoritmov.

Bratislava, 11. októbra 2012

KAPITOLA 1

Úvod

Cieľom tohoto dokumentu je zozbierať a publikovať riešenia cvičení z oblasti geometrických algoritmov a ich zložitosti.

Ku skúške treba splniť aspoň jednu z nasledujúcich požiadaviek:

- vyriešiť dostatok úloh počas semestra alebo
- napísať písomku na konci semestra.

Teraz bližšie vysvetlenie každej z možností.

Úlohy riešené počas semestrau musia byť zo štyroch oblastí – konvexné obaly a ich konštrukcie, proximita, geometrické vyhľadávanie a prieniky. Z každej oblasti aspoň jedna úloha.

Za vypracovanie jednej úlohy získate max. 100 bodov, pričom rozdelenie bodov je nasledujúce:

- správne riešenie úlohy max. 50 bodov (maximálny počet bodov je uvedený v zátvorke pri zadaní cvičenia), rozhoduje kvalita riešenia, jasnosť metódy a podania
- napísanie riešenia v elektronickej podobe(LaTeX) max. 30 bodov, rozhoduje koľko práce je na tom, aby sa to dalo publikovať, či daný text obsahuje aj nejaký ilustračný obrázok apod.
- ak ste vymysleli iné (zaujímavé) alebo lepšie riešenie ako Vaši predchodcovia (t.j. nejaké riešenie je už uverejnené v tejto zbierke) získate prémiu max. 20 bodov (lepšie znamená šikovnejšie, pochopiteľnejšie, kratšie, prehľadnejšie apod.); ak dané cvičenie ešte nebolo riešené, túto prémiu získavate pri správnom riešení automaticky vo výške 10 bodov
- prémiových 10 bodov môžete získať, ak si zadanie cvičenia nájdete sami na internete, v knihách apod., a žiadne podobné nie je v tejto zbierke a odsúhlasím ho

Riešenie ľubovolného cvičenia môžete iterovať (najviac 2x), pričom zakaždým dostanete spätnú väzbu v podobe návodov na pokračovanie. Rovnaké riešenia ocením známkou a vydelím počtom rovnakých riešení (aj iteratívne, ak by sa objavili nové).

Ku skúške potrebujete z každej oblasti vyriešený aspoň jeden príklad na 25 bodov a dokopy aspoň 200 bodov. Kto získa aspoň 350 bodov, nemusí na skúške riešiť príklady. Kto dosiahne 400 bodov, má zaručenú skúšku. O známke sa dohodneme podľa kvality a množstva príkladov vyriešených počas semestra.

Ciele elektronického spracovania sú dva. Jednak sa naučíte písať texty, ktoré by mali byť čitateľné pre iných a jednak sa naučíte, ako má formálne taký text vyzerať a čo má obsahovať.

Ako pomôcku som vytvoril súbor `priklad.tar.bz2`, v ktorom nájdete základnú kostru texovského súboru k napísaniu riešenia úlohy. Obsahuje aj príklad uvedený nižšie.

Termíny na odovzdanie úloh sú:

- úloha č. 1: 11. 11. 2012,
- úloha č. 2: 2. 12. 2012
- úloha č. 3: 23. 12. 2012,
- úloha č. 4: 6. 1. 2013.

CVIČENIE 1.1 (20 bodov). *Nech* $\mathbf{p}_1, \ldots, \mathbf{p}_n \in \mathbb{E}^2$ *sú body. Zostrojte konvexný obal tejto množiny bodov tak, aby ste dosiahli optimálnu zložitosť algoritmu.*

Riešenie (Ján Riešiteľ, 2.9.2008): Body $\mathbf{p}_i$ zotriedime lexikograficky pomocou polárnych súradníc. Pričom najskôr triedime podľa uhla ϕ a potom podľa vzdialenosti r. Dbáme na to, aby stred súradnicovej sústavy bol vnútri konvexného obalu. To dosiahneme tak, že zaň zoberieme niektoré z ťažísk trojuholníkov s vrcholmi v daných bodoch $\mathbf{p}_i$.

Po zotriedení začneme s niektorým extremálnym vrcholom. Napr. s takým, ktorý má maximálnu súradnicu x. Ak je takých viac, zoberieme ten, ktorým má z nich maximálnu súradnicu y. V zásobníku budeme zostavovať konvexný obal. Vložíme tam práve nájdený extremálny vrchol a potom postupujeme podľa utriedenia. Druhý vrchol tam tiež vložíme. Pri treťom a každom ďalšom vrchole (označme ho $\mathbf{v}_j$) otestujeme, či je orientovaný uhol tvorený predposledným vrcholom zásobníka (označme ho $\mathbf{v}_{i-1}$), posledným vrcholom zásobníka (označme ho $\mathbf{v}_i$) a $\mathbf{v}_j$ viac ako π. Ak áno, zaradíme $\mathbf{v}_j$ na vrchol zásobníka. Ak nie, zmažeme vrchol zo zásobníka.

Algoritmus končí vtedy, keď spracujeme všetky body.

Zložitosť tohoto algoritmu je $\mathcal{O}(n \log n)$, keďže obsahuje triedenie a spracovanie bodov trvá nanajvýš $\mathcal{O}(n)$. □

Tento príklad by som obodoval nasledovne: je správne vyriešený a pomerne zrozumiteľne napísaný. Časť o začiatku algoritmu by mohla byť opísaná jasnejšie alebo ilustrovaná obrázkom. Postup by mohol byť dokumentovaný nejakým formálnejším algoritmom v pseudokóde. Za riešenie je 15 bodov, za napísanie (bez obrázka a formálneho algoritmu) je 15 bodov. Ďalších 20 bodov je za to, že ide o nový príklad. Dokopy teda 50 bodov.

Alternatívne si môžete zvoliť písomku z príkladov z tejto zbierky na konci semestra, ktora bude obsahovať 4 úlohy, po jednej z každej

tematickej oblasti. Na úspešné absolvovanie vyžadujem 50% bodov. **Body zo semestra sa v tomto prípade nepočítajú.**

Cvičenia v nasledujúcich častiach pochádzajú z rôznych zdrojov. časť z nich je z kníh ako napr. [**PS85**],[**dBvKOS00**], [**BY98**], časť je z internetu napr. [**Eri**] z a časť z nich ma napadla pri mojej práci. Za opravu mnohých chýb v texte vďačím Viktorovi Majorovi.

KAPITOLA 2

Štartovacie úlohy

CVIČENIE 2.1 (20 bodov). *Nájdite algoritmus, ktorý optimálne určí v množine n bodov z euklidovskej roviny, či sa medzi nimi nachádzajú tri kolineárne body.*

Riešenie (Marta Režnáková, 19.10.2008): Inými slovami, hľadáme také tri body, ktoré ležia na jednej priamke. ***Algoritmus:***

while(!$kolin \,\&\&\, (i < max)$){ *//rob kým neprídeš na koniec alebo nenájdeš kolineárne body*
    i++; *//posúvame sa v množine bodov*
    pre i-ty bod priraď ostatným polárne súradnice; *//pričom i-ty bod bude stred kružnice*
    j=0;
    while(!$kolin \,\&\&\, (j < max)$){ *//rob pre i-ty bod*
        vezmi bod z množiny; *// z neusporiadanej časti, bez i-teho bodu*
        if(medzi utriedenými bodmi existuje bod s rovnakou súradnicou){
            kolin=*true*;
        } else {
            zatrieď ho medzi už utriedené body;
            j++;
        }
    }
}

□

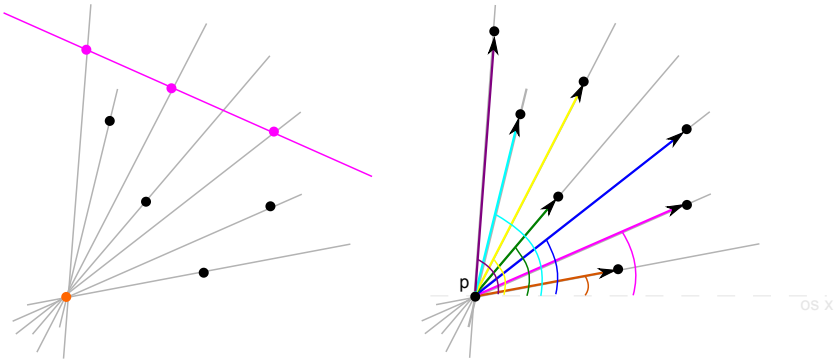
OBR. 2.1. Hľadanie kolineárnych bodov pomocou polárnych súradníc.

Riešenie (Martina Bátorová, 30.10.2008):

(1) **Triviálny prístup:** Vyberme v rovine bod, označme ho p a utrieďme body množiny podľa polárnych súradníc vzhľadom na tento bod. Teraz zostrojme priamky prechádzajúce bodom p a jedotlivými bodmi vstupnej množiny. Každá takáto priamka má smernicu - odchýlku od osi x. Stačí nám teda nájsť dva body, zodpovedajúce priamky ktorých zvierajú s osou x rovnaký uhol.

(2) **Komplikácie:** Takýto algoritmus by mal časovú zložitosť rádu $\mathcal{O}(n \log n)$ (utriedenie bodov $\mathcal{O}(n \log n)$ + výpočet smerníc $(n-1)\mathcal{O}(1)$ + utriedenie smerníc $\mathcal{O}(n \log n)$ + vyhľadanie rovnakej smernice $\mathcal{O}(n)$), čo by bolo veľmi dobré. Obrázok vľavo však ukazuje, že nie sme schopní zvoliť bod p tak, aby sme mali zaručené, že problém vďaka nemu rozhodneme správne. Vo všeobecnosti totiž môže nastať prípad, že konštruované priamky prechádzajúce bodom p prechádzajú ešte práve jedným vrcholom, ale i tak sa v množine nachádzajú tri kolineárne body. Preto musíme postup z bodu 1. aplikovať pre každý bod v množine.

Až keď preiterujeme všetky body a zistíme, že sme medzi nimi nenašli kolineárne, môžeme prehlásiť, že sa tam také nenachádzajú.



Obr. 2.2. Vľavo prípad komplikácie pri snahe o zjednodušenie algoritmu. Vpravo algoritmus pre jeden fixovaný bod.

(3) **Časová zložitosť:** Pri fixovanom bode potrebujeme utriediť zvyšné body, vypočítať smernice a vyhodnotiť kolinearitu.

Pri použití optimálneho triedenia vieme triediť v čase $\mathcal{O}(n \log n)$, výpočet jednej smernice trvá $\mathcal{O}(1)$ a vyhľadanie kolinearity (utriedenie smerníc a vyhľadanie rovnakých hodnôt) trvá $\mathcal{O}(n \log n)+ \mathcal{O}(n)$. Na spracovanie jedného bodu potrebujeme $\mathcal{O}(n \log n) + (n-1)\mathcal{O}(1) + \mathcal{O}(n \log n) + \mathcal{O}(n) = \mathcal{O}(n \log n)$ operácií.

Tento postup opakujeme pre všetky body, čo je n-krát. Dolný odhad zložitosti celého algoritmu je teda $\mathcal{O}(n^2 \log n)$.

Dôvod, prečo je uvedený odhad naozaj dolným odhadom, je nasledujúci:

(a) Potrebujeme použiť stabilné triedenie. Dolný odhad triedenia porovnávaním je $\mathcal{O}(n \log n)$. Hoci poznáme i triediace algoritmy pracujúce v čase $\mathcal{O}(n)$, tieto kladú na usporiadavanú množinu ďalšie požiadavky, napr. na interval, z ktorého prvky vyberáme. Keďže my chceme triediť všeobecnú množinu, dolná hranica zložitosti triedenia je $\mathcal{O}(n \log n)$.

(b) Uviedli sme si dôvod, prečo musíme celú procedúru opakovať pre všetky prvky vstupnej množiny, čo je n–krát.

Odhad časovej náročnosti je teda skutočne $\mathcal{O}(n^2 \log n)$.

(4) Procedúru môžeme potom zjednodušene zapísať napr. takto:

```
procedure NajdiKolinearneBody(Body P)
{
  zvol a oznac bod p=P[i] z mnoziny P;
  preloz bodmi p a P[j] (j = 0..i-1,i+1..n) priamky;
  vypocitaj smernice priamok;
  utried smernice;
  zisti, ci dva body zo zvysku P maju rovnaku smernicu
     ano: return 'TRUE';
     nie: vsetky body su oznacene -> return 'FALSE'
          inak chod od zaciatku
}
```

□

Riešenie (Jana Hlinková, 13.10.2008):

Úlohou je zistiť, či spomedzi n bodov niektoré tri ležia na jednej priamke. Majme body uložené v poli B, teda B[i] je i-ty bod, B[i].x, B[i].y sú jeho dve súradnice. Priamku prechádzajúcu bodom B[i] vieme parametricky popísať nasledovne:

$$p: \quad x = B[i].x + t.u_1 \qquad y = B[i].y + t.u_2 \qquad t \in \mathbb{R},$$

kde (u_1, u_2) je smerový vektor priamky p.

Nech B[i], B[j], B[k] sú kolineárne body. Potom samozrejme smerové vektory priamok určených rôznymi dvojicami týchto bodov sú rovnaké (po prípadnej normalizácii či škálovaní). Tento fakt možno teda využiť pri overovaní kolinearity.

Algoritmus by pracoval nasledovne:

Kým neboli nájdené tri kolineárne body alebo kým neboli spracované všetky dvojice bodov:

(1) Pre aktuálnu dvojicu B[i], B[j] vypočítaj smerový vektor $(u_1, u_2) = (B[j].x - B[i].x, B[j].y - B[i].y)$ a uprav na tvar $(1, \frac{u_2}{u_1})$ (ak $u_1 = 0$ tak na tvar $(1, POSITIVE\_INFINITY)$).
(2) Ulož hodnotu $\frac{u_2}{u_1}$ a smerník na dvojicu (i, j) do vyváženého binárneho vyhľadávacieho stromu (napr. Red-Black strom) (pričom prvky stromu sú usporiadané na základe $\frac{u_2}{u_1}$). Vieme, že pri ukladaní prvku do binárneho vyhľadávacieho stromu sa prvok porovnáva s prvkami vrcholov stromu na ceste od koreňa po výslednú pozíciu prvku. V našom prípade sa porovnávajú hodnoty $\frac{u_2}{u_1}$ zodpovedajúce smerovým vektorom priamok. Teda ak pri porovnávaní prvkov zaznamenáme rovnosť, znamená to, že niektoré dvojice bodov ležia na priamkach s rovnakým smerovým vektorom. Stačí už len overiť, či naozaj body ležia na jednej priamke. Ak áno, algoritmus skončí, ak nie, k vrcholu stromu obsahujúcemu $\frac{u_2}{u_1}$ sa pridá smerník na novú dvojicu (i,j).

Maximálna časová zložitosť sa dosiahne v prípade, že v danej množine nie sú žiadne tri kolineárne body, pričom prebehne $\mathcal{O}(n^2)$ iterácií. V každej iterácii výpočet smerového vektoru potrebuje $\mathcal{O}(1)$ času a vkladanie do stromu a porovnávanie $\mathcal{O}(\log n)$ času. Celkový horný časový odhad je teda $\mathcal{O}(n^2.\log n)$. Algoritmus potrebuje pomocnú pamäť veľkosti $\mathcal{O}(n^2)$.

```
Init:
        Point2d[] B = {B[1], B[2], ... , B[n]};
        RedBlackTree
        Boolean found = false;
        Integer i,j = 1;

while (!found && (i <= n)){
        j = i;
        while (!found && (j <= n)){
                Double u2/u1 = computeDirectionalVector(B[i], B[j]);
                found = RBT.insert(u2/u1, i, j);
                j++;
        }
        i++;
}
```

□

Riešenie (Peter Danko, 25.10.2008):

Kroky algoritmu:

(1) Body z danej množiny pospájame priamkami tak, aby každá dvojica bodov nám určovala jednu priamku. Čiže vytvoríme $\binom{n}{2}$ priamok čo trvá $\mathcal{O}(n^2)$, keďže porovnávame dvojice.

(2) Takto vytvorené priamky usporiadame podľa sklonu a rozdelíme do tried, t.j.: rovnobežné priamky budú v rovnakej triede.
(3) Následne pre každú priamku hľadáme jej prienik s y-ovou osou. Ak zistíme, že aspoň dve priamky rovnakej triedy majú totožný prienik s y-ovou osou, tak body incidentné s týmito priamkami sú kolineárne.

Osobitne musíme ošetriť body incidentné so zvislými priamkami, pretože v $\mathbb{E}^2$ nemajú tieto priamky jednoduchý prienik s y-ovou osou (napríklad, tri body s rovnakou x-ovou súradnicou sú kolineárne).

Ak v druhom kroku algoritmu použijeme na utriedenie priamok algoritmus s časovou zložitosťou $\mathcal{O}(n \log n)$, potom časová zložitosť nášho algoritmu bude $\mathcal{O}(n^2 \log n^2) = \mathcal{O}(n^2 \log n)$.

PRÍKLAD 2.1. *Na obr. 2.3 je daná množina bodov $A, B, C, D, E, F \in \boldsymbol{E}^2$. Po vykonaní prvého kroku algoritmu dostaneme priamky:*
$a = \overleftrightarrow{AB}$, $b = \overleftrightarrow{BC}$, $c = \overleftrightarrow{AC}$, $d = \overleftrightarrow{BD}$, $e = \overleftrightarrow{DE}$, $f = \overleftrightarrow{BE}$, $g = \overleftrightarrow{CE}$, $h = \overleftrightarrow{EF}$, $i = \overleftrightarrow{CF}$, $j = \overleftrightarrow{CD}$, $k = \overleftrightarrow{DF}$, $l = \overleftrightarrow{BF}$, $m = \overleftrightarrow{AF}$, $n = \overleftrightarrow{AE}$, $o = \overleftrightarrow{AD}$.

Druhý krok algoritmu nám vytvorí osem tried rovnobežných priamok. Prvá trieda obsahuje priamky a, b, c, druhá trieda obsahuje priamky d, e, f, m a tretia trieda obsahuje priamky g, h, i. Každá ďalšia trieda obsahuje práve jednu priamku.

V poslednom kroku algoritmu zistíme, že priamky d, e, f majú rovnaký prienik s y-ovou osou a teda body B, D, E sú kolineárne. Podobne priamky g, h, i majú spoločný prienik s osou y a s nimi incidujúce body C, E, F sú kolineárne. Priamky a, b, c nemajú s osou y jednoduchý prienik a body A, B, C majú rovnakú x-ovú súradnicu, takže aj táto trojica bodov je kolineárna.

Obr. 2.3 nám ilustruje tento príklad.

POZNÁMKA 2.1 (Sklon priamky). *Nech body $A = (x_a, y_a), B = (x_b, y_b)$ sú body na priamke. Potom sklon k priamky $\overleftrightarrow{AB}$ vypočítame ako pomer zmeny súradnice x a zmeny súradnice y:*

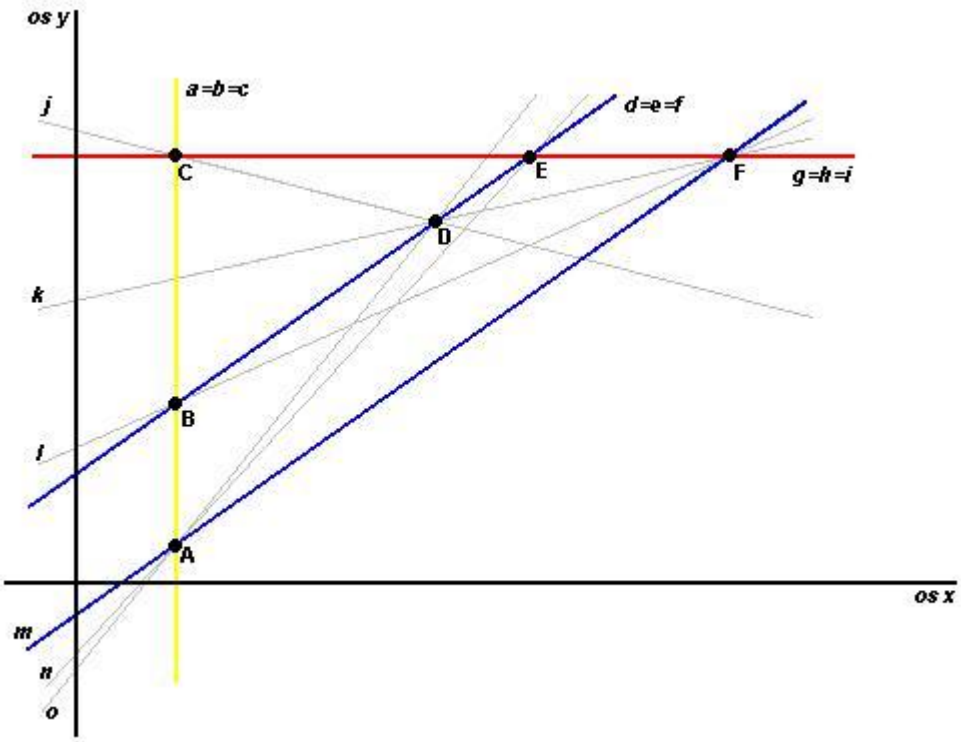
$$k = \frac{y_b - y_a}{x_b - x_a} = \frac{\Delta y}{\Delta x} \tag{1}$$

□

CVIČENIE 2.2 (15 bodov). *Opíšte čo najefektívnejšiu procedúru na zistenie prieniku dvoch priamok v $\mathbb{E}^2$. Ošetrite všetky prípady.*

Riešenie (Dušan Pácal, 28.9.2008): Predpokladajme, že obe priamky máme zadané parametricky:

$p = A + t.\vec{u}$ $\qquad$ $q = B + r.\vec{v}$ $\qquad$ $t, r \in \mathbb{R}$
$A = (a_1, a_2)$; $\vec{u} = (u_1, u_2)$ $\qquad$ $B = (b_1, b_2)$; $\vec{v} = (v_1, v_2)$



OBR. 2.3. Príklad množiny bodov A, B, C, D, E, F a priamok incidentných s týmito bodmi.

1. určíme rovnicu priamky v tvare $ax + by + c = 0$
Dostávame rovnicu: $-u_2.a_1 + u_1.a_2 = -c$, a z nej vypočítame c.
2. vypočítame determinant, nech $d = det \begin{vmatrix} u_1 & v_1 \\ u_2 & v_2 \end{vmatrix}$
3. ak $d = 0$, máme 2 možnosti: priamky sú totožné, alebo rovnobežné.
4. zistíme, či bod B leží na priamke p, či $-u_2.b_1 + u_1.b_2 + c = 0$?
5. ak áno, priamky sú totožné, ak nie priamky sú rovnobežné a nemajú spoločný prienik.
6. určíme prienik dvoch rôznobežných priamok:
$-u_2.(b_1 + r.v_1) + u_1.(b_2 + r.v_2) + c = 0$
Z rovnice vypočítame r a dosadíme do vyjadrenia priamky q.
7. dostávame súradnice bodu prieniku:
$x = b_1 + r.v_1$
$y = b_2 + r.v_2$

```
prienik2priamok(a1, a2, u1, u2, b1, b2, v1, v2) {
  double c = u2 * a1 − u1 * a2;
  double d = u1 * v2 − u2 * v1;
```

```
  if (d == 0) {
    if (−u2 ∗ b1 + u1 ∗ b2 + c == 0) return TOTOZNE;
    else return NEMAJU PRIENIK;
  } else {
    double r = (u2 ∗ b1 − u1 ∗ b2 − c)/(−d);
    double X = b1 + r ∗ v1;
    double Y = b2 + r ∗ v2;
    return POINT(X, Y);
  }
}
```

□

Riešenie (Viktória Bakurová, 5.11.2009): Nech máme dané v rovine dve priamky, označme ich p a q. Úlohou je nájsť prienik týchto priamok. V rovine môžu nastať tri prípady: priamky p a $q$sú rovnobežné, totožné alebo rôznobežné. V poslednom prípade majú práve jeden bod prieniku.

Priamku v rovine môžeme mať zadanú parametricky a všeobecne.

1. Nech priamka p aj priamka q sú dané parametricky.

$p : X = A + t\overrightarrow{u}, t \in (-\infty, \infty)$ t.j. $x = a_x + tu_x$

$y = a_y + tu_y, t \in (-\infty, \infty)$

$q : X = B + s\overrightarrow{v}, s \in (-\infty, \infty)$ t.j. $x = b_x + sv_x$

$y = b_y + sv_y, s \in (-\infty, \infty)$

a) ak smerové vektory priamok p a q sú rovnaké až na násobok, t.j. ak $u_x v_y = u_y v_x$, tak tieto priamky sú buď totožné alebo rovnobežné.

O tom, ktorá z týchto možností nastane, rozhodneme napr. tak, že zistíme, či bod A patriaci priamke p patrí aj priamke q. Ak áno, ide o totožné priamky.

Nech $v_x \neq 0 \wedge v_y \neq 0$. Súradnice bodu A dosadíme do rovnice priamky q a vyjadríme si parameter s:

$a_x = b_x + sv_x \Longrightarrow s = \frac{a_x - b_x}{v_x}$

$a_y = b_y + sv_y \Longrightarrow s = \frac{a_y - b_y}{v_y}$

Ak sa dané parametre rovnajú, t.j. $\frac{a_x - b_x}{v_x} = \frac{a_y - b_y}{v_y}$, priamky sú totožné, inak sú rovnobežné.

Nech teraz $v_x = 0 \wedge v_y \neq 0$. V tomto prípade porovnáme x-ovú súradnicu bodu A a bodu B. Ak sa rovnajú, teda $a_x = b_x$, priamky sú totožné, inak sú rovnobežné.

Analogicky, nech $v_x \neq 0 \wedge v_y = 0$. Porovnáme y-ovú súradnicu bodu A a bodu B.Ak sa rovnajú, teda $a_y = b_y$, priamky sú totožné, inak sú rovnobežné.

V prípade, že aj $v_x = 0 \wedge v_y = 0$, priamka by sa zredukovala na bod a nemá význam hovoriť o rovnobežnosti alebo totožnosti priamok.

b) ak smerové vektory priamok neurčujú rovnaký smer, tak sú priamky rôznobežné a hľadáme ich prienik:

Hľadáme taký parameter sa t, pre ktoré bude platiť:

$a_x + tu_x = b_x + sv_x$

$a_y + tu_y = b_y + sv_y$

Eliminujeme parameter s a vyjadríme si parameter t : $t = \frac{(a_x-b_x)v_y-(a_y-b_y)v_x}{u_yv_x-u_xv_y}$. Dosadením tohoto parametra do rovnice priamky p dostaneme $x-$ovú a $y-$ovú súradnicu prieniku.

$x = a_x + \frac{(a_x-b_x)v_y-(a_y-b_y)v_x}{u_yv_x-u_xv_y}u_x$

$y = a_y + \frac{(a_x-b_x)v_y-(a_y-b_y)v_x}{u_yv_x-u_xv_y}u_y$.

Algoritmus:

1. ak $u_xv_y = u_yv_x$ a $v_x \neq 0 \wedge v_y \neq 0$
   - ak $\frac{a_x-b_x}{v_x} = \frac{a_y-b_y}{v_y}$, potom totožné
   - ak $\frac{a_x-b_x}{v_x} \neq \frac{a_y-b_y}{v_y}$, potom rovnobežné
2. ak $u_xv_y = u_yv_x$ a $v_x = 0 \wedge v_y \neq 0$
   - ak $a_x = b_x$, potom totožné
   - ak $a_x \neq b_x$, potom rovnobežné
3. ak $u_xv_y = u_yv_x$ a $v_x \neq 0 \wedge v_y = 0$
   - ak $a_y = b_y$, potom totožné
   - ak $a_y \neq b_y$, potom rovnobežné
4. inak

$x = a_x + \frac{(a_x-b_x)v_y-(a_y-b_y)v_x}{u_yv_x-u_xv_y}u_x$

$y = a_y + \frac{(a_x-b_x)v_y-(a_y-b_y)v_x}{u_yv_x-u_xv_y}u_y$, prienik

2. Nech priamka p aj priamka q sú dané všeobecne.

$p : ax + by + c = 0$, normálový vektor priamky: $\overrightarrow{n_p} = [a, b]$

$q : dx + ey + f = 0$, normálový vektor priamky: $\overrightarrow{n_q} = [d, e]$

a) ak normálové vektory priamok p a q sú rovnaké až na násobok, t.j. ak $ae = bd$, tak tieto priamky sú buď totožné alebo rovnobežné.

O tom, ktorá z týchto možností nastane, rozhodneme porovnaním absolútnych členov v rovnici priamok. Ak $c = f$, priamky sú totožné, inak sú rovnobežné.

b) ak normálové vektory priamok neurčujú rovnaký smer, tak sú priamky rôznobežné a hľadáme ich prienik:

Z rovníc priamok eliminujeme premennú x a vyjadríme si premennú y : $y = \frac{af-cd}{bd-ae}$. Dosadením tejto premennej do rovnice priamky p a vyjadrením premennej $x$dostaneme $x-$ovú a $y-$ovú súradnicu prieniku.

$x = -\frac{b}{a}\left(\frac{af-cd}{bd-ac}\right) - \frac{c}{a}$

$y = \frac{af-cd}{bd-ac}$

Algoritmus:

1. ak $ae = bd$ a $c = f$, potom totožné
2. ak $ae = bd$ a $c \neq f$, potom rovnobežné
3. inak

ak $a \neq 0$, $x = -\frac{b}{a}\left(\frac{af-cd}{bd-ae}\right) - \frac{c}{a}$

ak $a = 0$, $x = -\frac{e}{d}\left(\frac{af-cd}{bd-ae}\right) - \frac{f}{d}$

$y = \frac{af-cd}{bd-ae}$, prienik

3. Nech priamka p je daná všeobecne a priamka q parametricky.
$p : ax + by + c = 0$, normálový vektor priamky: $\overrightarrow{n_p} = [a, b]$
$q : X = Q + s\overrightarrow{v}$, $s \in (-\infty, \infty)$ t.j. $x = q_x + sv_x$
$y = q_y + sv_y$, $s \in (-\infty, \infty)$

a) ak normálový vektor priamok p je kolmý na smerový vektor priamky q t.j. ak ich skalárny súčin je nulový, čiže $av_y + bv_x = 0$, tak tieto priamky sú buď totožné alebo rovnobežné.

O tom, ktorá z týchto možností nastane, rozhodneme dosadením súradníc bodu Q ležiaceho na priamke q do rovnice priamky p. Ak $aq_x + bq_y + c = 0$, priamky sú totožné, inak sú rovnobežné.

b) ak normálový vektor priamky p a smerový vektor priamky q nie sú kolmé, tak sú priamky rôznobežné a hľadáme ich prienik:

Do rovnice priamky p dosadíme za x a y rovnice z priamky q a vyjadríme si parameter s: $s = \frac{-aq_x-bq_y-c}{av_x+bv_y}$. Dosadením tejto hodnoty do rovnice priamky q dostaneme x-ovú a y-ovú súradnicu prieniku.

$x = q_x + \frac{-aq_x-bq_y-c}{av_x+bv_y}v_x$
$y = q_y + \frac{-aq_x-bq_y-c}{av_x+bv_y}v_y$
Algoritmus:
1. ak $av_y + bv_x = 0$ a $aq_x + bq_y + c = 0$, potom totožné
2. ak $av_y + bv_x = 0$ a $aq_x + bq_y + c \neq 0$, potom rovnobežné
3. inak
$x = q_x + \frac{-aq_x-bq_y-c}{av_x+bv_y}v_x$
$y = q_y + \frac{-aq_x-bq_y-c}{av_x+bv_y}v_y$, prienik □

Cvičenie 2.3 (15 bodov). *Opíšte čo najefektívnejšiu procedúru na zistenie veľkosti orientovaného uhla dvoch priamok $\mathbb{E}^2$ daných parametricky. Ošetrite všetky prípady.*

Riešenie (Martina Bátorová, 26.10.2008): Označme priamky zo zadania a, b. Sú zadané parametricky, teda v tvare

$$\begin{aligned} a &= A + t\vec{\mathbf{u}} \\ b &= B + s\vec{\mathbf{v}}, \end{aligned}$$

pričom A je bod priamky a a $\mathbf{u} = (u_1, u_2)$ je jej smerový vektor, rovnako B je bod priamky b a $\mathbf{v} = (v_1, v_2)$ je jej smerový vektor; $s, t \in \mathbb{R}$.

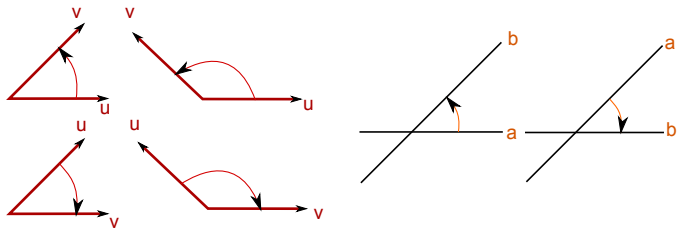
(Neorientovaný) uhol φ dvoch vektorov definujeme:

$$\varphi = \arccos \frac{|\vec{\mathbf{u}} \cdot \vec{\mathbf{v}}|}{|\vec{\mathbf{u}}||\vec{\mathbf{v}}|}.$$

Orientovaný uhol definujeme veľmi podobne, v tomto prípade však záleží na orientácii bázy $(\vec{\mathbf{u}}, \vec{\mathbf{v}})$ (t.j. na poradí, v akom vektory uvažujeme):

$$\varphi_{or} = \arccos \frac{\vec{\mathbf{u}} \cdot \vec{\mathbf{v}}}{|\vec{\mathbf{u}}||\vec{\mathbf{v}}|}.$$

Na obrázku sú príklady orientovaných uhlov vektorov a priamok.



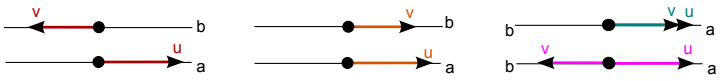
(Ne)orientovaný uhol priamok môžeme vypočítať pomocou toho istého vzorca, avšak musíme uvažovať iný obor hodnôt než pri (ne)orientovanom uhle dvoch vektorov:

(1) vektory: $\varphi \in [0, \pi]$ a $\varphi_{or} \in (-\pi, \pi]$
(2) priamky: $\varphi \in [0, \frac{\pi}{2}]$ a $\varphi_{or} \in (-\frac{\pi}{2}, \frac{\pi}{2}]$.

Označme $D := \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix}$. Môže nastať niekoľko možností:

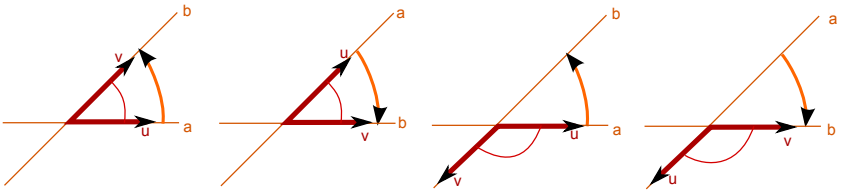
(1) Ak $D = 0$, potom vektory $\vec{\mathbf{u}}, \vec{\mathbf{v}}$ sú lineárne závislé, teda jeden je násobkom druhého. Vtedy sú priamky buď totožné alebo rovnobežné podľa toho, či A patrí priamke b (resp. B priamke a) alebo nie.
   (a) Ak $\vec{\mathbf{u}} = c\vec{\mathbf{v}}, c \in \mathbb{R}^+$, teda $\vec{\mathbf{u}} \cdot \vec{\mathbf{v}} > 0$, orientovaný uhol vektorov je 0.
   (b) Ak $\vec{\mathbf{u}} = -c\vec{\mathbf{v}}, c \in \mathbb{R}^+$, teda $\vec{\mathbf{u}} \cdot \vec{\mathbf{v}} < 0$, orientovaný uhol vektorov je π.

   V oboch prípadoch je orientovaný uhol priamok $\varphi_{or} = 0$.
   Možné polohy lineárne závislých vektorov a priamok nimi určených:



(2) Ak $D \neq 0$, potom vektory $\vec{\mathbf{u}}, \vec{\mathbf{v}}$ sú lineárne nezávislé a priamky sú rôznobežné. Orientovaný uhol vektorov potom zistíme z vyššie spomínaného vzorca na výpočet orientovaného uhla. Orientovaný uhol priamok φ_{or} potom vypočítame nasledovne:
   (a) Ak $D > 0 \wedge \varphi \in [0, \frac{\pi}{2}]$, potom $\varphi_{or} = \varphi$.
   (b) Ak $D > 0 \wedge \varphi \in (\frac{\pi}{2}, \pi]$, potom $\varphi_{or} = \varphi - \pi$.
   (c) Ak $D < 0 \wedge \varphi \in [0, \frac{\pi}{2}]$, potom $\varphi_{or} = -\varphi$.

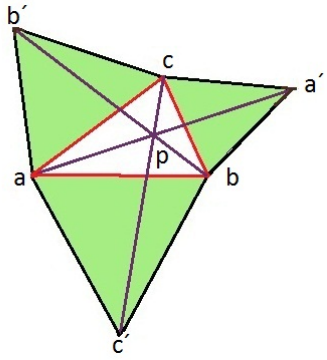
(d) Ak $D < 0 \wedge \varphi \in (\frac{\pi}{2}, \pi]$, potom $\varphi_{or} = \pi - \varphi$.
Jednotlivé situácie sú nakreslené na obrázku:



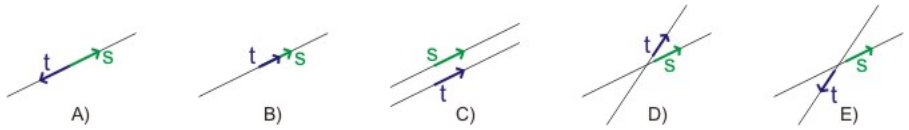
□

Riešenie (Marta Režnáková, 30.10.2008):
Pracujeme v E^2, čiže máme štyri možné výstupy: žiadne riešenie v prípade rovnobežných, 0°alebo 180°v prípade totožných v závislosti od ich smeru a orientovaný uhol v prípade rôznobežných priamok. Priamky máme zadané parametricky:
p_1: x = a_1 + c_1*s, y = a_2 + c_2*s, $s \in (-\infty, \infty)$
p_2: x = b_1 + d_1*t, y = b_2 + d_2*t, $t \in (-\infty, \infty)$

Algoritmus:
if($(c_1$*$d_2) == (c_2$*$d_1)$){ *//priamky sú buď rovnobežné alebo totožné, t.j. ich smerové vektory sú lineárne závislé*
if($((b_1$-$a_1)$*$c_2) == ((b_2$-$a_2)$*$c_1)$){ *//priamky sú totožné, v podmienke sa overuje, či bod druhej priamky leží v prvej priamke*
if($((c_1 * d_1) < 0)||((c_2 * d_2) < 0)$){ *//priamky majú opačný smer, viď obrázok A*
vráť 180°;
}else{vráť 0°;} *//priamky majú rovnaký smer, viď obrázok B*
}else{vráť: priamky nezvierajú žiaden uhol;} *//priamky sú rovnobežné, viď obrázok C*
}else if($((c_1 == 0)$ && $(c_2 == 0))$ || $((d_1 == 0)$ && $(d_2 == 0))$){ *//ak sú smerové vektory priamok nulové, resp. aspoň jeden z nich*
vráť chybovú hlášku;
}else{ *//priamky sú rôznobežné, aplikujeme vzorec pre výpočet orientovaného uhla:*
$\angle p_1p_2 = \arcsin \frac{\det(c,d)}{|c||d|}$; $c = (c_1, c_2)$, $d = (d_1, d_2)$ *//kde* $\det(c, d)$ *je determinant zo súradníc smerových vektorov vzhľadom na kladnú ortonormálnu bázu*
vráť uhol $\angle p_1p_2 = \arcsin \frac{c_1*d_2-c_2*d_1}{|c||d|}$; *//veľkosť uhla bude z intervalu (-*$\frac{\pi}{2}$*,* $\frac{\pi}{2}$*], kde znamienko vyjadruje kladnú, viď obrázok D, resp. zápornú orientáciu, viď obrázok E*
}



Obr. 2.4. Zostrojenie bodu p s uhlami 120 stupňov medzi polpriamkami k vrcholom a, b, c.



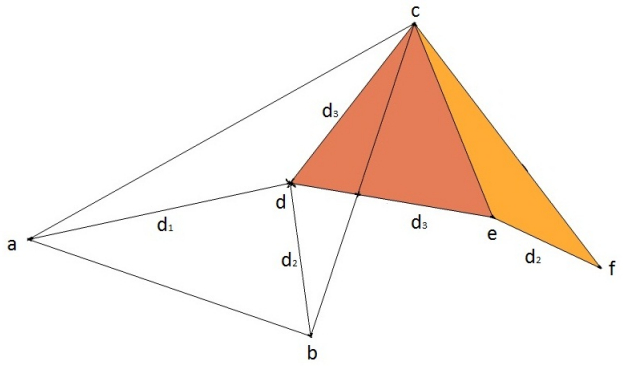
□

Cvičenie 2.4 (35 bodov). *V ľubovoľnom trojuholníku* **abc** *nájdite bod* **p** *taký, že* $|\mathbf{ap}| + |\mathbf{bp}| + |\mathbf{cp}|$ *je najmenšie.*

Riešenie (Silvia Makúchová, 23.11.2010): Budeme rozlišovať 2 prípady trojuholníkov. Prvým bude trojuholník, ktorého žiadny uhol nieje väčší ako 120° (1. prípad) a trojuholník, ktorého uhol je väčší ako 120° (2. prípad). Predpoklad (1. prípad) : Aby bola vzdialenosť $|ap| + |bp| + |cp|$ najmenšia, musí byť splnená vlastnosť, že uhly apb, bpc, cpa sú rovnaké a teda každý z nich má 120°. Dôvod si ukážeme neskôr. Bod p zostrojíme nasledovne. Nad každou stranou trojuholníka abc zostrojíme rovnostranný trojuholník. Máme teda trojuholníky $ac'b$, $ba'c$ a $cb'a$. Keď spojíme body aa', bb', cc' dostávame bod p (obr. 2.4).

Dôkaz:

Vezmeme si ľubovoľný trojuholník abc a v ňom bod d (obr. 2.5). Označme dĺžky d_1, d_2, d_3 nasledovne : $d_1 = |ad|$, $d_2 = |bd|$, $d_3 = |cd|$. Zostrojme rovnostranný trojuholník cde ako je na obrázku. Ďalej zostrojme trojuholník cef taký, aby bol zhodný s trojuholníkom cdb. Je zrejmé, že trojuholník cef je vlastne len otočením trojuholníka cdb o 60°. Takouto konštrukciou teda vždy dostaneme rovnostranný trojuholník cfb, kedže uhol bcf je 60° a strany cb a cf sa rovnajú. Z obrázku vidíme, že vzdialenosť $|de|$ je na začiatku zvolená vzdialenosť d_3, teda $d_3 = |de|$, podobne $d_2 = |ef|$ a $d_1 = |ad|$. Máme teda lomenú čiaru $adef$. Aby bola táto vzdialenosť čo najmenšia, volíme lomenú čiaru $adef$ ako úsečku. Úsečka $adef$ je úsečka aa' z predpokladu vo vete. Teda vidíme, že bod d leží na úsečke $adef$ (aa'). Analogicky postupujeme pri každom vrchole a teda bod d leží aj na bb' a cc'.



Obr. 2.5. Dôkaz minimality súčtu vzdialeností v prípade, že uhly sú menšie ako 120 stupňov.

(2. prípad)

Pre trojuholník, ktorého uhol je väčší ako 120° platí, že bod p bude totožný s bodom, pri ktorom tento uhol je.

Dôkaz :

Majme trojuholník abc (obr. 2.6 vpravo). Označme $d_1 = |ap|$, $d_2 = |bp|$, $d_3 = |cp|$. Zostrojme rovnostranný trojuholník acd ako je to na obrázku. Ďalej zostrojme rovnostranný trojuholník ape. Vidíme, že trojuholník aed je vlastne len otočením trojuholníka apc o 60°. Máme : $d_3 = |de|$, $d_1 = |ea|$.

Označme $d(p) = |bp| + |pe| + |ed|$. (dĺžka lomenej čiary $bped$), čo je vlastne $d_1 + d_2 + d_3$.

Označme $d(a) = |ab| + |ad|$, $d(a) < d(p)$, pre ľubovoľný bod p z trojuholníka. Vezmime bod p mimo trojulníka (obr. 3 vľavo). Potom existuje bod p' na hranici taký, že $d(p') < d(p)$, $(d(p') = |ap'| + |bp'| + |cp'|)$. To znamená, že pre ľubovoľný bod p mimo trojuholníka existuje bod p' na hranici taký, že platí : $d(p') < d(p)$.

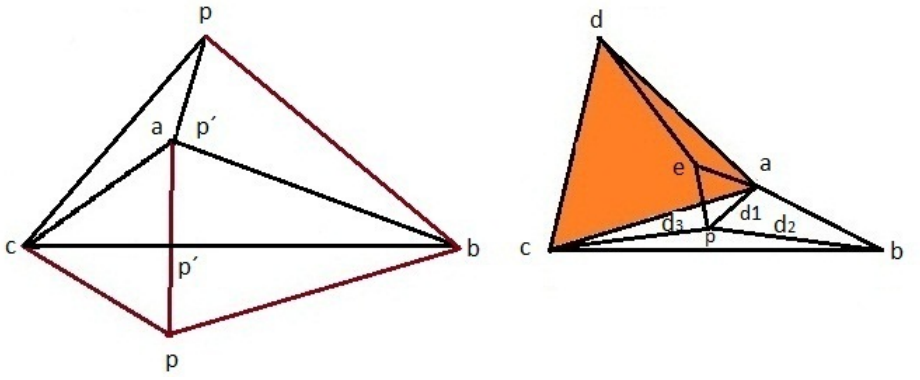
Pre bod p' z hranice trojuholníka máme $d(p') < d(p)$ a $d(a) < d(p')$ (trojuholníková nerovnosť) pre každý bod p mimo trojuholníka. Teda $d(a) < d(p)$, čo znamená, že bod p musí byť totožný s bodom a.

□

Riešenie (Gabriel Foltán, 22.10.2011): Riešenie je založené na predpoklade, že bod **p** existuje a je jedinný. Ďalej využívame vlastnosť, že ak sa posunie bod **a** po polpriamke $\overrightarrow{\mathbf{pa}}$, tak pozícia bodu **p** nezmení. Presvedčme sa, že to naozaj platí:

Sporom. Posuňme pozíciu bodu **a** pozdĺž polpriamky $\overrightarrow{\mathbf{pa}}$. Vznikne nám bod, označme si ho $\mathbf{a}'$ (obr. 2.7). Predpokladajme, že bod **p** potom zmení svoju pozíciu, označme si ju $\mathbf{p}'$. Potom platí:

$$|\mathbf{ap}| + |\mathbf{bp}| + |\mathbf{cp}| < |\mathbf{ap}'| + |\mathbf{bp}'| + |\mathbf{cp}'| \tag{2}$$



Obr. 2.6. Prípad uhla nad 120 stupňov

Rovnako pre trojuholník $\mathbf{a'bc}$ podľa predpokladu, platí:

$$|\mathbf{a'p'}| + |\mathbf{bp'}| + |\mathbf{cp'}| < |\mathbf{a'p}| + |\mathbf{bp}| + |\mathbf{cp}| \tag{3}$$

Po úprave sústavy nerovníc (2),(3) dostávame:

$$|\mathbf{ap}| + |\mathbf{a'p'}| < |\mathbf{a'p}| + |\mathbf{ap'}|$$

Kedže bod $\mathbf{a'}$ bol na polpriamke $\overrightarrow{\mathbf{pa}}$ vybratý tak, aby $|\mathbf{a'p}| + |\mathbf{aa'}| = |\mathbf{ap}|$, potom po úprave dostávame, že

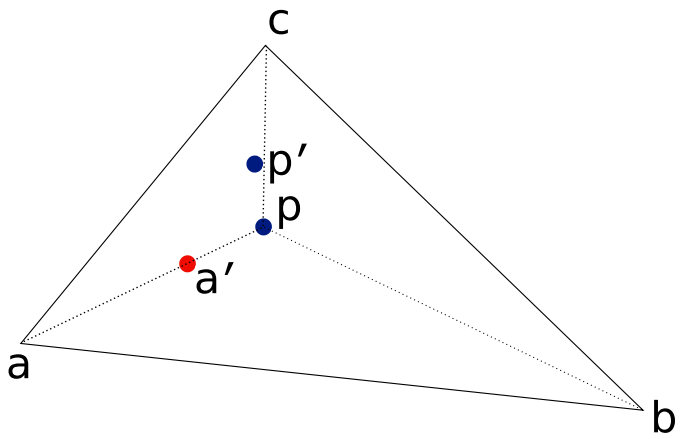
$$|\mathbf{aa'}| + |\mathbf{a'p'}| < |\mathbf{ap'}|$$

Čo je spor.

Využijúc tento predpoklad, hľadajme bod **p**:
Posuňme body **a** a **b** tak, aby trojuholník **abc** bol rovnostranný. To sa dá vždy. Napríklad, začnime posúvať bod **a**, tak aby platilo, že $|\mathbf{ab}| = |\mathbf{bc}|$. Trojuholník **abc** sa zmení na rovnoramenný. Je zrejmé, že pre rovnoramenný trojuholník platí, že bod **p** leží na osi symetrie tohto trojuholníka. Z tohto dôsledku, kdekoľvek na osi bude pozícia bodu **b**, trojuholník zastane rovnoramenný. Posuňme ho preto tak, aby trojuholník **abc** bol rovnostranný. Hľadaný bod **p** potom bude stredom opísanej kružnice ku tomuto rovnostranému trojuholníku. Pretože úsečky spájajúce bod **p** s vrcholmi ronvostranného trojuholníka **abc** zvierajú uhol 120°, rovnaký uhol musia zvierať pre ľubovoľný trojuholník, pre ktorý bod **p** existuje.

□

Riešenie (Gabriel Foltán, 28.12.2011):Úlohu vyriešime pomocou lokálne viazaných extrémov, konkrétne metódou Lagrangeovej funkcie a Lagrangeových multiplikátorov a použitím kosínusovej vety. Majme trojuholník **abc** (obr. 2.8). Dĺžky jeho strán sú A, B ,C. Z hľadaného bodu **p**, vedme polpriamky cez vrcholy trojuholníka a označme si ich $l_{\mathbf{a}}$,$l_{\mathbf{b}}$,$l_{\mathbf{c}}$. Dĺžky troch úsečiek k daným vrcholom z bodu **p**, si označíme



OBR. 2.7. Znázornenie situácie v trojuholníku

x, y, z, kde úsečka s dĺžkou x leží na $l_{\mathbf{a}}$, y na $l_{\mathbf{b}}$ a z na $l_{\mathbf{c}}$. Nech polpriamky $l_{\mathbf{a}}$,$l_{\mathbf{b}}$, zvieraju uhol α, polpriamky $l_{\mathbf{b}}$,$l_{\mathbf{c}}$ uhol β. Potom uhol medzi $l_{\mathbf{a}}$,$l_{\mathbf{c}}$ je $2\pi - \alpha - \beta$. Naša zadaná funkcia, ktorej viazaný extrém hľadáme(konkrétne minimum) má tvar

$$f(x, y, z) = x + y + z.$$

Prvá väzbová funkcia zohľadnuje kosínusovú vetu v trojuholníku **abp** aplikovanú na dĺžky jehostrán x, y, a A. Má tvar

$$g_1(x, y, \alpha) = x^2 + y^2 - 2xy\cos(\alpha) - A^2$$

Analogicky vytvoríme ďalšie dve väzbové funkcie g_2, g_3 pre trojuholníky **cbp** a **acp**.

Potom výsledná Lagrangeova funkcia má tvar

$$\begin{aligned} L(x, y, z, \lambda_1, \lambda_2, \lambda_3, \alpha, \beta) = x + y + z + \\ \lambda_1(x^2 + y^2 - 2xy\cos(\alpha) - A^2) + \\ \lambda_2(y^2 + z^2 - 2yz\cos(\beta) - B^2) + \\ \lambda_3(z^2 + x^2 - 2zx\cos(\alpha + \beta) - C^2) \end{aligned}$$

V poslednom riadku, sme využili fakt, že funkcia kosínus je periodická a párna a teda $\cos(2\pi - \alpha - \beta) = \cos(\alpha + \beta)$. Počítajme parciálne derivácie našej Lagrangeovej funkcie podľa premenných x,y,z,α,β a položme ich

rovné nule. Dostávame systém piatich rovníc :

$$\frac{\partial L}{\partial x} = 1 + \lambda_1(2x - 2y\cos(\alpha)) + \lambda_3(2x - 2z\cos(\alpha+\beta)) = 0$$
$$\frac{\partial L}{\partial y} = 1 + \lambda_1(2y - 2x\cos(\alpha)) + \lambda_2(2y - 2z\cos(\beta)) = 0$$
$$\frac{\partial L}{\partial z} = 1 + \lambda_2(2z - 2y\cos(\beta)) + \lambda_3(2z - 2x\cos(\alpha+\beta)) = 0$$
$$\frac{\partial L}{\partial \alpha} = \lambda_1 y\sin(\alpha) + \lambda_3 z\sin(\alpha+\beta) = 0$$
$$\frac{\partial L}{\partial \beta} = \lambda_2 y\sin(\beta) + \lambda_3 x\sin(\alpha+\beta) = 0$$

Posledné dve rovnosti sú už upravené, s prihliadnutím na to, že x, y, z sú nenulové dĺžky. (V prípade, že jedna z dĺžok by bola nulová, hľadaný bod $\mathbf{p}$ by musel byť vrcholom trojuholníka, ktorý tvorí tvorí danú usečku. V riešení tohto cvičenia od Sylvie Makúchovej je ukázané, že hľadaný bod $\mathbf{p}$ je skutočne tým vrcholom.)

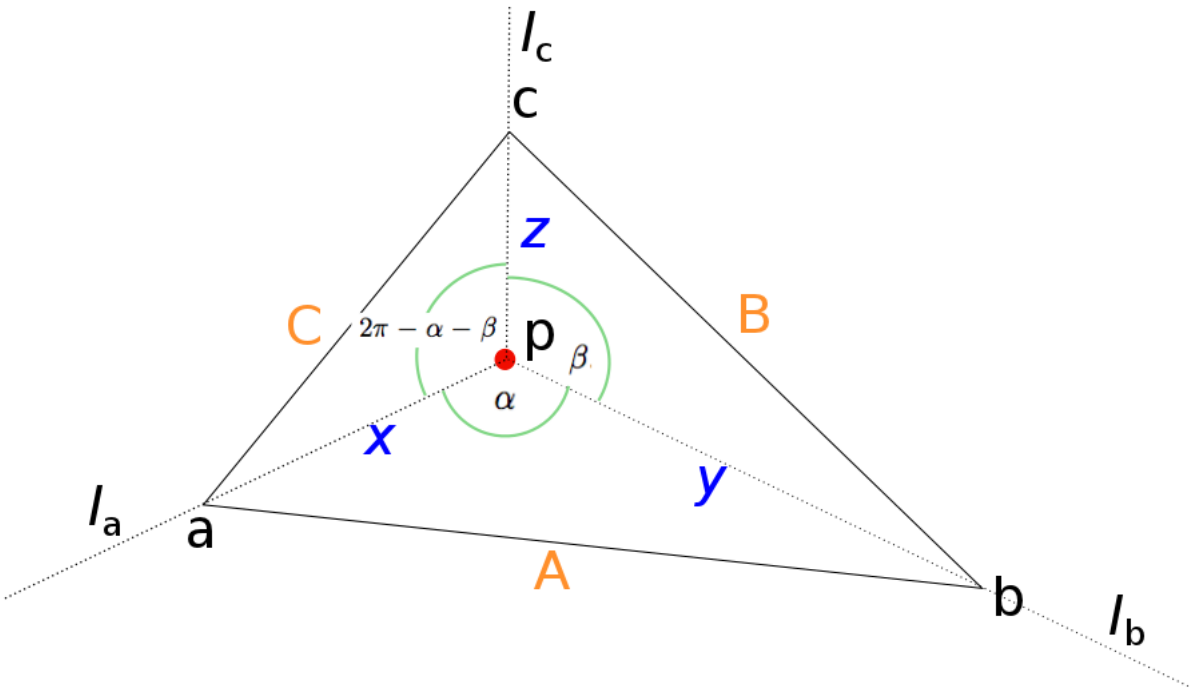
Po úpravách dostávame vzťahy

$$\sin(\alpha) = \sin(\beta)$$
$$\sin(\alpha+\beta) = -\sin(\beta)$$

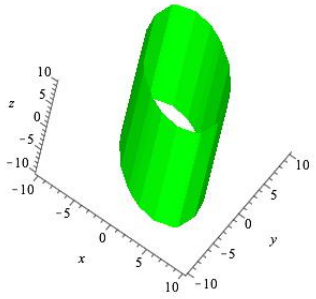
Odkiaľ dostávame $\alpha = \beta = 120°$. To znamená, že hľadaný bod musí spĺnať podmienku, že uhly medzi $l_{\mathbf{a}}$,$l_{\mathbf{b}}$, medzi $l_{\mathbf{a}}$,$l_{\mathbf{c}}$, a medzi $l_{\mathbf{b}}$,$l_{\mathbf{c}}$ musia byť rovnaké a rovné 120°.

V prípade, že pri jednom vrchole trojuholníka je uhol väčší než 120°, tak bod $\mathbf{p}$ je tým vrcholom. Dôkaz tohto faktu je uvedený v riešení tohto cvičenia od Sylvie Makúchovej(pozri vyššie), konkrétne v jej riešení je to uvedené ako (2. prípad).

Vizualizujme predošlý postup pre rovnostranný trojuholník s dĺžkou strany $A = B = C = 6$. Hľadáme extrém funkcie f na množine hodnôt x, y, z, ktorá musí vyhovovať všetký trom väzbovým funkciam g_1, g_2, g_3. Samotné väzbové funkcie nám predstavujú v priestore tri "sploštené"valce (napr. na obrázku 2.9 pre funkciu g_1). Teda hľadáme množinu bodov prieniku týchto troch "valcov". Ukázali sme, že $\alpha = \beta = 120°$ a zatiaľ predpokladáme x, y, z sú nenulové, kedže máme daný rovnostranný trojuholník. Tým sa definičný obor našich objektov obmedzí iba na 1. oktant a následne i samotný prienik budeme hľadať v prvom oktante. Ukazuje sa, že prienik týchto objektov vzhľadom na stanovené podmienky je iba jediný bod. Kedže máme rovnostranný trojuholník vieme, že hladaný pod $\mathbf{p}$ je stredom opísanej kružnice s polomerom r, teda vieme, že hľadané riešenie bude $r = x_0 = y_0 = z_0 = 2\sqrt{3}$. Skutočne, prienik našich troch valcov po aplikovaní daných podmienok(1.oktant) je daný bod, ktorý je zároveň aj riešením. Funkcia $f(x, y, z)$ sa nedá predstaviť, avšak my už riešenie poznáme, tak potom $x + y + z = 6\sqrt{3}$. To nám definuje rovinu. Zobrazme teda výsledok



OBR. 2.8. Znázornenie situácie v trojuholníku



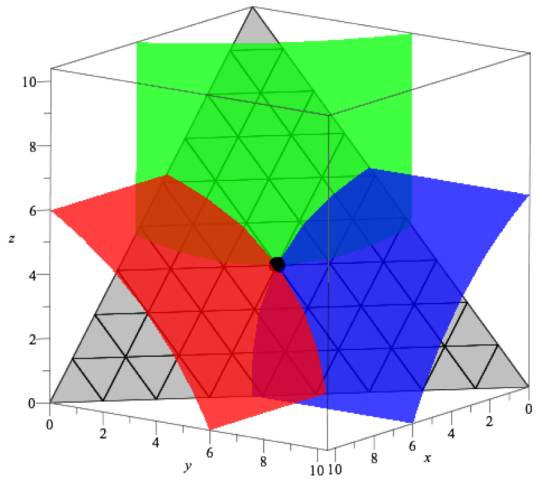
OBR. 2.9. väzbová funkcia g_1 a jej význam v priestore

do obrázka č.2.10. Všetky objekty v obrázku orežeme *bouding boxom*, určeným podmienkami $x \in < 0, 6\sqrt{3} >, y \in < 0, 6\sqrt{3} >, z \in < 0, 6\sqrt{3} >$. Vidíme, že hľadaný bod skutočne predstavuje **p**.

□

CVIČENIE 2.5 (30 bodov). *Opíšte dátovú štruktúru „spájateľná fronta" a operácie vkladania, vyberania, spájania a rozdeľovania na nej pomocou vhodne zvolenej stromovej dátovej štruktúry.*

Riešenie (Matej Hudák, 25.11.2010):



Obr. 2.10. prienik(čierna bodka) troch valcov (zelený, červený, modrý) v prvom oktante a orezaná rovina $x + y + z = 6\sqrt{3}$, ktorá predstavuje daný rovnostranný trojuholník

Spájateľná fronta je dátová štruktúra, ktorá dokáže pracovať s nasledujúcimi inštrukciami: vkladanie do fronty, vyberanie z fronty, nájdenie vrchola vo fronte, spájanie a rozdelenie fronty. Tieto inštrukcie pracujú pritom v čase $\mathcal{O}(log\, n)$, pričom n je počet vrcholov uložených vo fronte. Podľa zadania následne definujeme implementáciu vkladania, vymazania, spojenia a rozdelenia na fronte.

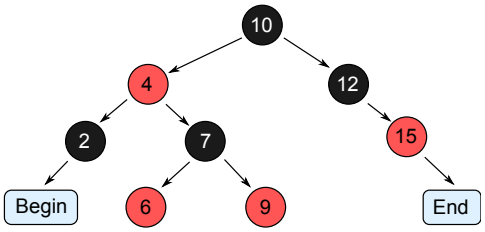
Na realizáciu spájateľnej fronty použijeme červeno-čierne stromy (ďalej RB-strom). RB-strom je binárny vyhľadávací strom, v ktorom každý uzol je rozširený naviac o jeden bit informácie reprezentujúci farbu tohto vrchola, ktorá môže byť buď červená alebo čierna. Zavedením určitých pravidiel, podľa ktorých možno vrcholy na ľubovoľnej z ciest od koreňa k listom ofarbiť iba určitým spôsobom, je možné zaistiť aby žiadna takáto cesta nebola viac ako dvakrát tak dlhá ako ktorákoľvek iná, čiže aby bol strom aspoň približne vyvážený.

Výhodou RB-stromov je práve ich vyváženosť, ktorá zaručuje, že budú časy behu týchto operácií nanajvýš rovné $\mathcal{O}(log\, n)$ v najhoršom prípade, kde n je počet vrcholov RB-stromu. Vlastnosti RB-stromov:

(1) Každý vrchol je buď červený alebo čierny.

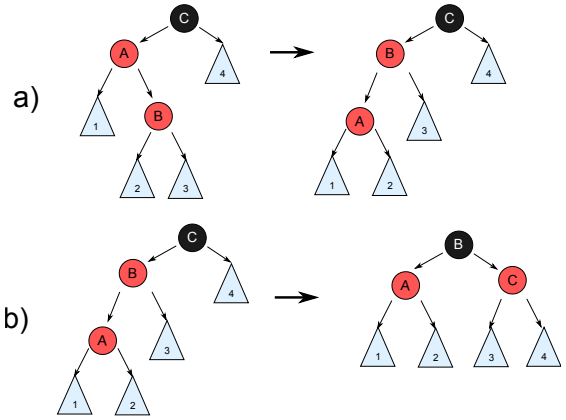
(2) Každý list (*nil*) je čierny. List považujeme za vrchol bez ďalších synov, pričom *nil* je pojem, ktorý používame pri rogramovaní na označenie listu.
(3) Čiernu výšku vrchola x definujeme ako počet čiernych vrcholov od tohto vrchola po list.
(4) Ak je vrchol červený, obaja jeho synovia sú čierni.
(5) Každá cesta z vrchola do ľubovoľného listu, ktorý je jeho nasledovníkom, má rovnaký počet čiernych vrcholov (pravidlo čiernych).

Výška stromu, ktorý obsahuje n vrcholov je nanajvýš $2log(n+1)$. Teda nájsť určitý element trvá $\mathcal{O}(log\, n)$ pričom obsahuje $\mathcal{O}(log\, n)$ porovnaní.

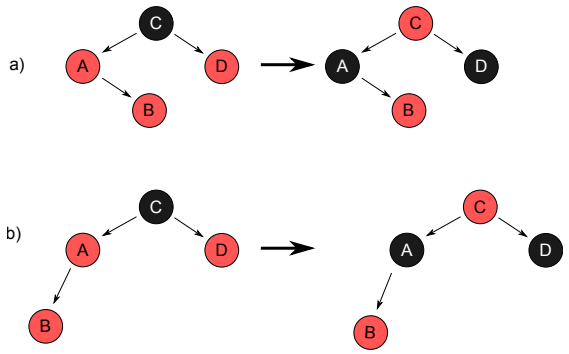


Obr. 2.11. Príklad *RB*-stromu

Pri vkladaní alebo vymazávaní vrchola z nášho stromu môže dočasne dôjsť k porušeniu rovnováhy. Preto aplikujeme lokálne rotácie (na obrázku 2). Rotácie menia konfiguráciu vrcholov vľavo na konfiguráciu vrcholov vpravo. Pracujeme z konštantným počtom vrcholov a teda v konštantom čase $\mathcal{O}(1)$, keďže nevyžadujeme žiadne porovnávania a meníme len smerníky na jednotlivé vrcholy. Takýchto situácií môže nastať niekoľko. Ďalej aplikujeme prefarbovanie vrcholov (na obrázku 3) s výsledkom opätovného dosiahnutia rovnováhy.



Obr. 2.12. Prvá a druhá rotácia. Objekty označené trojuholníkmi sú ďalšie možné podstromy.



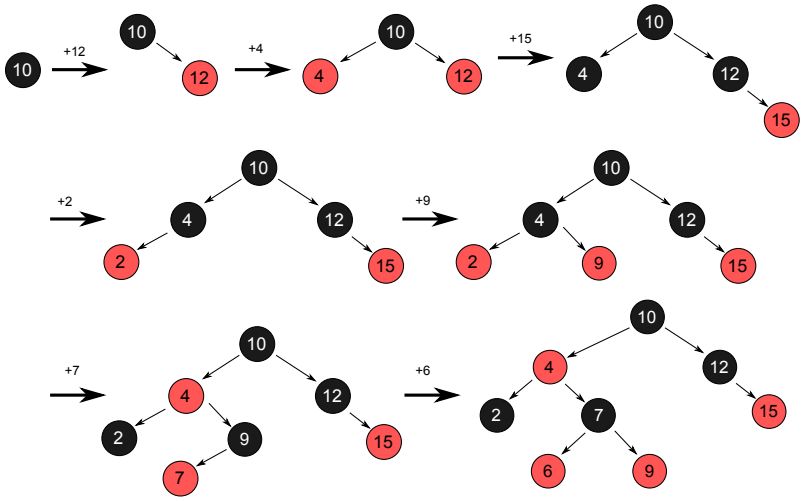
Obr. 2.13. Jednoduchý prípad prefarbenia

Vloženie vrchola

Nový vrchol x vždy vložíme ako červený vrchol ako do klasického BVS. Ak treba obnoviť vlastnosti stromu tak prefarbujeme ostatne vrcholy a robíme jednotlivé rotácie. Vloženie vrchola x do fronty Q možno uskutočniť v čase $\mathcal{O}(log\, n)$. Na obrázku 4 je znázornený proces vloženia. Vloženie vrchola môžeme popísať nasledujúcim pseudokódom:

```
insert(Q, x)
{
    color(x, RED);
    while(x != getRoot(Q))and(color(x.parent) = RED)do//porušenie vlastnosti 3 presúvame smerom ku koreňu, pričom zachovávame vlastnosť 4
        if(x.parent == x.parent.parent.left) //ak otec x je ľavým synom svojho otca
        {
            y = x.parent.parent.right;
            if(getColor(y) == RED)//riesime pripad vyvazenia prefarbením
            {
                color(x.parent, BLACK);
                color(y, BLACK);
                color(x.parent.parent, RED);
                x = x.parent.parent;
            }
            else if(x == x.parent.right)//riesime pripad vyvazenia ratáciami
            {
                x = x.parent;
                leftRotate(Q, x);
                color(x.parent, BLACK);
                color(x.parent.parent, RED);
                leftRotate(Q, x.parent.parent);
            }
        }
        else//ak otec x je pravým synom svojho otca
        {
            //podobne len treba vymeniť right a left
```

```
        }
    color(getRoot(Q), BLACK);
}

//funkcia leftRotate() použitá vyššie
leftRotate(Q, x)
{
    y = x.right; //definujeme y
    x.right = y.left;//ľavý podstrom y je teraz pravým podstromom x
    if(y.left != nil)
        y.left.parent = x;
    y.parent = x.parent;//otcom x bude teraz otec y
    if(x.parent == nil)
        root(Q, y);
    else if(x == x.parent.left)
        x.parent.left = y;
    else
        x.parent.right = y;
    y.left = x;//vrchol x je teraz ľavým synom y
    x.parent = y;//vrchol y je potom otcom x
}
```

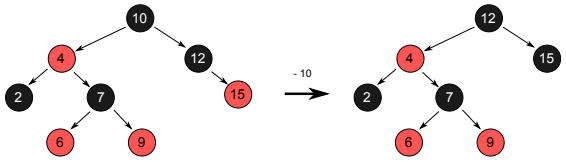


Obr. 2.14. Operácia vkladanie

Vymazanie vrchola

Podobne ako ostatné operácie, aj vymazávanie vrchola trvá $\mathcal{O}(log\, n)$. Vrchol vymažeme podobne ako z BVS. Ak je potrebné vyvažovať frontu po vymazaní nejakého vrchola tak opäť vyrovnáme frontu pomocou vyššie uvedených rotacií a prefarbovania vrcholov. Upravovať

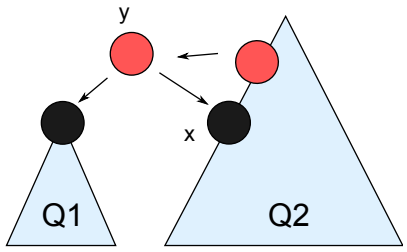
začíname v synovi vymazaného vrchola, alebo ak nemal žiadneho syna tak v príslušnom liste (*nile*).



Obr. 2.15. Príklad vymazania vrchola 10

Spojenie

Pri dvoch zadaných frontách Q_1 a Q_2 takých, že pre každé $x \in Q_1$ a $y \in Q_2$ je $x \leq y$, je možné spojiť Q_1 a Q_2. Na túto operáciu použijeme *spojenie* (*concatenate*).



Obr. 2.16. Spojenie dvoch front

Potom môžeme pseudokód zapísať nasledovne ($concatenate(x, Q)$):

(1) Prejdime od koreňa Q_2 až po listy po najľavejšej strane až kým nenájdeme čierny vrchol x taký, že čierna výška tohto vrchola je rovnaká ako čierna výška Q_1.
(2) Definujme otca tohto čierneho vrchola ako p.
(3) Vytvorme nový červený vrchol y a pomocou neho spojme koreň Q_1 a x ako dvoch synov, teda súrodencov vrchola y.
(4) Ak p je list (*nil*), koreňom Q_1 bude vrchol y. Inak bude otcom y vrchol p a koreňom Q_1 bude koreň Q_2.
(5) Aplikujme opäť vyváženie novej fronty od vrchola y.
(6) Nastavme koreň Q_2 na *nil*, aby Q_2 bolo prázdne.

Na zložitosť tejto operácie vplýva hlavne krok 1. Čas tejto operácie je preto $\mathcal{O}(log\, n_2)$ kde n_2 je veľkosť väčšej fronty z Q_1 a Q_2, pričom n_2 je najviac n, čo je veľkosť fronty Q. Celkový čas oprerácie spojenia je teda $\mathcal{O}(log\, n)$.

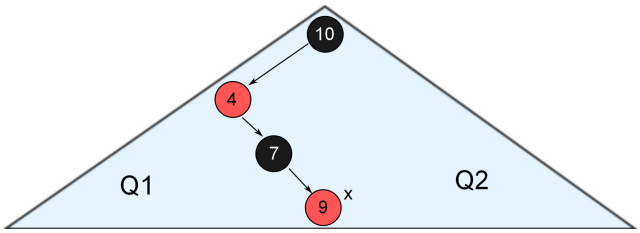
Kroky 3, 4, 5 a 6 zapíšeme takýmto pseudokódom:

```
y.left = root(Q1);
y.right = x;
color(y, RED);
If (p == nil)
```

```
    root(Q1) = y;
else
{
    y.parent = p;
    root(Q1) = root(Q2);
}
//vyvažovanie
root(Q2) = nil;
```

Rozdelenie

Pri zadanej fronte Q a vrcholu x (nemusí patriť ani jednej fronte) rozdelíme Q na Q_1 a Q_2, kde všetky vrcholy z Q_1 sú menšie ako x a všetky vrcholy z Q_2 sú väčšie, alebo rovné x. Na obrázku 7 vidíme príklad takého rozdelenia. Vrcholy v trojuholníku sú podstromy, ďalšie časti, pričom dôležitá je označená cesta od listu ku koreňu.



Obr. 2.17. Rozdelenie fronty Q na Q_1 a Q_2.

Pseudokódom túto operáciu môžme popísať nasledovne:

(1) Definujme dve prázdne fronty Q_1 a Q_2.
(2) Nech p je koreň Q.
(3) Pôjdeme od koreňa a budeme spájať pomocou *concatenate* podčasti fronty do separátnych front Q_1 a Q_2 až kým nenarazíme na vrchol x.
(4) Ak sa ešte zvýšili nejaké vrcholy tak ak sú menšie ako x, vložíme ich do Q_1 pomocou $insert(Q1, p)$. Inak ich vložíme pomocou $insert(Q2, p)$.

Výsledkom sú dve fronty Q_1 a Q_2. Hlavné vnorenie (v kroku 3) si vyžaduje $\mathcal{O}(log\, n)$ operácií spojenia a porovnaní. Keďže vnorením sa budeme pohybovať po najľavejšej strane Q_2 a najpravejľej strane Q_1, každý z týchto operácí môže bude vykonaná v konštantnom čase. Preto je celková zložitosť operácie rozdelenia $\mathcal{O}(log\, n)$. Krok 3. zapíšeme takýmto pseudokódom:

```
while (p != nil) do
{
    if(x < p)
    {
        catenate(p.right, Q2);
```

```
        p = p.left;
    }
    else (x >= p)
    {
        catenate(p.left, Q1);
        p = p.right;
    }
}
```

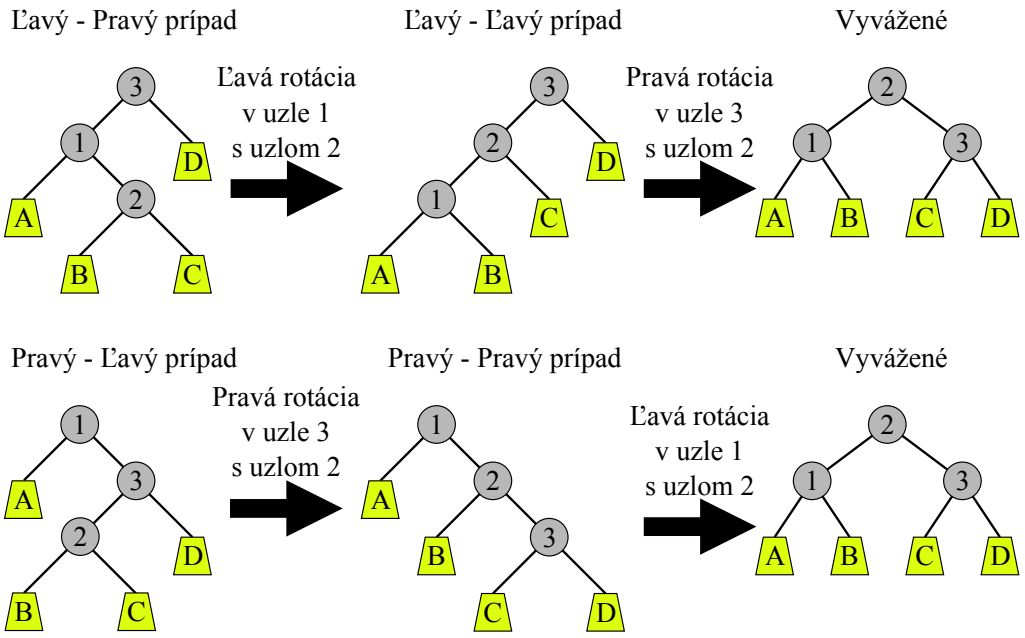
□

Riešenie (Michal Ferko, 30. 10. 2011): Spájateľnú frontu budeme reprezentovať pomocou AVL stromu. AVL strom je samo-vyvažovací binárny vyhľadávací strom, pre ktorý základné slovníkové operácie - hľadanie uzla, vloženie uzla a vymazanie uzla trvá $\mathcal{O}(\log n)$ pre strom s n prvkami. Aby sme ale mali kompletnú spájateľnú frontu, popíšeme ešte aj operácie spájania a rozdeľovania, ktoré budú trvať taktiež $\mathcal{O}(\log n)$.

Najprv teda popíšeme bežný AVL strom. AVL strom si v každom vrchole pamätá prvok, ktorý je v ňom uložený (pre nás to budú prirodzené čísla) a takzvaný *balančný faktor* - číslo $b \in \mathbb{Z}$, ktoré bude používané pri vyvažovaní stromu. Toto číslo pritom bude nadobúdať hodnoty $\{-2, -1, 0, 1, 2\}$. Pri vyváženom strome nadobúda balančný faktor iba hodnoty $\{-1, 0, 1\}$. Tento balančný faktor zodpovedá rozdielu výšok podstromov daného uzla.

Hľadanie uzla: Operácia hľadania uzla má za úlohu zistiť, či daný prvok sa nachádza v našej štruktúre. Vzhľadom na to, že pre AVL strom vždy platí, že v ľavom podstrome prvku sa nachádzajú prvky menšie ako hodnota prvku a v pravom podstrome sa nachádzajú prvky väčšie, je táto operácia pomerne jednoduchá (a identická s operáciou hľadania na binárnom vyhľadávacom strome). Začne sa v koreni stromu a skontroluje sa, či náhodou hodnota v koreni nie je práve naša hľadaná hodnota. Ak je, vrátime tento uzol. Inak rekurzívne spustíme procedúru na ľavého alebo pravého syna a podľa výsledkov zistíme, či sa prvok v strome nachádza.

```
function HladajPrvok(Uzol u, Prvok p)
{
    if (u.Hodnota == p) return NÁJDENÝ UZOL
    if (u.Hodnota < p)
        if (u.LavySyn != NULL) return HladajPrvok(u.LavySyn, p)
        else return PRVOK NEBOL NÁJDENÝ
    if (u.Hodnota > p)

        if (u.PravySyn != NULL) return HladajPrvok(u.PravySyn, p)
        else return PRVOK NEBOL NÁJDENÝ
}
```



OBR. 2.18. Možné situácie, ktoré nastávajú pri pridávaní prvkov a treba ich opraviť lebo strom nie je vybalancovaný. Situácie Ľavý-Pravý a Pravý-Ľavý sa transformujú na Ľavý-Ľavý a Pravý-Pravý, ktoré potom vieme jednoducho vyvážiť. Všetko sa pritom deje pomocou rotácii na binárnom strome.

Keďže náš strom bude vyvážený, jeho maximálna výške je $\mathcal{O}(\log n)$, teda maximálny počet volaní funkcie HladajPrvok je $\mathcal{O}(\log n)$.

Vkladanie uzla: Táto operácia dostane ako vstup hodnotu prvku, ktorý má vložiť do nášho stromu. Ak sa ale tento prvok už v strome nachádza, tak to oznámi používateľovi. Najprv sa teda do nášho binárneho vyhľadávacieho stromu vloží prvok a následnou úpravou balančých faktorov sa zistí, či treba strom vyvážiť. Ak áno, vyváži sa. Vyvažovanie sa pritom robí tak, že po pridaní uzla jeho otcovi balančný faktor znížime o 1 (ak je ľavým synom) alebo zvýšime o 1 (ak je pravým synom). Postupujeme ďalej ku koreňu a pričítavame tieto hodnoty. Ak narazíme na prvok, ktorý by po zmene faktoru nadobudol hodnotu $+2$ alebo -2, je treba strom vyvážiť. Situácie, ktoré môžu nastať sú zobrazené na obrázku 2.18.

Pri ceste hore následne upravujeme balančné faktory nasledujúcim spôsobom. Ak nový uzol bol pridaný do ľavého podstromu, tak znížime balančný faktor o 1. Ak nový uzol bol pridaný do pravého podstromu, tak zvýšime balančný faktor o 1. Keď po týchto operáciach je balančný faktor 2 alebo -2, tak treba vyvážiť strom. Nech P je uzol, v ktorom nastala jedna z týchto situácii, a nech L je ľavý syn P a R je pravý syn P. Potom:

(1) Ak P má balančný faktor $+2$, tak podstrom s koreňom R prevažuje podstrom s koreňom L. Treba podľa balančného faktoru v R rozhodnúť, ktorý z prípadov nastal: Pravý-Ľavý ak faktor R je -1, Pravý-Pravý ak faktor R je $+1$.
(2) Ak P má balančný faktor -2, tak podstrom s koreňom L prevažuje podstrom s koreňom R. Treba podľa balančného faktoru v L rozhodnúť, ktorý z prípadov nastal: Ľavý-Pravý ak faktor L je $+1$, Ľavý-Ľavý ak faktor L je -1.

Ako podstromy vyvážime je znázornené na obrázku 2.18, po takejto úprave sa nanovo zrátajú balančné faktory.

Vymazanie uzla: Pri vymazávaní listu alebo prvku s jedným synom nevznikne žiaden problém, tieto operácie sú pomerne jednoduché. Samozrejme sa následne musia upraviť balančné faktory (opačným smerom ako sa upravovali pri vkladaní uzla) a cestou ku koreňu sa strom vyvažuje.

Na rozdiel od vkladania uzla môže nastať situácia, keď faktor v P je 2 (resp. -2) a faktor v R (resp. L) je 0. Takýto prípad vyriešime ľavou (resp. pravou) rotáciou v P s uzlom R (resp. L).

Ak ale vymazávaný prvok má dvoch synov, musíme ho nahradiť iným prvkom (ktorý následne vymažeme z nášho stromu). V čase $\mathcal{O}(\log n)$ sa nájde nasledovník (najmenší prvok jeho pravého podstromu) alebo predchodca (najväčší prvok jeho ľavého podstromu) vymazávaného uzla, jeho hodnota nahradí hodnotu vymazávaného uzla a následne vymazávame prvok, ktorého hodnota nahradila náš vymazaný prvok. Tento prvok môže mať maximálne jedného syna, lebo je buď najpravší alebo najľavší v podstrome, preto ho môžeme vymazať postupom pre list alebo prvok s jedným synom. Operácia vymazávania trvá celkovo $\mathcal{O}(\log n)$ - $\mathcal{O}(\log n)$ trvá nájdenie vymazávaného prvku, $\mathcal{O}(\log n)$ trvá nájdenie predchodcu alebo nasledovníka a potom $\mathcal{O}(\log n)$ trvá vymazanie predchodcu alebo nasledovníka (spolu s vyvažovaním).

Spojenie dvoch AVL stromov: Máme fronty F_1 a F_2, pričom $\forall x \in F_1, y \in F_2$ platí $x \leq y$, čiže všetky prvky z F_1 sú menšie ako hociktorý z prvkov F_2. Pre správne fungovanie algoritmu je vhodné si pre každý strom pamätať jeho výšku, čo zmení predchádzajúce operácie minimálne a nijako nezmení časovú zložitosť. Potom vieme v čase $\mathcal{O}(1)$ získať výšku stromu (ak by sme ju mali nanovo spočítať, trvalo by to $\mathcal{O}(n)$, lebo by sme museli navštíviť každý vrchol stromu raz). Predpokladajme teda, že F_2 je vyšší (druhá situácia je symetrická).

Potrebujeme nájsť vhodný vrchol v F_2, do ktorého zavesíme strom F_1. Najprv vymažeme z F_1 prvok s najväčšou hodnotou, nech je to L. Tento prvok využijeme na vytvorenie nového uzlu v spojenom strome. Je potrebný, lebo vrchol, do ktorého by sme chceli zavesiť F_1 môže mať pravého syna a chcelo by to zložitejšie úpravy stromu, aby sa umožnilo zavesenie F_1. Následne v F_2 zostupujeme vždy do ľavého podstromu až kým nenájdeme taký vrchol R, že výška podstromu s koreňom R je

rovnaká ako výška F_1 po odstránení L. Na miesto R umiestnime prvok L a jeho ľavý podstrom bude F_1 a pravý podstrom bude podstrom určený vrcholom R. Podstrom s koreňom L má výšku o 1 väčšiu ako R, preto zvýšime balančný faktor otca novovzniknutého vrcholu L o 1. Keďže sa zmenil balančný faktor, treba strom znovu vyvážiť z tohto vrcholu (tak ako po vymazávaní, lebo môže nastať situácia -2, 0 alebo 2, 0).

Takouto konštrukciou sme zachovali štruktúru, vyváženosť aj balančné faktory nášho AVL stromu. Celkovo táto operácia trvá $\mathcal{O}(\log n)$, ak $n = max(|F_1|, |F_2|)$. Vymazanie L z F_1 trvá $\mathcal{O}(\log n)$. Vyhľadanie vrcholu R trvá $\mathcal{O}(\log n)$, následné zmeny vrcholov stromu trvajú $\mathcal{O}(1)$ a nakoniec vyváženie stromu z otca novovzniknutého vrcholu trvá $\mathcal{O}(\log n)$.

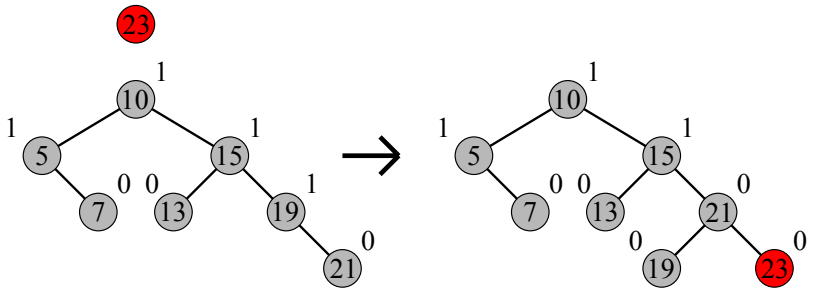
Rozdelenie na dva AVL stromy: Táto operácia má rozdeliť strom F podľa prvku q na dva podstromy F_1 a F_2 také, že všetky prvky z F_1 sú menšie ako q a všetky prvky z F_2 sú väčšie alebo rovné ako q.

Najprv vyhľadáme v F prvok q, prípadne pozíciu, na ktorú by sme prvok q vložili, ak $q \notin F$. Zapamätáme si cestu od koreňa ku pozícii q. Na tejto ceste máme postupne vrcholy, pre ktoré si zapamätáme, či sú väčšie alebo menšie ako q. Operácia je analogická ako postup pre červeno-čierne stromy (používa sa aj pre klasické binárne vyhľadávacie stromy), akurát operujeme na AVL stromoch.

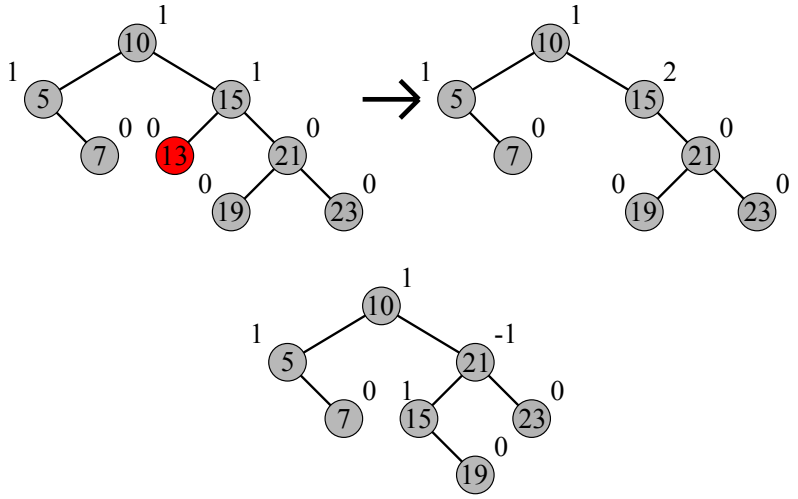
Najprv vytvoríme dve prázdne fronty F_1 a F_2. Ak sme vo vrchole, ktorý je väčší alebo rovný q, tak jeho ľavý podstrom spojíme pomocou operácie spájania front s frontou F_1. Ak sme vo vrchole, ktorý je menší ako q, tak jeho pravý podstrom spojíme pomocou operácie spájania front s frontou F_2. Následne treba prvky, ktoré sú na ceste od koreňa ku prvku q pridať do F_1 alebo F_2. Celkovo to znamená $\mathcal{O}(\log n)$ operácii spojenia a $\mathcal{O}(\log n)$ operácii vloženia. To by znamenalo, že celková zložitosť je $\mathcal{O}(\log^2 n)$. Vďaka tomu, že napájame fronty na F_1 vždy na najmenší prvok a na F_2 vždy na najväčší prvok, možno operácie spojenia ako aj vkladania uzla vykonať v čase $\mathcal{O}(1)$ (a to tak, že na ceste od najmenšieho listu ku koreňu stačí každý prvok vyvážiť rotáciou na strome maximálne raz). Preto v skutočnosti táto operácia bude trvať $\mathcal{O}(\log n)$.

Príklad vloženia uzla: Na obrázku 2.19 vidno vloženie uzla s hodnotou 23 do existujúceho AVL stromu. Najprv sa nájde pozícia, kde sa má uzol vložiť (ako pravý syn uzla 21). Následne sa vloží, jeho balančný faktor sa inicializuje na 0 a smerom ku koreňu upravujeme balančné faktory. Z uzla 23 prejdeme do uzla 21, keďže sme prišli z pravého podstromu, zvyšujeme faktor o 1. Faktor je 1, môžeme pokračovať ďalej smerom hore.

Uzlu 19 opäť zvýšime balančný faktor o 1 a nadobudne hodnotu 2, čo znamená, že musíme uzol vyvážiť. Ak použijeme označenie z popisu



Obr. 2.19. Vloženie uzla do AVL stromu



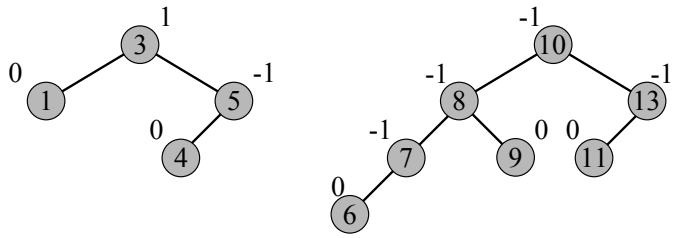
Obr. 2.20. Vymazanie uzla z AVL stromu

pridávania uzla, tak P je 19, R je 21 a L je prázdny uzol. Podľa balančných faktorov v P a R je jasné, že nastal prípad Pravý-Pravý a teda rotáciou vľavo v uzle 19 s uzlom 21 vyvážime strom.

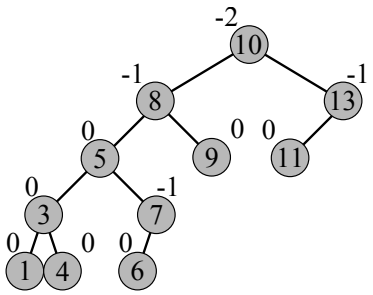
Uzol 19 po rotácii nemá žiadnych synov, faktor nastavíme na 0. 21 má synov 19 a 23 (s faktormi 0), faktor nastavíme na 0. To znamená koniec vyvažovania.

Príklad vymazania uzla: Pokračujme s predchádzajúcim prípadom. Vymažme uzol 13. Keďže bol z ľavého podstromu 15 vymazaný uzol, zvýšime faktor o 1. Faktor je teraz 2, musíme vyvážiť. Nastala špeciálna situácia, ktorá pri vkladaní uzla nastať nemôže (faktor 15 je 2 a faktor 21 je 0). Strom vyvážime ľavou rotáciou v 15 s uzlom 21. Faktory sa upravia podľa rotácie a výsledok ako aj medzikroky sú zobrazené na obrázku 2.20.

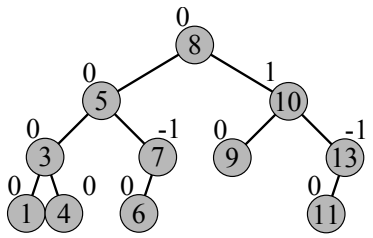
Príklad spojenia: Na obrázku 2.21 sú dva AVL stromy, ktoré chceme spojiť do jedného. Výška F_1 je menšia ako výška F_2, takže postupujeme tak ako sme popísali. Z F_1 odstránime najvyšší prvok 5 (označený L) a vznikne jednoduchý vyvážený strom s koreňom 3, ktorý má dvoch synov 1 a 4. V F_2 zostúpime z koreňa doľava dvakrát, čím



Obr. 2.21. Stromy F_1 a F_2, ktoré chceme spojiť



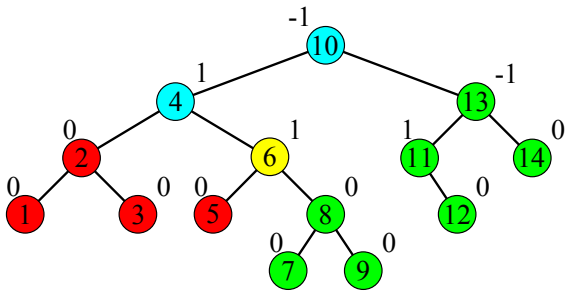
Obr. 2.22. Medzikrok spojenia



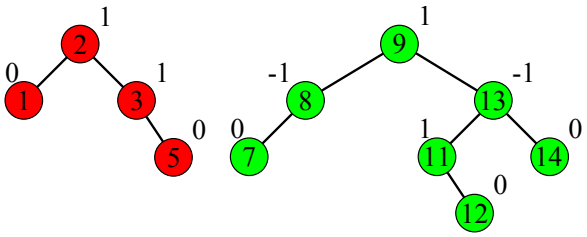
Obr. 2.23. Výsledok spojenia stromov

sa dostaneme na prvok 7 (označený R). Podstrom vo vrchole 7 má rovnakú výšku ako strom F_1 bez prvku 5. Prvok 7 teda nahradíme prvkom 5, pričom jeho ľavý podstrom bude upravený F_1 a jeho pravý podstrom bude podstrom vo vrchole 7. Táto situácia je naznačená na obrázku 2.22. Vďaka konštrukcii vrchola 5 v upravenom strome je jeho balančný faktor 0. Vrcholu 8 zvýšime faktor o 1 a ďalej sa tvárime, ako keby sme práve pridali vrchol 5 do tohto stromu, a preto spustíme vyvažovanie smerom ku koreňu. Vrchol 8 má faktor -1, lebo mal -1, prirátali sme 1 za spojenie stromov a odrátali 1 za to, že sa to dialo v ľavom podstrome 8. Vrcholu 10 sa faktor zníži o 1, teda upravíme strom vo vrchole 10 pravou rotáciou s vrcholom 8. Dostávame sa ku výsledku na obrázku 2.23.

Príklad rozdelenia: Na obrázku 2.24 vidno situáciu na rozdelenie. Ak vezmeme cestu od koreňa ku prvku, podľa ktorého rozdeľujeme strom (žltý prvok 6), tak napravo od tejto cesty sú podstromy (červené), ktoré sa spoja do F_2, a naľavo (zelené), ktoré sa spoja do F_1.



Obr. 2.24. AVL strom, ktorý má byť rozdelený na dva podľa prvku 6



Obr. 2.25. Medzikrok rozdelenia AVL stromu z obrázku 2.24

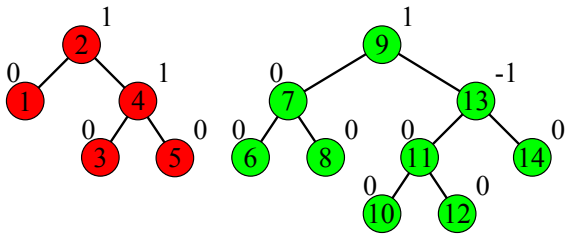
Taktiež sú to prvky ľavého a pravého podstromu prvku, podľa ktorého strom rozdeľujeme.

Algoritmus bude postupovať od koreňa ku prvku 6 a tvoriť stromy F_1 a F_2. Začíname v prvku 10. Je väčší ako 6, preto jeho pravý podstrom s koreňom 13 pridáme do zatiaľ prázdneho F_2. Vrchol 10 si pridáme do zoznamu vrcholov, ktoré na konci algoritmu pridáme do F_2 jednoduchým vkladaním. Zostúpime do vrchola 4 a vykonáme symetrickú operáciu, ako vo vrchole 10. Ľavý podstrom vrchola 4 pridáme do zatiaľ prázdneho F_1 a prvok 4 si uložíme do zoznamu prvkov, ktoré treba na konci pridať do F_1.

Pokračujeme vo vrchole 6. Našli sme vrchol, podľa ktorého rozdeľujeme strom. Vezmeme jeho pravý (resp. ľavý) podstrom a operáciou spojenia ho pripojíme do F_2 (resp. F_1). Výsledok týchto operácii je zobrazený na obrázku 2.25. Vo výsledku patrí vrchol 6 do stromu F_2, teda ho pridáme do zoznamu na pridanie (v tomto zozname máme aj prvok 10).

Prvá časť algoritmu prebehla, ostáva nám pridať prvok 4 do F_1 a prvky 10 a 6 do F_2. Výsledok je zobrazený na obrázku 2.26.

□



Obr. 2.26. Výsledok rozdelenia AVL stromu z obrázku 2.24

KAPITOLA 3

Konvexné obaly

CVIČENIE 3.1 (30 bodov). *Je daná množina obsahujúca n bodov v rovine. Zostrojte jednoduchý mnohouholník, ktorého vrcholy sú body danej množiny. Ukážte, že* $\Omega(n \log n)$ *je dolná hranica časovej zložitosti.*

Riešenie (Martin Škorupa, 31.10.2008): Majme pole všetkých bodov **A**, výsledné pole poradia bodov jednoduchého mnohouholníka **B** a pole pre konvexný obal **KO**. Jednoduchý mnohouholník hľadáme nasledovne:

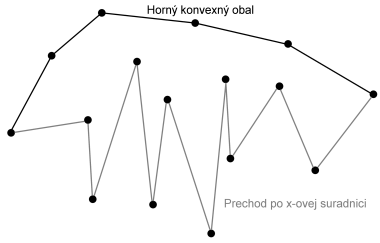
(1) Usporiadame **A** podla x-ovej súradnice, v prípade rovnosti x-ovej podľa y-ovej súradnice. $\mathcal{O}(n \log n)$
(2) Nájdeme konvexný obal bodov. (predpokladáme že začína v najľavejšom bode a ide v smere chodu hodinových ručičiek) $\mathcal{O}(n \log n)$
(3) Prehľadávame **A** a hľadáme body patriace do **KO** (v tom istom poradí ako su v **KO**), ak nejaký nájdeme, vyberieme ho z **A** a vložíme ho do **B**. V **B** teda máme hornú časť konvexného obalu od najľavejšieho po najpravejší bod a v **A** všetky ostatné body. $\mathcal{O}(n)$
(4) Ideme zozadu **A** (teda od najväčšieho x k najmenšiemu). Zaradom vyberáme body z **A** a vkladáme ich do **B**. $\mathcal{O}(n)$
(5) V **B** máme jednoduchý mnohouholník v čase $\mathcal{O}(n \log n)$.

Dôkaz:

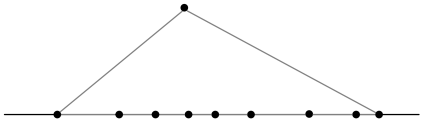
- To že bolo pole **A** usporiadané podľa x a potom podľa y nám zaručuje, že v kroku 4. sa ani jedna hrana nepretne, lebo buď x-ová súradnica klesá alebo ak zostáva rovnaká tak klesá y-ová súradnica. Zároveň sa ale ani nepretne s hornou časťou konvexného obalu. Teda **B** je jednoduchý mnohouholník. (poz. obr. 3.1
- Majme množinu $n-1$ bodov ležiacich na x-ovej osi. Posledný bod môže byť ľubovoľne inde ako na x-ovej osi. Na nájdenie jednoduchého mnohouholníka pre takúto množinu potrebujeme usporiadať body na x-ovej osi. Teda časová zložitosť nájdenia jednoduchého mnohouholníka je $\mathcal{O}(n \log n)$ (poz. obr. 3.2).

□

Riešenie (Tomáš Gieci, 3.11.2008):
a) Zostrojte jednoduchý mnohouholník



OBR. 3.1. Nájdenie horného konvexného obalu a spätný prechod po x-ovej súradnici



OBR. 3.2. Nájdenie usporiadania bodov na priamke

Množinu bodov si označíme $P = \{p_1, \ldots, p_n\}$ (obr. 3.3, podobr. 1)

Vyberieme náhodne jeden bod z množiny P a označíme ho x, okrem bodu p_1.

(1) Utriedime body množiny P podľa polárnych súradnic voči priamke určenej bodmi x a p_1. Novú usporiadanú množinu si označíme $V = \{v_1, \ldots, v_{n-1}\}$ (obr. 3.3, podobr. 2)
(2) Bod v_i spojíme s bodom v_{i+1} (obr. 3.3, podobr. 3)
(3) Opakujeme krok 2, kým neprídeme na koniec usporiadanej množiny P.
(4) Vytvoríme hrany $\overline{xv_1}, \overline{v_n x}$
(5) Zoznam hrán nám vytvára jednoduchý mnohouholník (obr. 3.3, podobr. 4)

Časová zložitosť utriedenia neusporiadanej množiny bodov (bod 1.) je $(n-1) * (\log(n-1))$ a teda dolná časová zložitosť tohto kroku je $\Omega(n \log n)$.

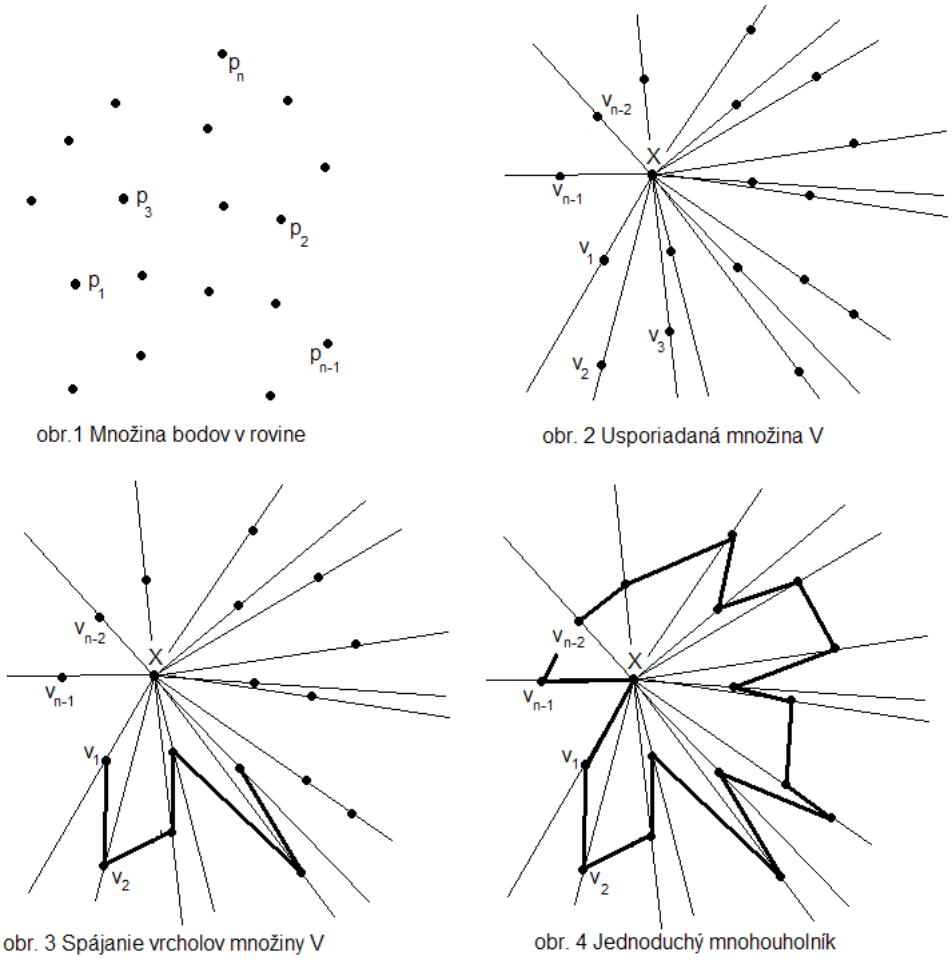
Časová zložitosť bodu 2. je $O(1)$, bodu 3. je $O(\mathrm{n})$ a bodu 4. je $O(1)$.

Celková dolná časová zložitosť algoritmu je $\Omega(n \log n)$.

b) Ukážte, že $\Omega(n \log n)$ je dolná hranica časovej zložitosti

Predpokladajme, že časová zložitosť zostrojenia jednoduchého mnohouholníka nie je $\Omega(n \log n)$. Hľadanú časovú zložitosť označme $\Omega(x)$.

- Nemôžeme predpokladať, že $\Omega(x) > \Omega(n \log n)$, pretože sme v algoritme uvedenom v bode a) ukázali že existuje menšia časová zložitosť.
- Predpokladajme, že $\Omega(x) < \Omega(n \log n)$. Časová zložitosť vytvorenia konvexného obalu z jednoduchého mnohouholníka je



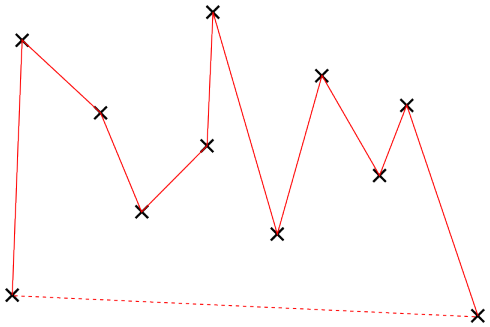
OBR. 3.3. Obrázky k riešeniu k cvičeniu 3.1.

$\Omega(n)$. Z toho vyplýva, že časová zložitosť na vytvorenie konvexného obalu z ľubovoľnej množiny bodov je menšia ako $\Omega(n \log n)$. Ale vieme, že časová zložitosť vytvorenie konvexného obalu je $\Omega(n \log n)$. Dostávame spor s naším predpokladom, čiže $\Omega(x) = \Omega(n \log n)$.

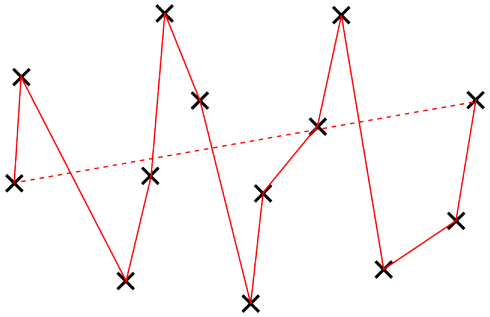
□

Riešenie (Jakub Mišún, 18.11.2011): Jedným zo spôsobov ako zostrojiť jednoduchú krivky z n bodov je usporiadať body podľa x-ovej súradnice (respektíve podľa y-ovej súradnice) a pospájať ich v tomto poradí. Keď však chceme túto krivku uzavrieť na jednoduchý mnohouholník spojením prvého a posledného bodu, nie vždy je to možné. Takýto postup môžeme zvoliť, len v špeciálnych prípadoch keď prvý

a posledný bod sú body z dvoma najmenšími alebo najväčšími y-ovými (respektíve x-ovými) súradnicami. Tento prípad vidíme na obrázkoch 3.4, 3.5.



Obr. 3.4. Konštrukcia je v tomto prípade veľmi jednoduchá



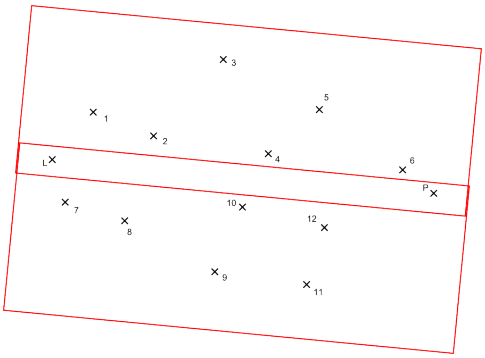
Obr. 3.5. Vo väčšine prípadov uzavretie nie je tak jednoduché

V ostatných prípadoch je potrebné algoritmus mierne upraviť. Potrebujeme si zaistiť neprerušované uzavretie krivky. To zaistíme nasledovne:

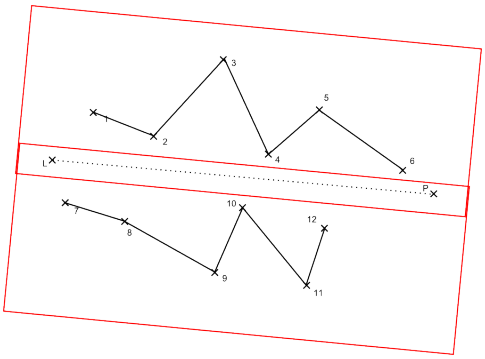
Nájdeme si bod množiny s najmenšou a najväčšou súradnicou x. Označme ich L a P. Tieto určujú priamku, ktorá delí všetky zvyšné body do dvoch skupín. Body "nad"priamkou usporiadame podľa súradnice x a pospájame ich v tomto poradí. Dostaneme jednoduchú krivku. Rovnako utriedime a pospájame i body "pod"priamkou. Obe krivky sú jednoduché a určite sa nepretínajú, pretože ich oddeľuje priamka LP. Tieto krivky sa už jednoducho spoja a to tak, že začiatočné body oboch kriviek spojíme s bodom L a koncové body s bodom P. Dostávame uzavretú krivku.

ZLOŽITOSŤ:

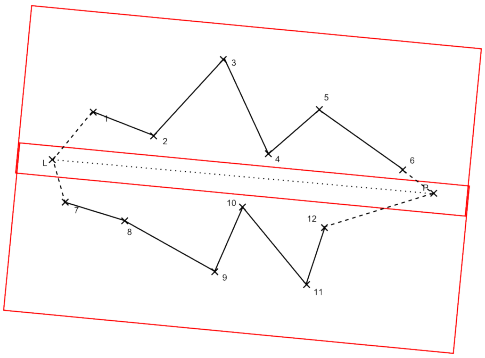
(1) Usporiadanie n bodov podľa súradnice. Ak sa niekoľko bodov zhoduje v súradnici, tak tieto utriedime podľa druhej súradnice ($O(n \log n)$)
(2) Nájdenie bodov L a P ($O(2)$)



OBR. 3.6. Rozdelenie bodov



OBR. 3.7. Dve oddelené jednoduché krivky



OBR. 3.8. Výsledný jednoduchý mnohouholník

(3) Rozdelenie bodov do dvoch skupín ($O(n)$)
(4) Body v oboch skupinách už sú sporiadané.
(5) Pospájanie bodov oboch skupín podľa poradia. ($O(n)$)

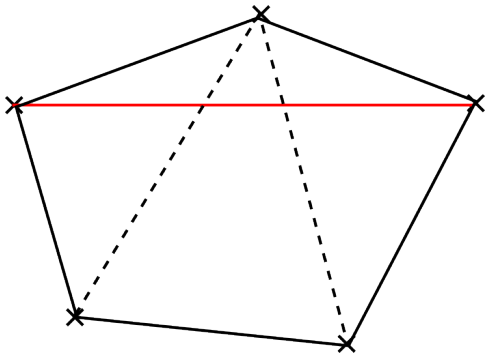
Teda celková časová náročnosť algoritmu je $O(n \log n)$.

ALGORITMUS:

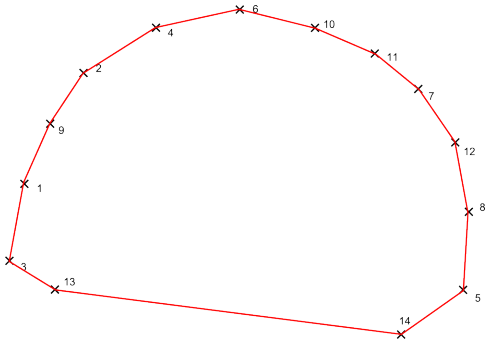
```
BEGIN()

int P[n];
XSort(P);

L=P[0];
R=P[n];

for (int i = 0 ; i ≤ n ; i++)
    if ( LR.P[i] ≥ 0)
       P[i] → UpList;
    else
       P[i] → DownList;

nu=UpListLenght-1;
nd=DownListLenght-1;

MoveTo(UpList[0].x,UpList[0].y);

for (int i=1; i < nu ;i++)
    LineTo(UpList[i].x,UpList[i].y);

MoveTo(UpList[0].x,UpList[0].y);

for (int i = 1 ; i < nd ; i++)
    LineTo(DownList[i].x,DownList[i].y);

LineTo(UpList[nu].x,UpList[nu].y);

END()
```

Teraz ukážeme dolnú hranicu algoritmu. Vezmime si množinu n bodov, pre ktorú platí, že jej kovexný obal obsahuje všetkých n bodov. Pre takúto množiny takisto platí, že je jednoduchý mnohouholník s danými vrcholmi je zároveň aj jej konvexným obalom, pretože ak by sme spojili usečkou iné ako susedné body obalu, rozdelíme zvyšné body do dvoch polrovín, pričom body jednej nespojíme s bodmi z druhej bez preťatia deliacej úsečky práve kvôli konvexnosti.

Pre takéto množiny teda platí, že zostrojenie jednoduchého mnohouholníka je aj zostrojením konvexného obalu. Avšak dolná hranica zložitosti nájdenia konvexného obalu je $\Omega(n \log n)$ a teda i dolná hranica zložitosti nájdenia jednoduchého mnohouholníka je $\Omega(n \log n)$, v opačnom prípade by nastal spor. □



OBR. 3.9. Konvexný obal je jediný jednoduchý



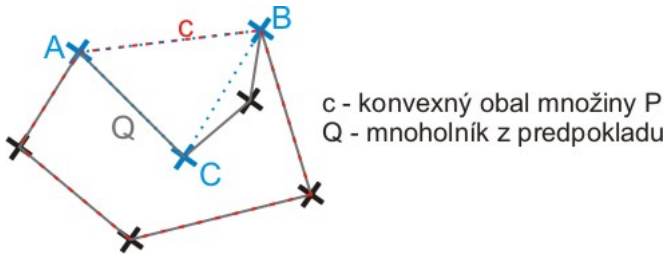
OBR. 3.10. Jednoduchý mnohouholník a konvexný obal sú totožné

CVIČENIE 3.2 (20 bodov). *Dokážte, že mnohouholník Q s najmenším obvodom obsahujúcim neprázdnu konečnú množinu bodov P je konvexný. Ukážte, že každá konvexná množina obsahujúca P obsahuje aj mnohouholník Q.*

Riešenie (Marta Režnáková, 26.10.2008): Predpokladajme, že existuje taký mnohouholník Q, ktorého obvod je najmenší a nie je to konvexný obal množiny P. Že nie je konvexným obalom znamená, že existuje taká úsečka $\overline{AB}$, ktorá spája vrcholy mnohouholníka Q, no nepatrí doňho, a teda ani nie je jeho stenou. Existuje teda aspoň jeden bod, označme si ho C, ktorý je vrcholom mnohouholníka Q a leží na ceste medzi bodmi A a B. Z predpokladu, že mnohouholník Q má najmenší obvod, možno usudzovať nerovnosť $|\overline{AB}| > |\overline{AC}| + |\overline{CB}|$, čo je však v rozpore s trojuholníkovou nerovnosťou. Analogicky dokazujeme aj pri ostatných bodoch, ako je bod C.

Predpokladajme, že existuje konvexná množina obsahujúca P a zároveň neobsahujúca Q. Keďže obsahuje P, obsahuje aj všetky vrcholy Q a nakoľko ide o konvexnú množinu, obsahuje aj všetky úsečky medzi nimi, čiže aj všetky strany Q, čo je v rozpore s predpokladom, že daná konvexná množia neobsahuje Q.

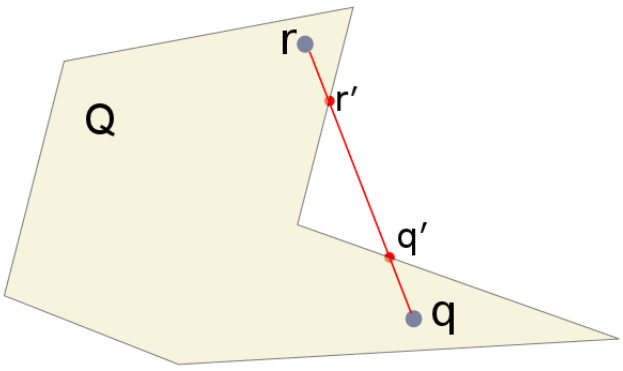
□



Obr. 3.11. Konvexný obal má najmenší obvod zo všetkých mnohouholníkov obsahujúcich danú množinu vrcholov.

Riešenie (Gabriel Foltán, 26.10.2011): Sporom. Myšlienka dôkazu spočíva na predpoklade, že mnohouholník Q má najmenší obvod a nie je konvexný. Kedže nie je konvexný, musí existovať nejaká dvojica bodov $\mathbf{q}$,$\mathbf{r}$, taká, že $\mathbf{q}$,$\mathbf{r} \in Q$, ale úsečka $\overline{\mathbf{qr}}$ nepatrí Q. Zo všetkých dvojíc bodov úsečky $\overline{\mathbf{qr}}$, zoberme také $\mathbf{q}'$,$\mathbf{r}' \in Q$, ktoré sú k sebe najbližšie a zvyšok úsečky $\overline{\mathbf{q}'\mathbf{r}'}$ nepatrí Q.

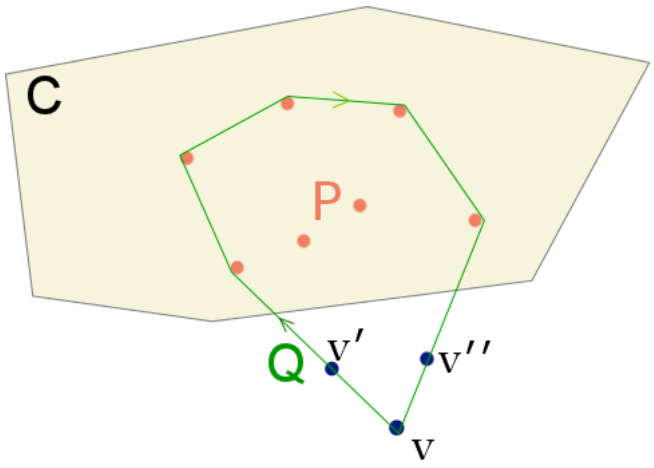
Zostrojme teraz mnohouholník Q' tak, že v pôvodnom mnohouholníku odstránime postupnosť vrcholov medzi $\mathbf{q}'$ a $\mathbf{r}'$ a pridáme namiesto toho stranu $(\mathbf{q}'\mathbf{r}')$. Q' obsahuje neprázdnu konečnú množinu bodov P, pretože obsahuje aj Q a Q obsahuje P. Q' má menší obvod ako Q pretože z trojuholníkovej nerovnosti vyplýva, že dĺžka strany $(\mathbf{q}'\mathbf{r}')$ je menšia ako dĺžka odstránenej postupnosti strán. To je spor s predpokladom, že Q má najmenší obvod. Preto nekonvexný mnohouholník nemôže byť mnohouholník s najmenším obvodom, ktorý obsahuje neprázdnu konečnú množinu bodov.



Obr. 3.12. Mnohouholník Q a vybratá dvojica $\mathbf{q}$,$\mathbf{r}$, ktorá skráti obvod mnohouholníka.

Teraz ukážeme, že každá konvexná množina obsahujúca P obsahuje aj mnohouholník Q

Sporom. Nech množina C je konvexná množina obsahujúca P. Predpokladajme, že Q je mnohouholník s najmenším obvodom obsahujúcim P, ale taký, že C neobsahuje Q. Najprv ukážeme, že existuje vrchol z Q, ktorý nie je v C. Ak by každý vrchol z Q bol v C, potom kedže C je konvexná množina, tak každá strana spájajúca dva vrcholy musí byť tiež v C. Preto, ak C neobsahuje P, potom existuje vrchol z Q ale nepatriaci C. Označme si ten vrchol $\mathbf{v}$. Vieme, že $\mathbf{v}$ nepatrí P, pretože C obsahuje P. Definujme si vrchol $\mathbf{v}'$ ako novú pozíciu vrchola $\mathbf{v}$ po posunutí v smere hodinových ručičiek o nejaké ϵ po strane orientovaného mnohouholníka, ktorá východza z $\mathbf{v}$. Analogicky si definujme vrchol $\mathbf{v}''$ posunutí v protismere hodinových ručičiek o ϵ po strane orientovaného mnohouholníka, ktorá vchádza do $\mathbf{v}$ (situácia znázornená na obrázku 3.13.). Pretože $\mathbf{v}$ nie je v P, a ak ϵ bolo dostatočne malé, potom trojuholník $\mathbf{v}\mathbf{v}'\mathbf{v}''$ neobsahuje žiadne body z P. Z trojuholíkovej nerovnosti, dĺžka strany $\mathbf{v}'\mathbf{v}''$ je menej ako suma dĺžok strán $\mathbf{v}\mathbf{v}'$ a $\mathbf{v}\mathbf{v}''$. Preto môžeme zmenšiť obvod mnohouholníka Q nahradením vrchola $\mathbf{v}$ vrcholmi $\mathbf{v}'$ a $\mathbf{v}''$ a stále budeme mať mnohouholník obsahujúci P. To je spor s predpokladom, že mnohouholník Q je mnohouholník s najmenším obvodom obsahujúci P. Preto mnohouholník Q s najmenším obvodom patrí ľubovolnej konvexnej množine obsahujúcej P.



Obr. 3.13. Skrátenie obvodu mnohouholníka „urezaním“ dostatočne malého trojuholníka s vrcholom $\mathbf{v}$.

□

Cvičenie 3.3 (30 bodov). *Ukážte, že dva disjunktné konvexné mnohouholníky majú práve štyri spoločné oporné priamky.*

Riešenie (Martin Škorupa, 10.11.2008):

Majme dva disjunktné konvexné mnohouholníky **A** a **B**. Musí existovať priamka **p**, ktorá rozdeľuje rovinu na dve polroviny, pričom **A** a **B** ležia v opačných polrovinách a s **p** nemajú spoločný prienik. Transformujme rovinu tak, aby **p** bola totožná s **x**-ovou osou a **A** ležalo v I. kvadrante a **B** ležalo v IV. kvadrante. Pre všetky **X** z **x**-ovej osi existujú práve dve priamky $\mathbf{a_1}$, $\mathbf{a_2}$ také, že $\mathbf{a_1}$, $\mathbf{a_2}$ sú oporné pre **A** a práve dve priamky $\mathbf{b_1}$, $\mathbf{b_2}$ také, že $\mathbf{b_1}$, $\mathbf{b_2}$ sú oporné pre **B** (poz. obr. 3.14). Nech pre každé **X** z **x**-ovej osi sú $\mathbf{a_1}$, $\mathbf{a_2}$, $\mathbf{b_1}$, $\mathbf{b_2}$ vždy rovnako usporiadané ($\mathbf{a_2}$, $\mathbf{a_1}$, **x**, $\mathbf{b_1}$, $\mathbf{b_2}$). Majme funkcie:
$\mathbf{f}_\alpha(x)$: $\mathbb{R} \to (0, \pi)$, ktorá pre bod X[x,0] **x**-ovej osi vráti $|\measuredangle_{or}\mathbf{xa_1}|$.
$\mathbf{f}_\beta(x)$: $\mathbb{R} \to (0, \pi)$, ktorá pre bod X[x,0] **x**-ovej osi vráti $|\measuredangle_{or}\mathbf{xa_2}|$.
$\mathbf{f}_\gamma(x)$: $\mathbb{R} \to (0, \pi)$, ktorá pre bod X[x,0] **x**-ovej osi vráti $|\measuredangle_{or}\mathbf{b_1x}|$.
$\mathbf{f}_\delta(x)$: $\mathbb{R} \to (0, \pi)$, ktorá pre bod X[x,0] **x**-ovej osi vráti $|\measuredangle_{or}\mathbf{b_2x}|$.

Funkcie $\mathbf{f}_\alpha(x)$, $\mathbf{f}_\beta(x)$, $\mathbf{f}_\gamma(x)$, $\mathbf{f}_\delta(x)$ sú rastúce a spojité.

Keďže **p** oddeľuje **A** a **B**, tak všetky spoločné oporné priamky musia pretínať **p**. Bodom $\mathbf{X} \in \mathbf{p}$ prechádza spoločná oporná priamka práve vtedy, ak $\mathbf{a_1} = \mathbf{b_1}$, alebo $\mathbf{a_1} = \mathbf{b_2}$, alebo $\mathbf{a_2} = \mathbf{b_1}$, alebo $\mathbf{a_2} = \mathbf{b_2}$. Teda práve vtedy, keď $\mathbf{f}_\alpha(x) + \mathbf{f}_\gamma(x) = \pi$, alebo $\mathbf{f}_\alpha(x) + \mathbf{f}_\delta(x) = \pi$, alebo $\mathbf{f}_\beta(x) + \mathbf{f}_\gamma(x) = \pi$, alebo $\mathbf{f}_\beta(x) + \mathbf{f}_\delta(x) = \pi$. Na základe spojitosti, rastúcosti a definičných oborov každý tento prípad nastáva práve raz. Teda dva disjunktné konvexné mnohouholníky majú práve štyri spoločné oporné priamky.

□

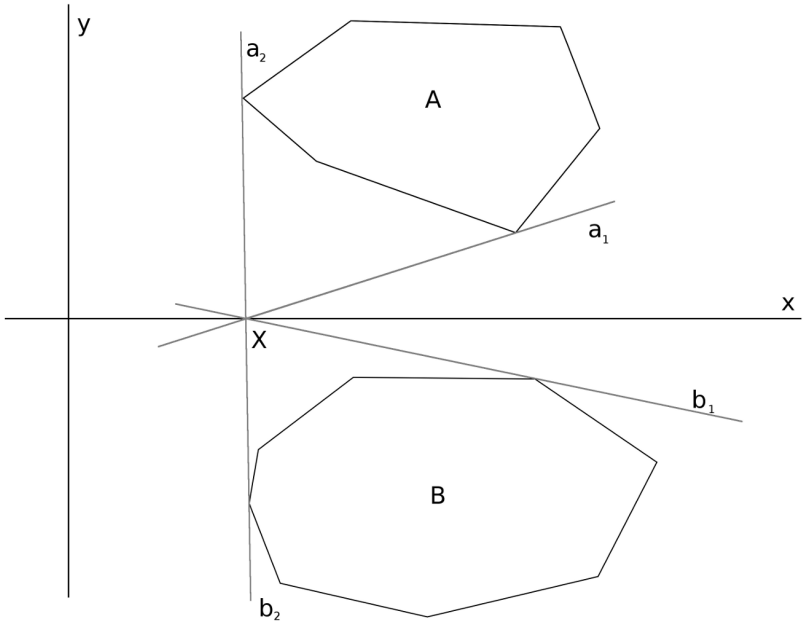
Riešenie (Viktória Bakurová, 23.11.2009): Majme 2 disjunktné konvexné mnohouholníky P_1 a P_2. Keďže sú disjunktné, existuje priamka, ktorá tieto mnohouholníky oddeľuje. Urobme transformáciu, ktorej výsledkom bude, že oddeľujúca priamka bude totožná s osou ya mnohouholníky budú ležať v kladnej polrovine určenej osou x.Hľadajme ich spoločné oporné priamky.

(1) V mnohouholníku P_1 nájdeme bod s maximálnou y-ovou a minimálnou y-ovou súradnicou, ak je ich viac, vyberieme taký, ktorý má maximálnu x-ovú súradnicu. Označíme si ich P_{1max} a P_{1min}.

V mnohouholníku P_2 tiež nájdeme bod s maximálnou y-ovou a minimálnou y-ovou súradnicou, ak je ich viac, vyberieme taký, ktorý má maximálnu x-ovú súradnicu. Označíme si ich P_{2max} a P_{2min}.

Porovnáme P_{1max} a P_{2max} a vyberieme maximum, označme ho p_{max}. Tak isto porovnáme P_{1min} a P_{2min} a vyberieme minimum, označme ho p_{min}.

Ak zostrojíme priamky rovnobežné s osou x cez nájdené minimum p_{min} a maximum p_{max}, mnohouholníky P_1 a P_2 budú ležať medzi nimi.



OBR. 3.14. Oporné priamky k mnohuholníkom A a B zostrojené z bodu X na oddeľujúcej priamke (os x).

Pri hľadaní maxima a minima môžu nastať 4 možnosti:

(a) maximum aj minimum je z P_1
(b) maximum aj minimum je z P_2
(c) maximum je z P_1 a minimum je z P_2
(d) maximum je z P_2 a minimum je z P_1

(2) Nájdeme 2 spoločné oporné priamky mnohouholníkov. Nech nastala možnosť, že maximum aj minimum je z P_1.

Priamku p_{max} otáčame v smere chodu hodinových ručičiek okolo bodu P_{1max} kým nenarazíme buď na hranu mnohouholníka P_1 alebo na vrchol mnohouholníka P_2 a hľadáme spoločnú opornú priamku P_1 a P_2:

- ak narazíme na vrchol P_2, máme hľadanú opornú priamku
- ak narazíme na hranu P_1, otáčame okolo druhého vrcholu ležiaceho na tejto hrane, hľadáme opornú priamku ďalej rovnakým algoritmom

Takúto opornú priamku isto nájdeme, lebo P_2 leží medzi priamkami p_{min} a p_{max}.

Priamku p_{min} otáčame v smere proti chodu hodinových ručičiek okolo bodu P_{1min} kým nenarazíme buď na hranu mnohouholníka P_1 alebo na vrchol mnohouholníka P_2 a hľadáme spoločnú opornú priamku P_1 a P_2:

- ak narazíme na vrchol P_2, máme hľadanú opornú priamku

- ak narazíme na hranu P_1, otáčame okolo druhého vrcholu ležiaceho na tejto hrane, hľadáme opornú priamku ďalej rovnakým algoritmom

Takúto opornú priamku nájdeme z rovnakého dôvodu, ako predtým.

Rovnako by sme našli 2 spoločné oporné priamky aj pre ostatné prípady, rozdiel je len v smere otačania priamok p_{min} a p_{max} :

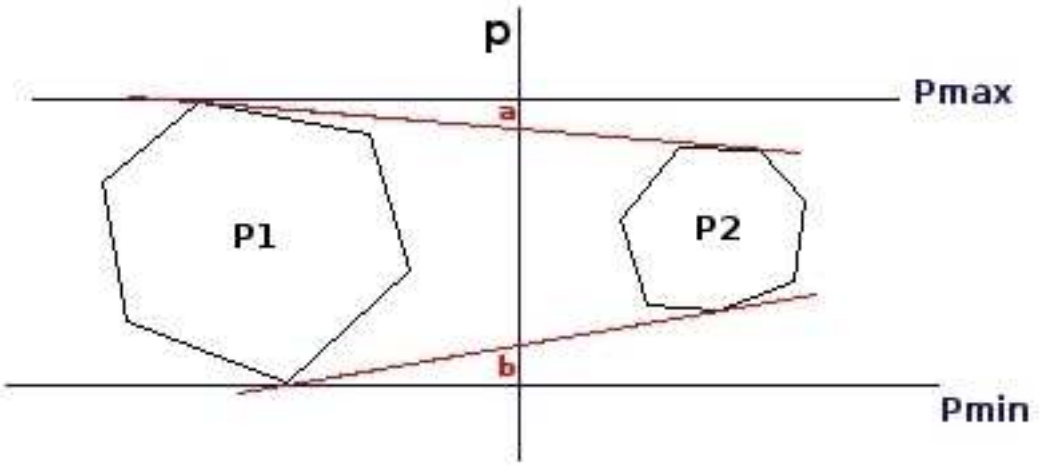
- ak maximum je z P_2, priamku p_{max} otáčame v smere proti chodu hodinových ručičiek
- ak minimum je z P_2, priamku p_{min} otáčame v smere chodu hodinových ručičiek

Takýmto spôsobom nájdeme 2 oporné priamky mnohouholníkov P_1 a P_2.

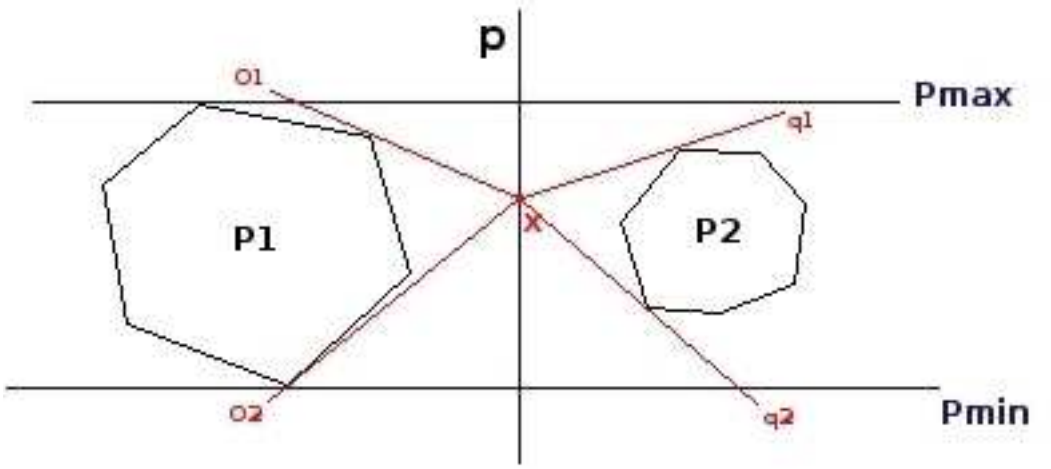
Treba si uvedomiť, že oporné priamky skonštruované takýmto spôsobom nemôžu byť totožné. Keby boli totožné, znamenalo by to, že mnohouholníky sú degenerované. (Krátke odôvodnenie: Priamka p_{min} rozdeľuje rovinu na dve polroviny. V jednej z týchto polrovín ("nad" p_{min}) ležia mnohouholníky P_1aj P_2. Označme si túto polrovinu p_{min+}. Rovnako priamka p_{max} rozdeľuje rovinu na dve polroviny. V jednej z týchto polrovín ("pod" p_{max})ležia mnohouholníky P_1aj P_2. Označme si túto polrovinu p_{max-}. Otáčajme priamku p_{min} a s ňou aj polrovinu p_{min+} a nájdime prvú spoločnú opornú priamku mnohouholníkov P_1 a P_2. Z konštrukcie tejto opornej priamky je jasné, že mnohouholníky sa stale budú nachádzať v polrovine označenej p_{min+}. Podobne aj otáčaním p_{max} a polroviny p_{max-}nájdime spoločnú opornú priamku mnohouholníkov. Je zrejmé, že aj v tomto prípade sa P_1 a P_2 budú nachádzať v polrovine označenej p_{max-}. Teraz si už stačí uvedomiť len to, že ak by boli nájdené oporné priamky totožné, tak mnohouholníky P_1 a P_2 by museli ležať súčasne aj v jednej aj v druhej polrovine vzhľadom na túto opornú priamku, takže jediná možnosť je, že ležia na spoločnej hranici, čiže na priamke. Ide teda o degenerované mnohouholníky.)

Z konštrukcie týchto oporných priamok je jasné, že ďalšie oporné priamky budú prechadzať len vrcholmi ležiacimi medzi vrcholmi, cez ktoré prechádzajú prvé 2 nájdené oporné priamky. Označme si prienik týchto oporných priamok s priamkou oddeľujúcou mnohouholníky a a b, pričom a bude mať väčšiu y-ovú súradnicu ako b. Potom ďalšie spoločné oporné priamky mnohouholníkov pretnú priamku p v bodoch, ktoré ležia na úsečke ab

(3) Majme mnohouholníky P_1 a P_2 oddelené priamkou p. Na priamke si zvoľme ľubovoľný bod X. Z tohoto bodu vieme zostrojiť



Obr. 3.15. Oporné priamky disjunktných mnohouholníkov P_1 a P_2, ktoré vznikli otáčaním priamok p_{min} a p_{max}



Obr. 3.16. Oporné priamky k mnohouholníkom P_1 a P_2 v ľubovoľnom bode ležiacom na oddeľujúcej priamke

práve dve oporné priamky k mnohouholníku P_1, označme si ich o_1 a o_2. Rovnako z tohoto bodu vieme zostrojiť práve dve oporné priamky k mnohouholníku P_2, označme si ich q_1 a q_2.

Skúmajme uhol, ktorý zviera o_1 s p s vrcholom v bode X, pričom druhé rameno tohoto uhla smeruje do ∞ v smere osi y:

- ak ideme s bodom X do $-\infty$, uhol sa zmenšuje
- ak ideme s bodom X do ∞, uhol sa zväčšuje

Máme teda spojitú funckiu f, ktorá ľubovoľnému bodu Xz osi $y$priradí práve jeden uhol z intervalu $[0, \pi]$, teda $f : \mathbb{R} \to [0, \pi]$. Táto funkcia je rastúca.

Skúmajme uhol, ktorý zviera q_2 s p s vrcholom v bode X, pričom druhé rameno tohoto uhla smeruje do $-\infty$ v smere osi y:

- ak ideme s bodom X do $-\infty$, uhol sa zväčšuje
- ak ideme s bodom X do ∞, uhol sa zmenšuje

Máme teda spojitú funckiu g, ktorá ľubovoľnému bodu Xz osi $y$priradí práve jeden uhol z intervalu $[0, \pi]$, teda $g : \mathbb{R} \to [0, \pi]$. Funkcia g je klesajúca.

Z uvedených vlastností funkcií fa $g$vyplýva, že existuje práve jeden bod prieniku týchto dvoch funkcií, čiže existuje práve jeden taký bod X na priamke p, že tieto uhly sa budú rovnať. To znamená, že priamky o_1 a q_2 splývajú a sú spoločnou opornou priamkou P_1 a P_2.

Všimnime si ešte, že v tomto prípade nemusíme prechádzať všetky body X na p, stačí prehľadať interval (a, b), pretože úsečky spájajúce vrcholy P_1 a P_2, cez ktoré bude prechádzať oporná priamka, leží medzi 2 už nájdenými opornými priamkami. Táto úsečka pretne priamku p v bode $X \in (a, b)$.

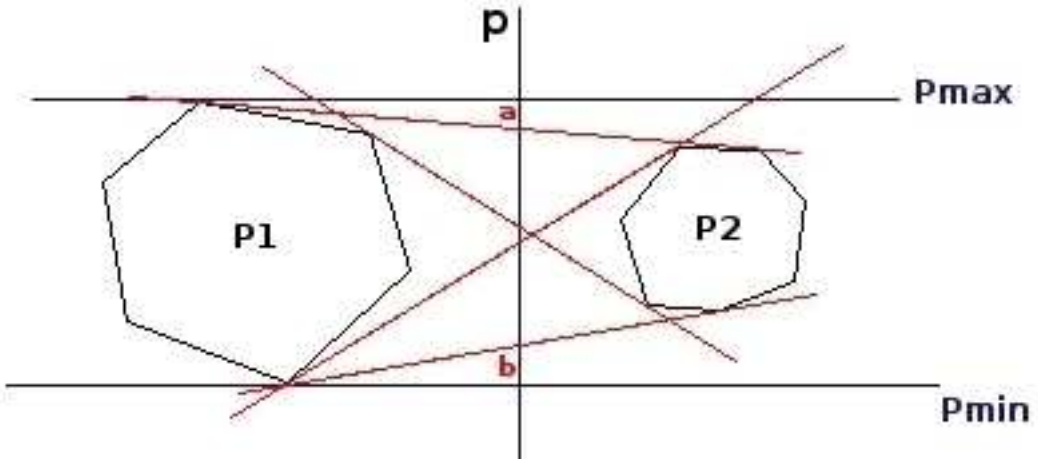
Analogicky si všimnime uhol, ktorý zviera o_2 s p s vrcholom v bode X, pričom druhé rameno tohoto uhla smeruje do $-\infty$v smere osi ya uhol, ktorý zviera q_1 s p s vrcholom v bode X, pričom druhé rameno tohoto uhla smeruje do ∞ v smere osi y.Tieto uhly sa správajú rovnako ako tie v predchadzajúcom prípade, preto opäť vieme zostrojiť dve spojité funkcie, ktoré ľubovoľnému bodu Xz osi $y$priradí práve jeden uhol z intervalu $[0, \pi]$, pričom jedna y týchto funkcií bude klesajúca a druhá rastúca. Preto existuje práve jeden taký bod X na priamke p, že sa uhly budú rovnať. To znamená, že priamky o_2 a q_1 splývajú a sú spoločnou opornou priamkou P_1 a P_2. Znovu stačí aj v tomto prípade prehľadať len body z intervalu (a, b).

Kedže vieme, že všetky ďalšie spoločné oporné priamky daných mnohouholníkov pretnú priamku p v nejakom bode z intervalu (a, b) a vyššie uvedené funkcie pretnú túto priamku práve v jednom bode, je zrejmé, že ďalšie spoločné oporné priamky neexistujú.

Našli sme 4 spoločné oporné priamky k mnohouholníkom P_1 a P_2.

□

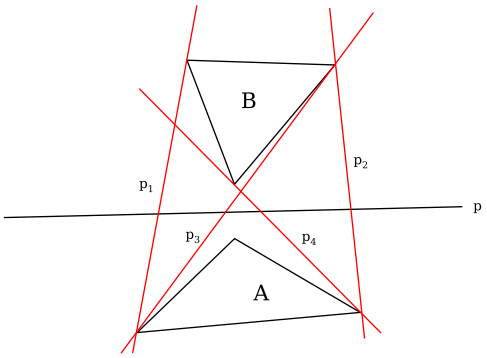
Riešenie (Martin Manduch, 22.11.2009): Majme dva disjunktné konvexné mnohouholníky A, B. Každá spoločná oporná priamka prechádza nejakým vrcholom A a nejakým vrcholom B. Spoločnú opornú



OBR. 3.17. Spoločné oporné priamky k dvom disjunktným mnohouholníkom

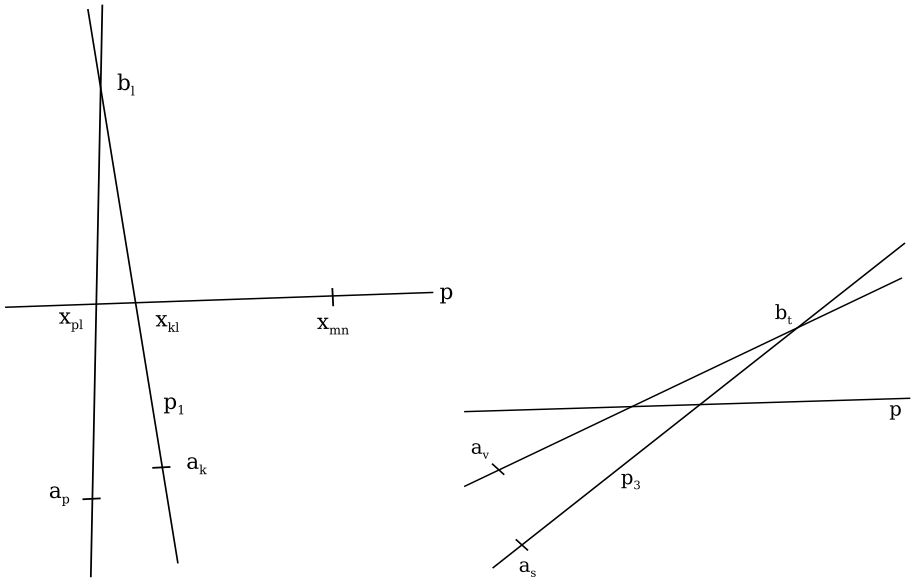
priamku **ab** ,kde **a** je vrchol A a **b** je vrchol B orientujme podľa vektora $\mathbf{b} - \mathbf{a}$. Môžu nastať štyri vzájomné polohy takto orientovaných spoločných priamok a konvexných mnohouholníkov A, B (obr. 3.18):

(1) A, B ležia v pravej polrovine od opornej priamky
(2) A, B ležia v ľavej polrovine
(3) A leží v pravej, B v ľavej polrovine
(4) A leží v ľavej, B v pravej polrovine



OBR. 3.18. Spoločné oporné priamky.

Priamky v týchto polohách označme postupne p_1 až p_4. Keďže A, B sú disjunktné, existuje priamka p, ktorá ich oddeľuje. Nech $\mathbf{a}_1 \dots \mathbf{a}_n$ sú vrcholy$A$ a $\mathbf{b}_1 \dots \mathbf{b}_m$ sú vrcholy$B$. Označme $\mathbf{x}_{ij} = \mathbf{a}_i\mathbf{b}_j \cap p$. Priamku p orientujme tak, že A je od nej napravo. Nech k, l, m, n sú také, že orientovaná úsečka $\mathbf{x}_{kl}\mathbf{x}_{mn}$ má najväčšiu dĺžku a je rovnako orientovaná ako priamka p. Tvrdíme, že priamka p_1 je určená vrcholmi $\mathbf{a}_k, \mathbf{b}_l$ a p_2 je určená vrcholmi $\mathbf{a}_m, \mathbf{b}_n$ Dokážeme to sporom pre p_1. Pre p_2 by sme postupovali analogicky. Nech existuje vrchol z A, ktorý je naľavo od



OBR. 3.19. Pozície priamok p_1 a p_3.

orientovanej priamky $\mathbf{a}_k\mathbf{b}_l$. Označme ho $\mathbf{a}_p$ a nech $\mathbf{x}_{pl} = \mathbf{a}_p\mathbf{b}_l \cap p$. Lenže vidíme, že dĺžka úsečky $\mathbf{x}_{pl}\mathbf{x}_{mn}$ je väčšia ako dĺžka $\mathbf{x}_{kl}\mathbf{x}_{mn}$ (obr. 3.19 vľavo), čo je spor. Podobne by sme to ukázali, ak by bod naľavo bol z B.

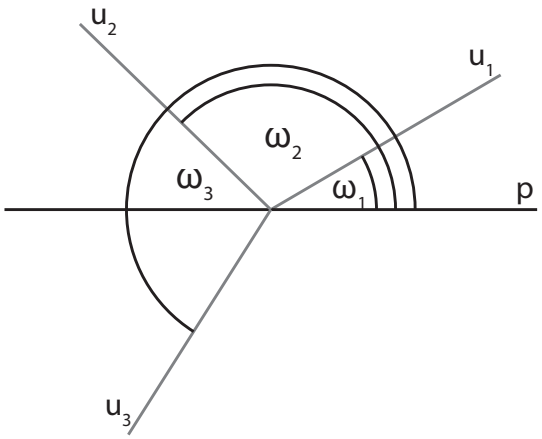
Ešte potrebujeme nájsť p_3 a p_4. Tvrdíme, že sú to priamky určené vrcholmi z A a B, ktoré zvierajú s p najmenší (p_3) a najväčší (p_4) orientovaný uhol.

Dokážeme to sporom pre p_3. Pre p_4 je dôkaz analogický. Nech je priamka p_3 určená vrcholmi $\mathbf{a}_s$ a $\mathbf{b}_t$ a existuje vrchol z A, ktorý je naľavo od p_3. Označme ho $\mathbf{a}_v$. Ale potom uhol priamky $\mathbf{a}_v\mathbf{b}_t$ je menší ako uhol $\mathbf{a}_s\mathbf{b}_t$ (obr. 3.19 vpravo), čo je spor. Podobne by sme to ukázali pre vrchol z B.

Ukázali sme, že existujú 4 spoločné oporné priamky, ešte ukážme, že ich nie je viac. Ak by okrem priamky p_1 existovala ešte jedna priamka v 1. polohe, musela by mať body $\mathbf{a}_k$,$\mathbf{b}_l$ napravo a keďže pretína priamku p, jej prienik s p by bol vľavo od $\mathbf{x}_{kl}$, čo nie je možné. Pre p_2 sa to ukáže analogicky. Podobne, ak by existovala ešte jedna priamka v 3. polohe, mala by bod $\mathbf{a}_s$ napravo a $\mathbf{b}_t$ naľavo. Pretínala by sa s priamkou p a zvierala by s ňou menší orientovaný uhol ako priamka p_3, čo sa nemôže stať. Pre p_4 by sme to dokázali analogicky. □

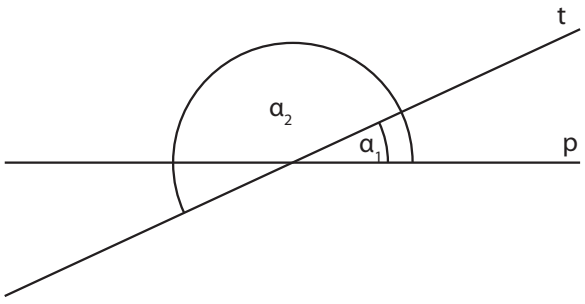
CVIČENIE 3.4 (30 bodov). *V rovine sú dané dva disjunktné konvexné mnohouholníky P_1, P_2. Koľko môžu mať P_1 a P_2 spoločných oporných priamok? Nájdite oporné polroviny algoritmom založeným výlučne na uhloch medzi hranami a vodorovnými priamkami idúcimi cez vrcholy.*

Riešenie (Michal Chládek, 22.11.2009): Nech počet vrcholov prvého mnohouholníka je n a počet vrcholov druhého mnohouholníka je m. Označme vrcholy prvého mnohouholníka $a_0, \ldots, a_{n-1}$ a vrcholy druhého mnohouholníka $a_n, \ldots, a_{n+m-1}$. Ďalej označme vodorovné priamky idúce cez vrcholy mnohouholníkov $p_0, \ldots, p_{n+m-1}$(priamka p_i ide cez vrchol a_i). Uhol medzi priamkou idúcou cez vrchol a hranou incidentnou s vrcholom budeme chápať ako uhol z intervalu $\langle 0, 2\pi)$ znázornený na obrázku 3.20.



Obr. 3.20. Uhly medzi úsečkami u_1, u_2, u_3 a priamkou p.

Pri uhle priamky s vodorovnou priamkou zadefinujeme dva uhly ako vidno na obrázku 3.21.



Obr. 3.21. Uhol medzi priamkou t a vodorovnou priamkou p.

Uhly α_1 a α_2 delia interval $\langle 0, 2\pi)$ na dve časti($\langle 0, \alpha_1\rangle \cup \langle \alpha_2, 2\pi)$ a $\langle \alpha_1, \alpha_2\rangle$). Uvažujme kedy je priamka t prechádzajúca vrcholom a_i oporná. Priamka t je oporná ak uhly obidvoch hrán mnohouholníka incidentných s vrcholom a_i patria do rovnakej časti intervalu(platí len pre konvexné mnohouholníky).

Teraz môžeme pristúpiť k hľadaniu oporných polrovín. Vieme, že oporné priamky sú presne štyri. Z toho ale dve určujú opačné polroviny pre každý mnohouholník. Teda spoločné oporné polroviny sú dve.

Algoritmus popíšeme len pre hľadanie jednej opornej polroviny. Pri hľadaní druhej opornej polroviny by sme postupovali obdobne. Keďže mnohouholníky sú konvexné, existuje deliaca priamka. Označme ju d. Zrotujeme mnohouholníky tak aby priamka d bola rovnobežná s osou y. Zvoľme náhodne dva vrcholy a_i a a_j, tak aby vrchol a_i patril prvému a a_j druhému mnohouholníku. Označme priamku prechádzajúcu týmito vrcholmi t.

Ďalej označme nasledujúce uhly:

α_1**:** uhol medzi hranou $a_{i-1}a_i$ a priamkou p_i
α_2**:** uhol medzi hranou $a_{i+1}a_i$ a priamkou p_i
β_1**:** uhol medzi hranou $a_{j-1}a_j$ a priamkou p_j
β_2**:** uhol medzi hranou $a_{j+1}a_j$ a priamkou p_j

Iteratívne nájdeme nasledujúcim algoritmom založenom na konvexnosti mnohouholníkov vrcholy s maximálnou y-ovou súradnicou.

$i \leftarrow$ náhodné číslo z množiny $0, \ldots, n-1$
if $\alpha_1 < \pi$ **then**
  while $\alpha_1 < \pi$ **do**
    $i \leftarrow i - 1$
    if $i < 0$ **then**
      $i \leftarrow n$
    end if
    vypočítaj uhly α pre nový index i
  end while
else
  while $\alpha_2 < \pi$ **do**
    $i \leftarrow i + 1 \; mod \; n$
    vypočítaj uhly α pre nový index i
  end while
end if

Vrchol s maximálnou y-ovou súradnicou v prvom mnohouholníku je a_i. Podobne zistíme maximálny vrchol aj v druhom mnohouholníku, len namiesto uhlov α_1 a α_2 použijeme uhly β_1 a β_2 a namiesto indexu i budeme meniť index j.

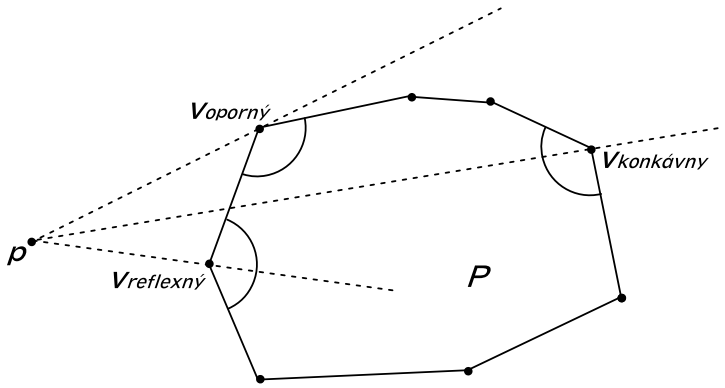
Teraz ešte upravíme indexy i a j tak aby, priamka idúca cez vrcholy a_i a a_j určovala opornú polrovinu.

V nasledujúcom algoritme predpokladáme, že všetky vrcholy prvého mnohouholníka majú menšiu x-ovú súradnicu ako vrcholy druhého mnohouholníka a že najväčšia y-ová súradnica vrcholu v prvom mnohouholníku je menšia ako najväčšia y-ová súradnica vrcholu v druhom mnohouholníku. Ak tomu tak nie je, môžeme to zabezpečiť preznačením vrcholov a otočením obidvoch mnohouholníkov.

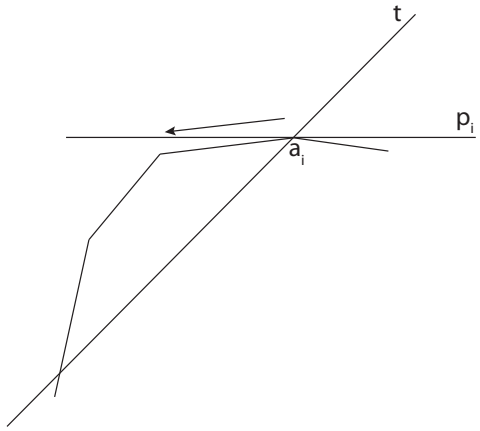
$i \leftarrow$ náhodné číslo z množiny $0, \ldots, n-1$
$j \leftarrow$ náhodné číslo z množiny $n, \ldots, n+m-1$
$t \leftarrow$ priamka prechádzajúca vrcholmi a_i a a_j

\\test či je priamka t nie je oporná
while (α_1 a α_2 nepatria do rovnakej časti intervalu určeného priamkou t) a (β_1 a β_2 nepatria do rovnakej časti intervalu určeného priamkou t) **do**
  \\test či je priamka t nie je oporná pre prvý mnohouholník
  if α_1 a α_2 nepatria do rovnakej časti intervalu určeného priamkou t **then**
    \\test, ako upraviť indexi, keď nepoznáme orientáciu mnohouholníka
    if $\alpha_1 < \alpha_2$ **then**
      $i \leftarrow i - 1$
      if $i < 0$ **then**
        $i \leftarrow n$
      end if
    else
      $i \leftarrow i + 1 \; mod \; n$
    end if
    vypočítaj uhly α_1 a α_2 pre nový index i
  else if β_1 a β_2 nepatria do rovnakej časti intervalu určeného priamkou t **then**
    if $\beta_1 > \beta_2$ **then**
      $j \leftarrow j - 1$
      if $j < n$ **then**
        $j \leftarrow n + m - 1$
      end if
    else
      $j \leftarrow j + 1$
      if $j = n + m$ **then**
        $j \leftarrow n$
      end if
    end if
    vypočítaj uhly β_1 a β_2 pre nový index j
  end if
  $t \leftarrow$ priamka prechádzajúca vrcholmi a_i a a_j
end while

Algoritmus upravuje indexy i a j tak, aby postupne konvergovali k indexom, ktoré určujú spoločnú opornú priamku.



Obr. 3.23. Tri druhy vrcholov podľa klasifikácie.



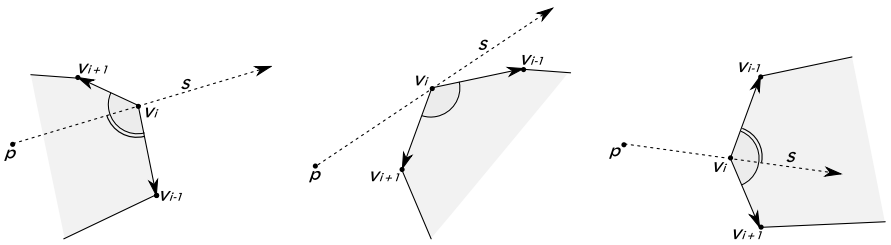
Obr. 3.22. Zmena indexu v smere šípky.

Výsledkom algoritmu sú indexy i a j, ktoré určujú oporný priamku $a_i a_j$. Keďže sme postupovali z najvyšších vrcholov, hľadaná oporná polrovina je pod opornou priamkou. □

Cvičenie 3.5 (20 bodov). *Daný je bod p a vrchol v konvexného mnohouholníka P. Načrtnite algoritmus klasifikácie vrchola v vzhľadom na $\overline{pv}$(konkávny, oporný, reflexný).*

Riešenie (Elena Dušková, 29.10.2008): Podľa definície (pozri obr. 3.23): Konkávny -– vnútro úsečky PV má neprázdny prienik s vnútorným uhlom pri vrchole V. Oporný -– vnútro úsečky PV má prázdny prienik s vnútorným uhlom pri vrchole V a ramená uhla pri vrchole V sú v tej istej polrovine. Reflexný -– vnútro úsečky PV má prázdny prienik s vnútorným uhlom pri vrchole V a ramená uhla pri vrchole V nie sú v tej istej polrovine.

Označme: $s = V - P$, u_1, u_2 vektory ramien uhla pri vrchole V (k predchádzajúcemu a nasledujúcemu vrcholu) Nasledovné riešenie nepracuje s extrémnymi prípadmi: 180 stupňový uhol medzi u_1 a u_2



OBR. 3.24. (a) konkávny vrchol V_i, (b) oporný vrchol V_i, (c) reflexný vrchol V_i.

(nastáva problém neidentifikovateľnosti vnútornej a vonkajšej časti n-uholníka) a nulová vzdialenosť 2 bodov na konvexnom obale (nulový vektor).

V nasledujúcom texte budeme potrebovať:

$$\begin{aligned} \cos(uv) &= \text{skalar}(u,v) : (|u||v|) \\ \text{skalar}(u,v) &= (u.x * v.x + u.y * v.y) \\ \sin(uv) &= \det(u,v) : (|u||v|) \\ \det(u,v) &= (u.x * v.y \breve{} u.y * v.x) \\ |u| &= \sqrt{(u.x^2 + u.y^2)} \end{aligned}$$

(4) ostrý uhol medzi vektormi $u, v \leftrightarrow 1 > skalar(u,v) > 0$

(5) pravý uhol medzi vektormi $u, v \leftrightarrow skalar(u,v) = 0$

(6) tupý uhol medzi vektormi $u, v \leftrightarrow -1 < skalar(u,v) < 0$

$$\begin{aligned} \sin(\alpha+\beta) &= \sin\alpha\cos\beta + \cos\alpha\sin\beta \\ \cos(\alpha+\beta) &= \cos\alpha\cos\beta - \sin\alpha\sin\beta \\ \sin(\alpha-\beta) &= \sin\alpha\cos\beta - \cos\alpha\sin\beta \\ \cos(\alpha-\beta) &= \cos\alpha\cos\beta + \sin\alpha\sin\beta \end{aligned}$$

Úloha sa podľa definície delí na 2 časti:

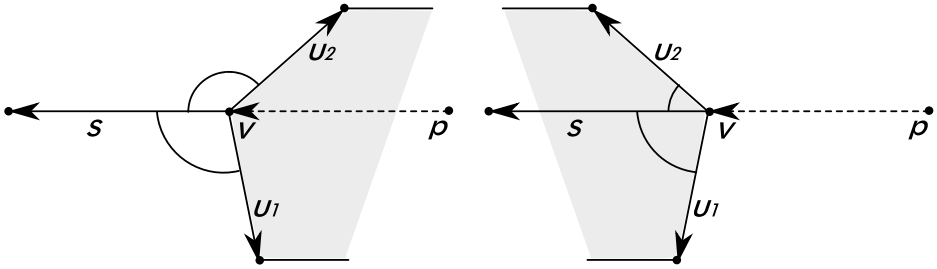
(1) zistenie či PV má prienik s vnútorným uhlom,

(2) či sa obidva vektory nachádzajú v rovnakej polrovine.

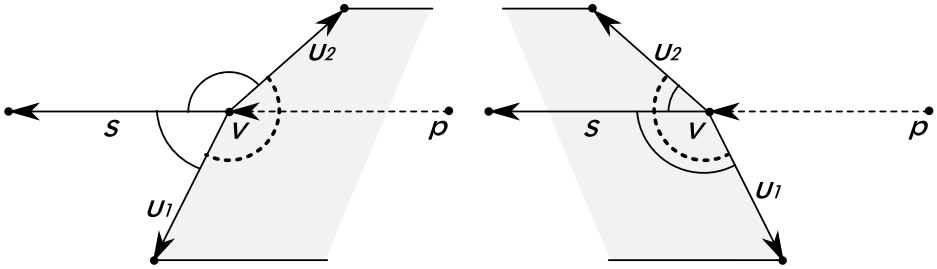
1. Špeciálne, ak sú obidva uhly su_1 aj su_2 tupé (ich súčet je určite väčší ako 180°) a zároveň vektory u_1, u_2 ležia v rozdielnych polrovinách priamky PV, tak vrchol je konkávny. Ak by boli obidva uhly su_1 aj su_2 ostré (ich súčet je menší ako 180°) a vektory u_1 a u_2 ležia v rozdielnych polrovinách priamky PV, tak určite ide o reflexný vrchol.

V prípade, keď je jeden uhol tupý, druhý ostrý a zároveň sú vektory u_1, u_2 v rozdielnych polrovinách, tak nevieme určiť, o ktorý prípad sa jedná len na základe ostrosti/tuposti uhlov.

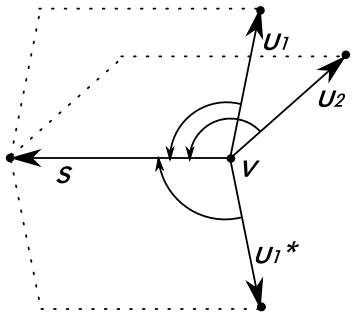
Ak sa súčet uhlov $u_2s + su_1$ rovná uhlu u_2u_1 (zistíme pomocou sin a cos uhlov, vzťahy uvedené vyššie), tak je to reflexný vrchol, v opačnom



Obr. 3.25. Konkávny a reflexný vrchol



Obr. 3.26. konkávny (uhol u_2s + su_1 sa nerovná uhlu u_2u_1) a reflexný vrchol ($u_2s + su_1 = u_2u_1$)



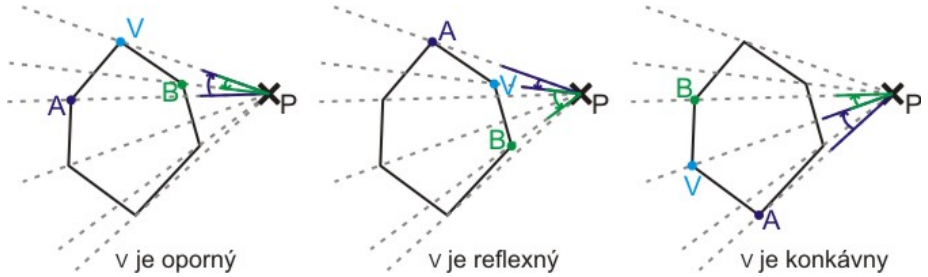
Obr. 3.27. Geometrická interpretácia determinantu dvoch vektorov.

prípade to je konkávny vrchol. Toto platí pre všetky možnosti, kde u_1 a u_2 sú na rozdielnych stranách priamky PV. V algoritme budeme najskôr realizovať podmienku 2, čím už budeme mať zaručené rozloženie vektorov u_1, u_2 na opačných polrovinách priamky PV. Pokiaľ by sme uvažovali orientované uhly, mohlo by byť poradie podmienok opačné.

2. Geometrický význam determinantu dvoch vektorov v rovine je orientovaný obsah rovnobežníka. To znamená, že ak počítame $\det(s, u1)$ a $\det(s, u2)$ a vektory u_1, u_2 sú v rovnakej polrovine od s, tak dostaneme rovnaké znamienko determinantu, veľkosť nás v tomto prípade nezaujíma. Pre $\det(s, u1)$ a $\det(s, u_1^*)$ dostaneme opačné znamienka, lebo majú opačnú orientáciu (sú v rôznych polrovinách od s).

Algoritmizácia postupu:

```
Funkcia Determinant(a,b: vektor)
        //determinant 2*2
        Result = a.x * b.y - b.x * a.y
end

Funkcia Skalár(u,v: vektor)
      //skalárny súčin 2 vektorov v 2 rozmeroch
      Result = (u.x * v.x + u.y * v.y)
end

Funkcia Veľkosť(u: vektor)
      //|u| = sqrt (u.x^2 + u.y^2)
      Result = sqrt (u.x^2 + u.y^2)
end

Funkcia Sin(u,v: vektor)
      //sin (uv) = det (u,v) : (|u||v|)
      Result = Determinant(u,v) : (Veľkosť(u) * Veľkosť(v))
end

Funkcia Cos(u,v: vektor)
      //cos (uv) = skalar (u,v) : (|u||v|)
      Result = Skalár(u,v) : (Veľkosť(u) * Veľkosť(v))
end

Funkcia Rovnaká_polrovina(s,u1,u2 : vektor)
         // determinanty majú rovnaké znamienko => rovnaká polrovina
      If ((Determinant(s,u1) * Determinant(s,u2)) > 0)
         Result = TRUE
      Else
         Result = FALSE
end

Funkcia Prienik_s_vnútorným_uhlom(s,u1,u2 : vektor)
      //ak (Uhol(u2,u1) = Uhol(u2,s) + Uhol(s,u1))
      //sin (u+v) = sin u cos v + cos u sin v
      If (Sin (u2,u1) = Sin(u2,s)*Cos(s,u1) + Cos(u2,s)*Sin(s,u1))
         Result = FALSE
      Else
         Result = TRUE
end

Funkcia Typ_vrcholu(i : poradie_vrcholu)
      If (Rovnaká_polrovina(s,u1,u2))
         Result = ,,Oporný‘‘
```



Obr. 3.28

```
      Else
         If (Prienik_s_vnútorným_uhlom(s,u1,u2))
            Result = ,,Konkávny‘‘
         Else
            Result = ,,Reflexný‘‘
end
```

□

Riešenie (Marta Režnáková, 26.10.2008): Majme mnohouholník zadaný ako postupnosť bodov (kde ako nasledovník posledného bodu je prvý bod a predchodca prvého posledný bod) a majme bod V, ktorý je vrcholom daného mnohouholníka a bod P ležiaci mimo mnohouholníka (poz. 3.28). Bod ležiaci pred bodom V v danej postupnosti bodov označme A a bod ležiaci za bodom V v danej postupnosti bodov označme B. Majme orientované uhly $\alpha = \angle_{or}APV$ a $\beta = \angle_{or}VPB$. Ak sú opačne orientované, bod V je oporný, ak sú oba kladne orientované, bod V je reflexný a ak sú oba záporne orientované, bod V je konkávny.

Algoritmus:

- zisti orientáciu uhlov $\alpha = \angle_{or}APV$ a $\beta = \angle_{or}VPB$;

- ak je báza smerových vektorov priamok $\overline{PA}$ a $\overline{PV}$ orientovaná kladne, je aj uhol α orientovaný kladne

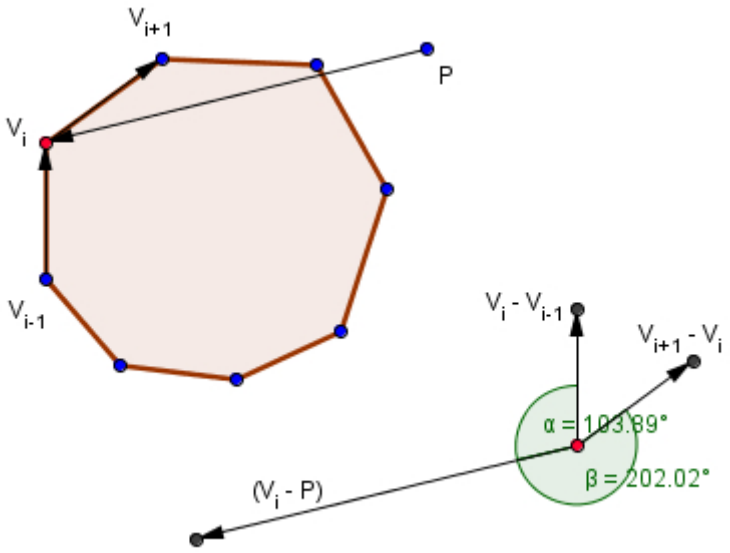
- orientovaný uhol môžeme vypočítať pomocou vťahov:

$\alpha = \angle_{or}APV = \arccos \frac{u.v}{|u||v|}$, kde u.v je skalárny súčin vektorov u a v, alebo $\alpha = \angle_{or}APV = \arcsin \frac{\det(u,v)}{|u||v|}$, kde u je smerový vektor $\overline{PA}$ a v je smerový vektor $\overline{PV}$

- výsledný uhol α je z intervalu $(-\pi, \pi >$, kde znamienko určuje orientáciu

- analogicky postupujeme aj pre uhol β

if(α a β sú opačne orientované){
    bod v je oporný;
}else if(α je kladne orientovaný){ *//stačí nám kontrolovať orientáciu jedného z uhlov, pretože rozdielna orientácia je už vylúčená*
    bod v je reflexný;



Obr. 3.29. Uhly dôležite pre klasifikáciu bodov.

```
}else{ //uhol α môže byť už len záporný
    bod v je konkávny;
}
```

□

Riešenie (Rastislav Halamíček, 30.10.2008): Naším cieľom je zostaviť algoritmus, ktorý pre daný vstup (súradnice skúmaného bodu P a index vrcholu konvexného mnohouholníka) vyhodnotí vzťah bodu P k vrcholu V_i jednou z možností V_i je vzhľadom na daný bod oporný, konkávny alebo reflexný.

Na obrázku 3.29 máme znázornenú situáciu, v ktorej je V_i vzhľadom na P konkávny. (Na tom istom obrázku by bol oporný napríklad V_{i+1} a reflexný by bol ktorýkoľvek vrchol úsečky, ktorú pretína vektor $V_i - P$.)

Pre identifikáciu typu vrcholu budeme skúmať tri dôležité vektory:

$$V_i - P, \quad V_{i+1} - V_i, \quad V_i - V_{i-1}.$$

Predpokladajme, že rovina je kladne orientovaná dvojicou vektorov $\vec{\mathbf{e}}_1, \vec{\mathbf{e}}_2$. Umiestnime tieto tri vektory do jedného bodu, a potom sa zaoberajme myšlienkou, akú orientáciu roviny určí usporiadaná dvojica vektorov $V_i - V_{i-1}$, $V_i - P$ a tiež usporiadaná dvojica vektorov $V_i - P$, $V_{i+1} - V_i$. Zaujíma nás iba súhlasnosť, či nesúhlasnosť s orientáciou roviny. Teda, dvojica vektorov $(\vec{\mathbf{a}}, \vec{\mathbf{b}})$ je orientovaná súhlasne s orientáciou roviny, ak $\det(\vec{\mathbf{a}}, \vec{\mathbf{b}}) > 0$ a orientovaná nesúhlasne ak $\det(\vec{\mathbf{a}}, \vec{\mathbf{b}}) < 0$.

Obrázok nám možno pomôže nahliadnuť, že platí takáto klasifikácia:

dvojica vektorov	$V_i - V_{i-1}, V_i - P$	$V_i - P, V_{i+1} - V_i$	typ vrcholu
znamienko determinantu	+	+	oporný
	-	-	oporný
	+	-	konkávny
	-	+	reflexný

Myšlienky teda možno poľahky zosumarizovať do algoritmu pre zistenie typu vrcholu s konštantnou zložitosťou. V algoritme sa predpokladá, že konvexný mnohouholník je reprezentovaný zoznamom vrcholov. Ak $i - 1 < 1$ alebo $i + 1 > n$, kde n je počet vrcholov, preskakuje sa na koniec, resp. začiatok zoznamu vrcholov (zoznam je cyklický).

```
označenia
vp = vektor V_i − P
v1v = vektor V_{i+1} − V_i
vv1 = vektor V_i − V_{i−1}

vstup
    index i skúmaného vrcholu
    súradnice skúmaného bodu P

počítaj
    vyrátaj súradnice vektorovov vp, v1v, vv1
    d1 = sgn ( det (vv1, vp))
    d2 = sgn ( det (vp, v1v))

výstup
    ak (d1=d2) vráť (P je k Vi oporný)
    ak (d1<d2) vráť (P je k Vi reflexný)
    ak (d1>d2) vráť (P je k Vi konkávny)
```

Poznámky: Ešte môže nastať jeden zádrheľ – determinant môže byť nulový. To by znamenalo, že P leží na predĺžení strany konvexného mnohouholníka. Neviem, ako sa klasifikuje takýto bod. Potom možno celú úsečku chápať ako opornú, a teda aj obidva jej koncové vrcholy. Samozrejme, situácia sa dá triviálne ošetriť aj v algoritme pridaním podmienky.

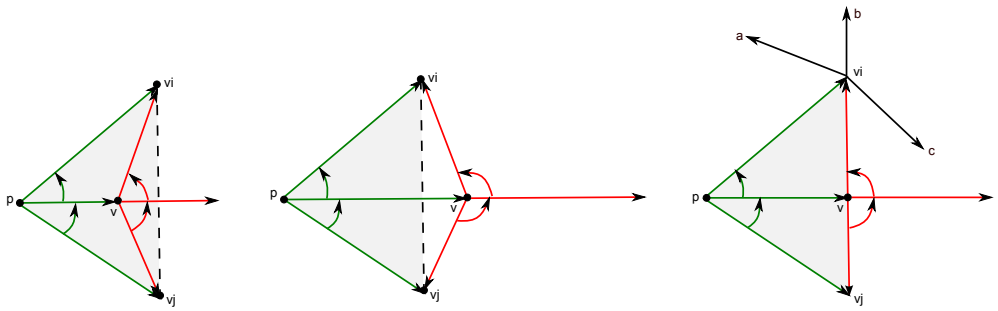
Keďže situácia sa odohráva v rovine, determinant počítame klasicky $\det(\vec{\mathbf{a}}, \vec{\mathbf{b}}) = a1.b2 - a2.b1$. □

Riešenie (Martina Bátorová, 29.10.2008):

(1) Vrchol je *oporný*, ak celý mnohouholník P leží v jednej polrovine určenej priamkou pv. Keďže P je konvexný, opornosť vrchola sa dá jednoducho overiť nasledovne: označme susedné body vrchola v ako v_i a v_j (teda uhol v P pri v je $\angle v_j v v_i$). Teraz zistime, ako je orientovaná báza $b_i := \{v - p, v_i - p\}$ a $b_j := \{v_j - p, v - p\}$, teda vypočítajme znamienko determinantov

oboch báz. Ak sú rovnako orientované, t.j $\text{sgn}(b_i) = \text{sgn}(b_j)$, bod v nie je oporný, pretože vrcholy v_i a v_j ležia vzhľadom na priamku pv v rôznych polrovinách. V prípade nerovnosti znamienok je bod v oporný.

(2) V prípade *konkávneho* a *reflexného* vrcholu rovnako ako v prípade oporného zistime orientácie báz b_i a b_j. Bod je konkávny alebo reflexný v prípade, že znamienka determinantov sú rovnaké. Teraz však potrebujeme kritérium, ktoré rozhodne, akého typu je v.



Obr. 3.30. Na obrázku vľavo vidíme príklad reflexného bodu, v strede príklad konkávneho bodu, napravo príklad, kedy nevieme rozhodnúť iba pomocou priamych susedov bodu v.

(3) Ak sa na obrázku pozrieme na trojuholník Δpv_jv_i vidíme, že bod v leží buď v jeho *vnútri*, *vonku*, alebo na *hranici*. Na klasifikáciu v využijeme veľkosti uhlov v bázach b_i, b_j.

Vypočítajme uhly vektorov $\angle(v_j - v, v - p)$ a $\angle(v - p, v_i - v)$, teda skalárne súčiny $s_j := (v_j - v).(v - p)$ a $s_i := (v - p).(v_i - v)$:

(a) Ak jedno z čísel s_i, s_j je kladné a druhé je kladné, záporné alebo 0, bod v klasifikujeme ako *reflexný*.

(b) Ak jedno z čísel s_i, s_j je záporné a druhé je záporné alebo 0, bod v klasifikujeme ako *konkávny*.

(c) Ak obe čísla s_i, s_j sú 0, čiže oba počítané uhly sú pravé, bod v nevieme klasifikovať iba na základe jeho priamych susedov v_i, v_j. V takomto prípade sa musíme pozrieť na nasledovníka v: pre prípad a je bod konkávny, pre prípad c reflexný. Prípad b nás znova stavia pozície, kedy musíme opakovať celú klasifikáciu. Toto vetvenie je však určite konečné, pretože uvažovaný mnohouholník je konvexný a má len konečný počet hrán - v nejakom kroku napokon konkávnosť/reflexnosť rozhodneme.

(4) Procedúru môžeme potom zjednodušene zapísať napr. takto:

```
procedure KlasifikujBod(Bod p, Bod v, Mnohouholnik P){
```

```
        vi := P.next(v);
        vj := P.prev(v);
        bi := {v - p, vi - p};
        bj := {vj - p, v - p};
        di := Det(bi);
        dj := Det(bj);
        if (Sgn(di) != Sgn(dj)) {
          return 'oporny';
        }
        else {
          si := Skalarne(v - p, vi - v);
          sj := Skalarne(vj - v, v - p);
          if ( (si==0) && (sj==0) ) {
            v  := vi;
            vi := P.next(v);
            vj := P.prev(v);
            opakuj rozhodovanie z vetvy 'else';
          }
          if ( (si>0) && ( (sj==0) || (sj<0) || (sj>0) ) ||
               (sj>0) && ( (si==0) || (si<0) || (si>0) ) )
            return 'reflexny';
          if ( (si<0) && ( (sj==0) || (sj<0) ) ||
               (sj<0) && ( (si==0) || (si<0) ) )
            return 'konkavny';
        }
    }
```

□

CVIČENIE 3.6 (50 bodov). *Nech S je množina obsahujúca n bodov s celočíselnými súradnicami v rovine. Súradnice sú dané v rozmedzí* 1 *a n^d, kde d je konštanta. Ukážte, že konvexný obal S sa dá nájsť v lineárnom čase.*

Riešenie (Jana Hlinková, 31.10.2008): Riešenie spočíva v použití radix sortu na utriedenie bodov podľa x-ovej súradnice a následnom využití Grahamovho prechodu. Keďže radix sort beží v čase $\mathcal{O}(n)$ ako aj Grahamov prechod, celková časová zložitosť bude $\mathcal{O}(n)$. Najprv popíšem algoritmus radix sort, a priložím dôkaz časovej zložitosti podľa *Design and Analysis of Computer Algorithms, David M. Mount.*

- Radix sort triedi čísla postupne podľa cifier, začínajúc na najmenej dôležitej pozícii. Nech P je pole celých čísel, ktoré majú maximálne d cifier (v ľubovoľnej číselnej sústave). Triedenie prebieha nasledovne: čísla, ktoré majú menej ako d cifier doplníme spredu nulami. Potom čísla triedime postupne podľa

jednotlivých cifier, od najmenšieho rádu po najväčší. Triedenie podľa jednej cifry vieme urobiť v $\mathcal{O}(n+k)$ čase, napríklad použitím spájaných zoznamov pre každú hodnotu cifry (v desiatkovej sústave bude 10 zoznamov: 0..9). Pri prechode poľom ukladáme prvky (čísla) do príslušného spájaného zoznamu podľa hodnoty cifry, nakoniec spájané zoznamy prepíšeme naspäť do pôvodného poľa, pričom prepisujeme najprv zoznam pre najmenšiu hodnotu, atď. Pokračujeme triedením podľa ďalšej cifry. Kedže cifier je d, celková časová zložitosť je $\mathcal{O}(d(n+k))$. Ak d je konštanta, potom $\mathcal{O}(d(n+k)) = \mathcal{O}(n+k) = \mathcal{O}(n)$ (pričom k je počet rôznych hodnôt cifry, čo je $\mathcal{O}(n)$).

- Majme pole čísel v rozsahu 1 až n^d. Najprv odčítame od každého čísla jednotku a dostaneme rozsah 0 až $n^d - 1$. Každé číslo z tohto rozsahu vieme v sústave so základom n reprezentovať d-ciferným číslom (d je konštanta). Môžeme použiť radix sort na utriedenie týchto čísel v n-kovej sústave, pričom časová zložitosť bude $\mathcal{O}(d(n+n)) = \mathcal{O}(n)$. Čísla napokon prepíšeme do pôvodného tvaru.

Na nájdenie konvexného obalu teraz môžeme použiť Grahamov prechod podľa *Computational Geometry, David M. Mount.*

P	d=1	d=2	d=3
123	731	109	109
249	112	112	112
112	562	123	123
562	123	731	249
731	249	249	359
359	359	359	562
109	109	562	731

OBR. 3.31. Ilustrácia radix sortu. (d označuje práve spracovávanú cifru)

□

Riešenie (Pavol Beluško, 3.11.2008): Pokúsime sa previesť tento prípad na prípad z cvičenia 3.7, takisto budú použité triediace algoritmy z cvičenia 3.7. Návrh algoritmu a analýza časovej zložitosti.

(1) Formálny krok prevodu súradníc na polárne preberieme z cvičenia 3.7, je $\mathcal{O}(n)$ zložitý.

(2) Utriedenie – v tomto kroku narážame na problémový rozdiel oproti prípadu z cvičenia 3.7. Máme totiž nelineárne veľa n^d hodnôt, čiže pôvodný algoritmus využívajúci Counting sort by vyžadoval $\mathcal{O}(n^d)$ krokov. Preto potrebujeme triediaci algoritmus modifikovať.

Pozn.: V nasledujúcom odstavci je popísané modifikované usporiadanie podľa jednej súradnice. Podobne ako v cvičení 3.7, aj tu musíme utriediť podľa oboch súradníc. Tento fakt nemá vplyv na asymptotickú zložitosť.

Máme teda n čísel spomedzi n^d hodnôt v rozmedzí $1 \dots n^d$, ktoré potrebujeme utriediť v lineárnom čase. Od čísel odpočítame 1, dostaneme ich tak do rozsahu $0..n^d - 1$. Pre n bodov to znamená $\mathcal{O}(n)$ operácií. Tieto čísla vieme zapísať v n-kovej sústave, pričom všetky čísla z nášho rozsahu vieme zapísať vo forme vektora dĺžky d, ktorý ho jednoznačne reprezentuje (Napr. nech $n = 3, d = 4$ a nech $i = 81 = n^d$. Potom $i - 1 = 80 = 2 * 3^3 + 2 * 3^2 + 2 * 3^1 + 2 * 3^0$, vo forme vektora $i - 1 =< 2, 2, 2, 2 >$). Konverziu jedného čísla možno vykonať v konštantnom čase, čo pre n bodov znamená $\mathcal{O}(n)$ operácií. Jednotlivé súradnice vektora budú v rozsahu $0 \dots n - 1$, spolu $\mathcal{O}(n)$ veľa hodnôt. Na $\mathcal{O}(n)$ veľa hodnotách beží Counting sort v čase $\mathcal{O}(n)$. Takto reprezentované čísla možno teda usporiadať pomocou Radix Sortu, ktorý na čiastkové triedenia súradníc vektorov používa Counting sort v čase $\mathcal{O}(n)$. Spätná konverzia a prirátanie 1 si vyžiada pre n bodov $\mathcal{O}(n)$ operácií.

Celý krok utriedenia bodov teda trvá $\mathcal{O}(n)$ dlho.

(3) Aplikujeme Grahamov prechod, ktorý je $\mathcal{O}(n)$ zložitý a získame konvexný obal.

Celková časová zložitosť algoritmu je teda $\mathcal{O}(n)$. □

Cvičenie 3.7 (50 bodov). *Nech S je množina obsahujúca n bodov s celočíselnými súradnicami v rovine. Súradnice sú dané v rozmedzí 1 a m. Nájdite algoritmus na zostrojenie konvexného obalu S v čase $\mathcal{O}(n + m)$.*

Riešenie (Pavol Beluško, 28.10.2008):Základom algoritmu je poznatok, že ak poznáme všetky možné hodnoty pre vstupné dáta, vieme ich utriediť v lineárnom čase $\mathcal{O}(m)$, kde m je počet možných hodnôt.

Postup:

(1) Body prevedieme do polárnych súradníc, aby sme na nich mohli použiť Grahamov prechod.

Ak by sme zvolili vlastný stred sústavy polárnych súradníc, prevodom všetkých možných usporiadaných dvojíc $[x, y], x \in$

$1 \dots m, y \in 1 \dots m$ by sme dostali m^2 rôznych polárnych súradníc čiže samotný prevod by bol $\mathcal{O}(m^2)$. Preto za stred sústavy súradníc zvolíme nevlastný bod osi y, čím limitujeme vplyv uhlovej zložky polárnych súradníc na m možných hodnôt. Vzdialenostná zložka polárnych súradníc bude nadobúdať hodnoty od $1 \dots m$. Môžme takto stotožniť kartézske súradnice s polárnymi.

(2) Body utriedime podľa polárnych súradníc v čase $\mathcal{O}(m)$ pomocou algoritmu Radix sort, na čiastkové triedenia súradníc použijeme lineárny algoritmus Counting sort (J. Procházka: skriptá Algoritmy a dátové štruktúry, kapitola triedenie v lineárnom čase).

Príklad na obrázku 3.32 zobrazuje utriedenú postupnosť; stred sústavy polárnych súradníc máme v nevlastnom bode osi y. V takomto prípade triedime najprv podľa druhej súradnice, potom podľa prvej.

Pozn.: Body majú dve súradnice a teda triedime dvakrát, na asymptotickú časovú zložitosť to však nemá vplyv. Táto zostáva $\mathcal{O}(m)$. Pozn.: Pri triedení po zložkách je dôležité použiť stabilné triedenie.

(3) Aplikujeme Grahamov prechod a získame konvexný obal.

Analýza časovej zložitosti:

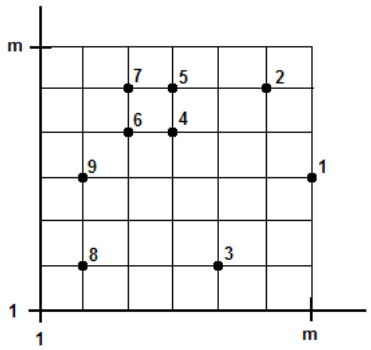
(1) Zmena sústavy súradníc (formálny krok) - $\mathcal{O}(n)$
(2) Utriedenie bodov - $\mathcal{O}(m)$
(3) Grahamov prechod - $\mathcal{O}(n)$

Celková časová zložitosť je teda $\mathcal{O}(n + m)$. Ak m je $\mathcal{O}(n)$, tak časová zložitosť je $\mathcal{O}(n)$.

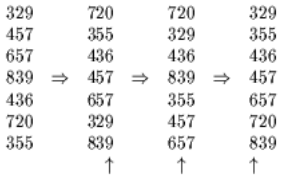
Analýza pamäťovej zložitosti: Counting sort používa 2 polia dlhé $\mathcal{O}(n)$ a jedno pole dlhé $\mathcal{O}(m)$. Vstupné a výstupné pole je dlhé $\mathcal{O}(n)$, takže pamäťová náročnosť je $\mathcal{O}(n + m)$.

Pseudokód (optimalizovaný - pri implementácii v tomto prípade nepotrebujeme uvažovať o pojme polárne súradnice):

```
KolekciaBodov RadixSort (KolekciaBodov Body)
{
  //triedime podľa druhej súradnice
  Body = CountingSortY(Body);
  //triedime podľa prvej súradnice
  Body = CountingSortX(Body);
  Return Body;
}

ConvexHull CreateConvexHull (KolekciaBodov Body)
{
  //zotrieď body
```



OBR. 3.32. Ilustračný obrázok utriedených bodov, pričom stred polárnych súradníc je umiestnený do nevlastného bodu osi y



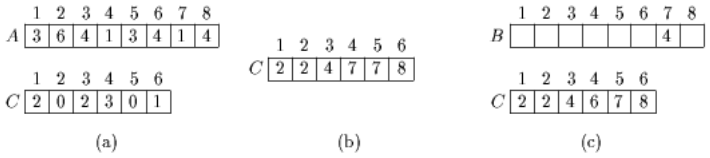
OBR. 3.33. Radix sort: Na vstupe je zoznam viaciferných čísel (v našom prípade viacrozmerných bodov). Túto postupnosť utrieďujeme postupne podľa jednotlivých pozícií číslic od cifry najnižšieho rádu po cifru najvyššieho rádu. Na obrázku je znázornená činnosť Radix sortu na zozname siedmych trojciferných čísel. Prvý stĺpec je vstup. Zvyšné stĺpce znázorňujú postupné utrieďovanie, aktuálnu cifru indikujú vertikálne šípky. Radix sort beží v čase $\mathcal{O}(n)$. (obrázok prevzatý zo skrípt ADŠ)

```
  Body = RadixSort(Body);

  //Grahamov prechod z prednášky
  Return Graham(Body);
}
```

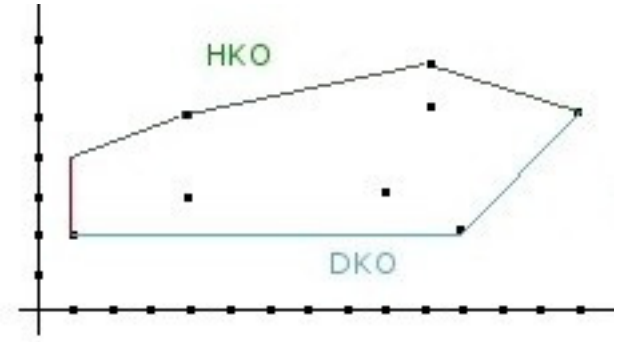
□

Riešenie (Barbora Gallusová, 31.10.2009): Konvexný obal skonštruujeme v troch krokoch. Najprv vyberieme kandidátov na horný(dolný) konvexný obal, potom spomedzi nich vyberieme tie body, ktoré tvoria horný(dolný) konvexný obal. Tretí krok bude spojiť tieto dva dokopy, ako vidieť na obr. 3.35.

1.krok: Máme n bodov v rovine. Súradnice bodov sú celé čísla v rozmedzí 1 až m. Ak sa zameriame na $x-$ovú súradnicu bodov, môžeme povedať, že kandidáti na horný konvexný obal budú len tie body, ktoré



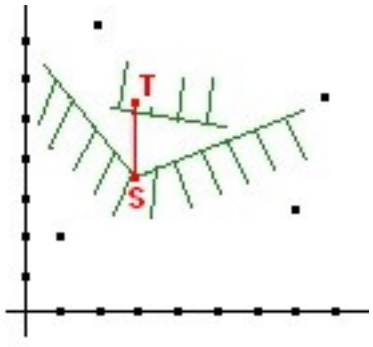
Obr. 3.34. Counting sort: Pole A obsahuje vstupnú postupnosť bodov. Pole C reprezentuje obor hodnôt (preto je dôležité, aby bol konečný), jeho prvky obsahujú počet výskytov jednotlivých hodnôt vo vstupnej postupnosti bodov (*obrázok a*). V ďalšom kroku postupujeme po poli C od najmenších hodnôt po najväčšie, pričom k i-temu prvku pričítame hodnotu $(i-1)$. prvku $(i = 2..m)$. Každý prvok poľa C teraz nesie informáciu, koľko bodov zo vstupu je menšich alebo rovných hodnote reprezentovanej daným prvkom (*obrázok b*). V ďalšom kroku postupujeme po poli A od jeho konca (kvôli stabilite triedenia), v našom prípade sa v prvom kroku jedná o bod s hodnotou 4, ten umiestnime do výstupného poľa B na pozíciu B[C[4]], následne hodnotu C[4] dekrementujeme o 1 (*obrázok c*) atď. Counting sort beží v čase $\mathcal{O}(n)$. (obrázok prevzatý zo skrípt ADŠ)



Obr. 3.35

majú maximálnu $y-$ovú súradnicu pre každé x. Keby to tak nebolo, nastal by spor s konvexnosťou(ak by bol bod S taký, že $S_y < T_y$, súčasťou KO, bod T nad ním by musel byť tiež v konvexnom obale aby neostal vonku, ale potom časť úsečky TS bude vonku, poz. obr. 3.36). Podobne to platí pre dolný konvexný obal, kandidáti budú len body s minimálnou $y-$ovou súradnicou pre každé x.

Zo všetkých n bodov teda vyberieme len kandidátov na horný, resp. dolný KO. Vytvoríme si dve polia H, D dĺžky m, vždy jedno pre kandidátov na horný(dolný) KO. Budeme postupne prehľadávať pole so zadanými n bodmi. Vždy sa pozrieme na $x-$ovú súradnicu



Obr. 3.36

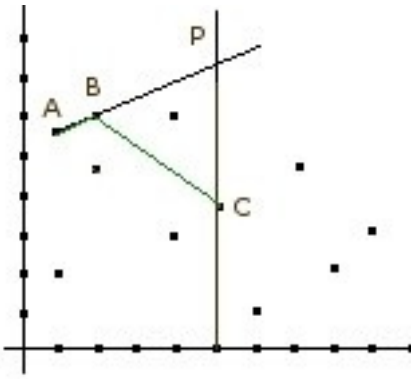
bodu a skúmame, či nie je jeho $y-$ová súradnica maximálna alebo minimálna. Robíme to tak, že sa pozrieme do vytvorených polí H, D na $x-$té miesto. Ak je toto miesto vH alebo D prázdne, skúmaný bod sa automaticky stáva kandidát na tomto mieste. Ak je miesto už obsadené, porovnáme $y-$ové súradnice skúmaného bodu s bodmi v $H(D)$. Ak je súradnica y skúmaného bodu väčšia(menšia) ako $y-$ová súradnica bodu v $H(D)$, skúmaný bod zaujme miesto kandidáta, ak nie, pokračujeme ďalej. Takto získame najviac $2m$ bodov, ktoré sú kandidátmi na tvorcov KO.

Časová zložitosť tohto kroku je $O(n)$, pretože pre každý z n bodov urobíme práve dve porovnávania, ktoré trvajú konštantný čas. Pamäťová zložitosť je $O(m+n)$ - použili sme dve polia o veľkosti m, jedno pole o veľkosti n.

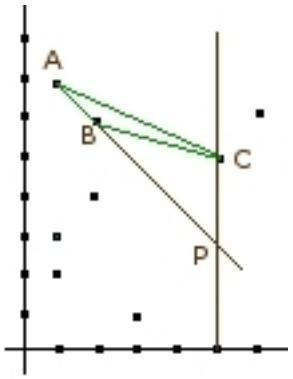
2.krok: Ďalší krok je spomedzi kandidátov vybrať len tie body, ktoré naozaj tvoria KO. Popíšeme postup na tvorbu horného KO. Obal budeme vytvárať pre body postupne, najprv len pre body $x = 1$ a $x = 2$ (resp. prvé dve x, kde sa nachádzajú body) a potom budeme postupne pridávať body s nasledujúcou $x-$súradnicou. Horný KO pre prvé dve x, kde sa nachádzajú body tvoria prvý dvaja kandidáti v H.

Pridaním nasledujúceho bodu z H, môžu nastať dve situácie. Buď sa kandidát stane tvorcom obalu a teda horný KO budú tvoriť tri body (poz. obr. 3.37), alebo sa stane tvorcom obalu, ale vyhodí predošlého tvorcu kvôli nekonvexnosti, poz. obr. 3.38.

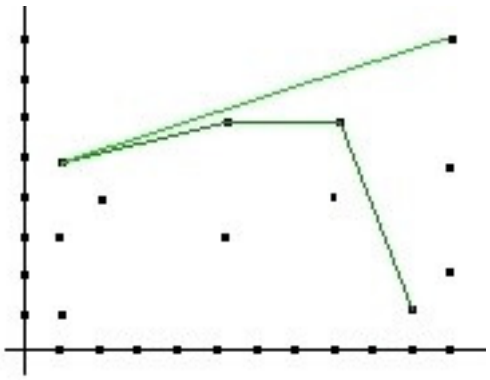
Či skúmaný bod vyhodí predošlý zistíme nasledovne. Označme skúmaný bod C, predošlé dva z H označíme A, B. Nájdeme priesečník priamky $\overleftrightarrow{AB}$ a priamky $x = x_C$ písmenom P, kde x_C je $x-$ová súradnica bodu C. $y-$súradnicu bodu P získame tak, že dosadíme x_C do rovnice $\overleftrightarrow{AB}$. Ak C leží pod týmto bodom, čiže $y_C < y_P$, zaradí sa medzi tvorcov horného KO a nevyhodí predošlý bod, pretože uhol ABC bude konvexný. Ak C leží nad týmto bodom, čiže $y_C > y_P$, zaradí sa medzi tvorcov horného KO a vyhodí predošlý bod B, pretože uhol



Obr. 3.37



Obr. 3.38



Obr. 3.39

ABC bude nekonvexný. Ak dôjde počas skúmania všetkých kandidátov k vylúčeniu bodu, je nutné sa spätne pozrieť a porovnať, či netreba vylúčiť ďalšie body. V najhoršom prípade dôjde k vylúčeniu všetkých predošlých bodov okrem prvého (poz. obr. 3.39).

Rovnakým spôsobom konštruujeme dolný KO, akurát ak je skúmaný bod C nad priesečníkom P tak nevylúči predošlý bod, ak je pod P, tak vylúči predošlý bod.

Časová zložitosť tohto vylučovania kandidátov je $O(m)$, pretože skúmame m bodov, z ktorých každý môže byť najviac raz pridaný a najviac raz vylúčený zo zoznamu bodov tvoriacich horný (dolný) KO. Porovnávanie konvexnosti je $O(1)$. Pamäťová zložitosť je $2O(m)+O(1) = O(m)$, pretože si pamätáme H, D a vyrátané konštanty a, b pre rovnicu priamky $\overleftrightarrow{AB}$ a $y-$súradnicu P. Takto postupne prezrieme celé $H(D)$ a nakoniec nám ostanú len body, ktoré skutočne tvoria horný(dolný) KO.

3.krok: Spojenie horného a dolného KO je časovej zložitosti $O(1)$. Najprv budú body H, potom od konca body z D. Pamäťová zložitosť je $2O(m) = O(m)$, pretože obe polia H, D majú dĺžku m.

Záver: Celková časová zložitosť je teda $O(n) + O(m) + O(1) = O(n+m)$. Celková pamäťová zložitosť je $O(m)+O(n) = O(m+n)$. □

Cvičenie 3.8 (25 bodov). *Popíšte inkrementálny (on line) algoritmus na vytvorenie konvexného obalu n-prvkovej množiny bodov v rovine a nájdite jeho zložitosť.*

Riešenie (Marian Maričák, 31.10.2008):Základná myšlienka inkrementálneho algoritmu je jednoduchá. Najprv si vyberieme dostatočne malú podmnožinu vstupu tak, že problém je ľahko riešiteľný. Potom pridávame body jeden po druhom, pričom v každom kroku zachovávame správnosť riešenia. Dá sa použiť napr. v situácii, keď máme k dispozícii časť bodov, ale nie všetky. Majme množinu $\mathbb{S} = \{p_1, p_2, \ldots, p_n\}$. V prvom kroku zostrojíme konvexný obal troch bodov, čo je trojuholník. Teraz chceme pridať bod p_i k existujúcemu konvexnému obalu $CH_{\text{i-1}}$. Existujú dve možnosti:

- p_i leží vnútri alebo na hranici $CH_{\text{i-1}}$. Túto vlastnosť overíme v lineárnom čase tak, že prechádzame po hranách $CH_{\text{i-1}}$ v smere hodinových ručičiek a overujeme, či p_i leží vždy vpravo od hrany. Ak áno, potom $CH_{\text{i-1}} = CH_i$.
- p_i leží mimo $CH_{\text{i-1}}$. V tomto prípade vieme nájsť dve oporné priamky z p_i ku $CH_{\text{i-1}}$, a teda $CH_i = CH_{\text{i-1}} \cup p_i$.

Pre nájdenie obidvoch oporných vrcholov nám stačí jeden prechod $CH_{\text{i-1}}$.

INCREMENTAL-CONVEX-HULL($\mathbb{S}$)
$CH =$ trojuholník(p_1, p_2, p_3)
for$(i = 4; i \leq n; i++)$ **do**
ak p_i leží mimo CH
nájdi oporné vrcholy
odstráň hrany (a vrcholy) medzi opornými vrcholmi

hspace1.5cmdo CH pridaj p_i a dve nové hrany z oporných vrcholov do p_i
return CH

Analýza algoritmu

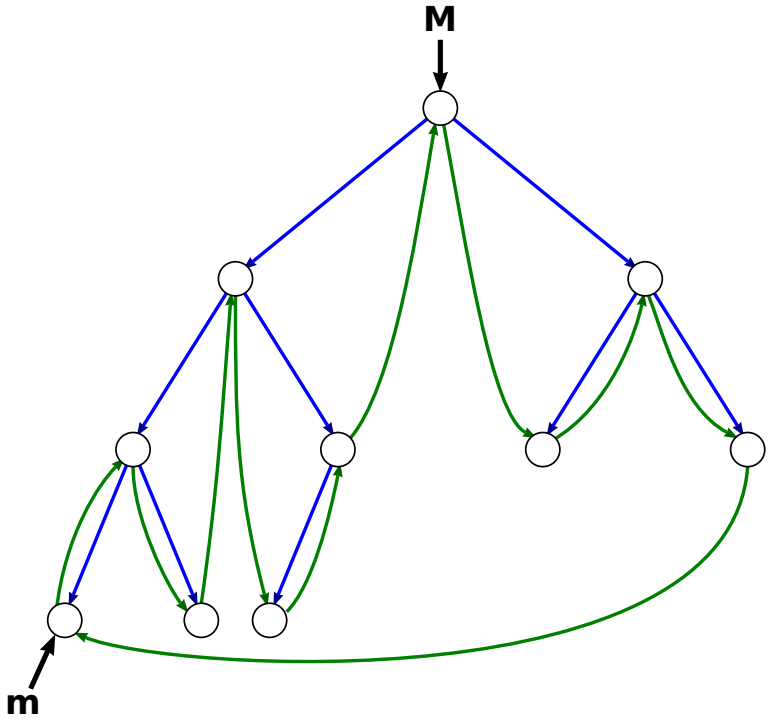
V každom kroku algoritmu prechádzame po hranách $CH_{\text{i-1}}$. V najhoršom prípade prejdeme v každom kroku všetky hrany. Zložitosť inkrementálneho algoritmu v najhoršom prípade je teda $3+4+\cdots+(n-1) = \mathcal{O}(n^2)$. **Inkrementálny on line algoritmus** Ak chceme dostať on line algoritmus, teda algoritmus, v ktorom po pridaní n-tého vrcholu je zložitosť asymptoticky $\mathcal{O}(n \log n)$, musíme použiť vhodnú dátovú štruktúru. Myšlienka zostáva rovnaká. Najprv potrebujeme určiť polohu p_i vzhľadom na $CH_{\text{i-1}}$. Funkcia $LOKALIZUJ(p_i)$ nám povie, či p_i leží mimo, na hranici alebo vnútri $CH_{\text{i-1}}$. Konvexný obal $CH_{\text{i-1}}$ si rozdelíme na horný a dolný, pričom každý z nich reprezentujeme ako spájateľnú frontu. Spájateľná fronta má tvar vyváženého binárneho vyhľadávacieho stromu. Horný konvexný obal si označíme ako T_H a dolný ako T_D. V uzle spájateľnej fronty budeme uchovávať dvojicu (v_x, v), kde v je vrchol a v_x jeho x-ová súradnica. V listoch budú uložené buď hrany, alebo prázdne polroviny. Funkcia $LOKALIZUJ(p_i)$ začne prehľadávať T_H a T_D podľa x-ovej súradnice bodu p_i. Cieľom je nájsť hranu e_D alebo vrchol v_D, tak aby kolmica na os x, vedená bodom p_i, preťala hranu e_D resp. by prechádzala vrcholom v_D (poz. obr. 3.42). Analogicky nájdeme hranu e_H resp. vrchol v_H v hornom konvexnom obale. Môžu nastať štyri prípady:

- ak nenájdeme žiadne hrany (vrcholy) - p_i je MIMO $CH_{\text{i-1}}$
- ak p_i leží na e_D (v_D) alebo leží na e_H (v_H) - p_i leží NA $CH_{\text{i-1}}$
- ak sa p_i nachádza nad e_D (v_D) a zároveň pod e_H (v_H) - p_i je VNÚTRI $CH_{\text{i-1}}$
- inak - p_i je MIMO $CH_{\text{i-1}}$

Vyhľadávanie v spájateľnej fronte má časovú zložitosť $\mathcal{O}(\log n)$, čiže časová zložitosť funkcie $LOKALIZUJ(p_i)$ bude $\mathcal{O}(\log n)$. Teraz vieme polohu p_i vzhľadom k $CH_{\text{i-1}}$. Vkladáme p_i do $CH_{\text{i-1}}$. Znovu máme štyri možnosti:

A. p_i leží VNÚTRI alebo NA $CH_{\text{i-1}}$, žiadna zmena
B. p_i leží MIMO a zároveň POD T_D, vlož p_i do T_D
C. p_i leží MIMO a zároveň NAD T_H, vlož p_i do T_H
D. p_i leží MIMO a zároveň VĽAVO alebo VPRAVO od $CH_{\text{i-1}}$, vlož p_i do T_H aj do T_D

Pri vkladaní p_i do T_H (pre T_D sa vkladanie urobí analogicky) potrebujeme nájsť oporné vrcholy. Popíšeme iba nájdenie ľavého oporného vrchola, pretože pravý nájdeme analogicky. Začneme prehľadávať T_H



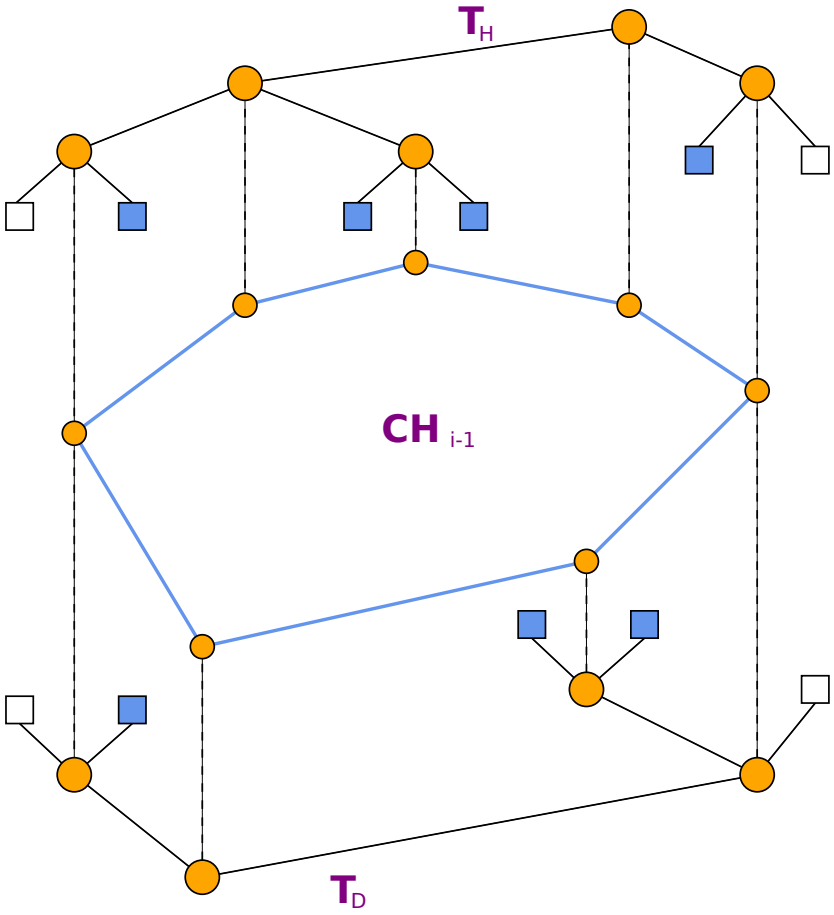
Obr. 3.40. Spájateľná fronta (M - koreň spájateľnej fronty, m - najmenší uzol, teda vrchol s najmenšou x-vou súradnicou).

od koreňa. Nech v je náš súčasný uzol. Ak je v oporný, zastavíme a vrátime v. Ak je v reflexný vzhľadom na p_i, pokračujeme v hľadaní v ľavej vetve. Ak je konkávny, pokračujeme vľavo alebo vpravo, podľa toho, či p_i je vľavo alebo vpravo od v.

Našli sme jeden prípadne dva oporné vrcholy. Teraz môžeme upraviť T_H konštantným počtom operácií ROZDEĽ a SPOJ. Obidve tieto operácie vieme v spájateľnej fronte realizovať v čase $\mathcal{O}(\log n)$. Napríklad po nájdení oboch oporných vrcholov rozdelíme T_H na dve časti v ľavom opornom vrchole. Následne rozdelíme časť, ktorá obsahuje pravý oporný vrchol na dve časti. Časť medzi oboma opornými vrcholmi odstránime a zvyšné dve časti spojíme. Následne vložíme p_i na príslušné miesto.

Analýza časovej zložitosti on line inkrementálneho algoritmu

Nech má náš konvexný obal n vrcholov. Určenie polohy bodu v procedúre ($LOKALIZUJ(p_i)$) nám trvá v najhoršom prípade $\mathcal{O}(\log n)$. Pri hľadaní oporných vrcholov môžeme prejsť najviac $\log n$ vrcholov (od

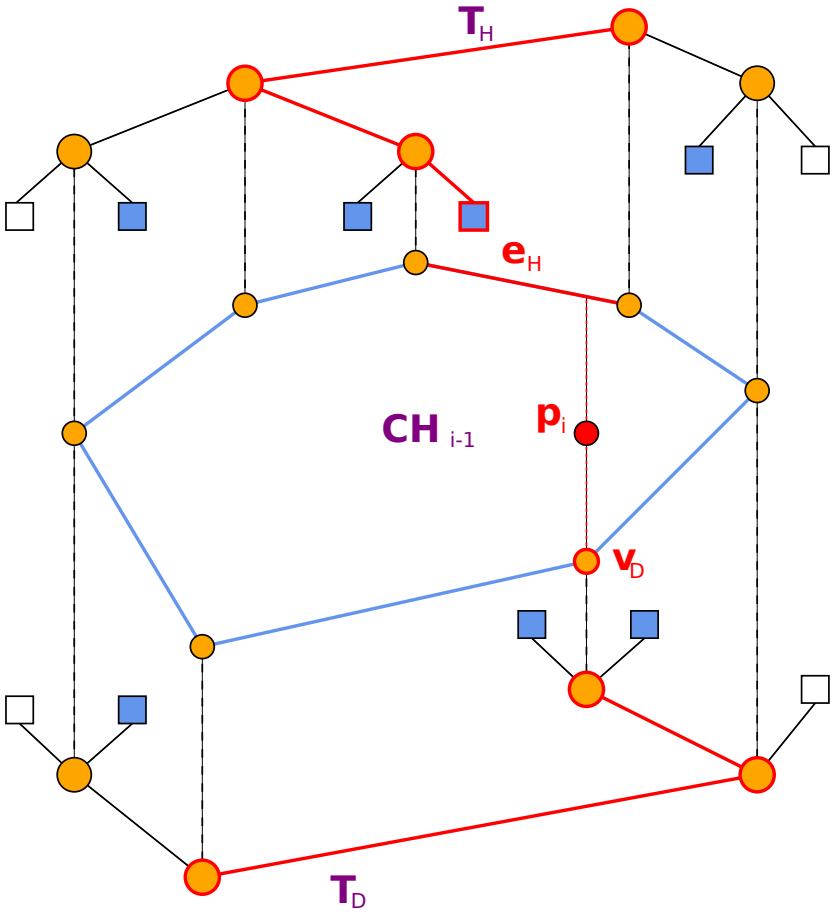


OBR. 3.41. Horný a dolný konvexný obal reprezentované ako spájateľná fronta (kvôli prehľadnosti sme vynechali smerníky, ktoré boli na obrázku 3.40 zobrazené zelenou farbou).

koreňa až po list). Určiť typ vrchola (oporný, reflexný alebo konkávny) vieme realizovať v čase $\mathcal{O}(1)$. Takže nájdenie oporných vrcholov trvá $\mathcal{O}(\log n)$. Odstránenie vrcholov medzi opornými vrcholmi trvá vďaka použitiu spájateľnej fronty tiež $\mathcal{O}(\log n)$. Vloženie p_i trvá $\mathcal{O}(\log n)$, takže asymptotická časová zložitosť inkrementálneho on line algoritmu je $\mathcal{O}(n \log n)$. □

CVIČENIE 3.9 (15 bodov). *Nech S je množina obsahujúca n úsečiek v rovine. Dokážte, že konvexný obal S je totožný s konvexným obalom $2n$ koncových bodov úsečiek z množiny S.*

Riešenie (Elena Dušková, 13.10.2008): Úsečka je konvexný jednorozmerný útvar. Podľa jednej z definícií konvexného mnohouholníka platí, že ak útvar obsahuje 2 body, tak obsahuje aj úsečku nimi určenú. Z n úsečiek máme $2n$ koncových bodov, ktoré keď patria konvexnému



OBR. 3.42. Určenie polohy p_i vzhľadom k $CH_{\text{i-1}}$ pomocou spájateľnej fronty.

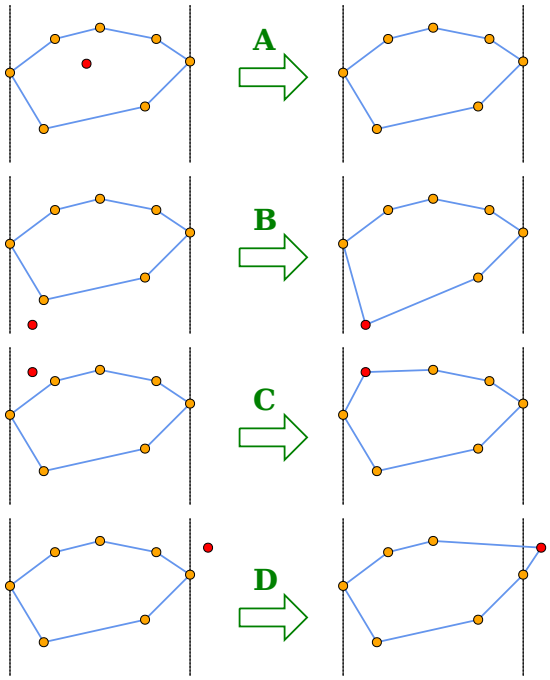
mnohouholníku, tak tam patria aj dané úsečky. Pri tvorbe konvexného obalu sú pre nás podstatné koncové body úsečky a nie úsečka ako taká (pozri obr. 3.44).

Sú 3 možnosti polohy úsečky vzhľadom na konvexný obal.

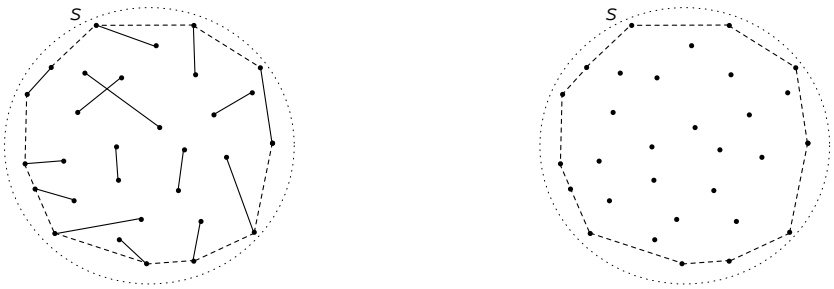
- Úsečka je vo vnútri konvexného mnohouholníka, nemá s hranicou žiadne spoločné body,
- jeden z koncových bodov úsečky patrí hranici konvexného mnohouholníka,
- celá úsečka leží na hranici (buď tvorí hranu, alebo je jej podmnožinou).

□

Riešenie (Lukáš Apalovič, 14.10.2008): Označme M množinu koncových bodov úsečiek z množiny S. Chceme ukázať, že $\text{conv}(M) = \text{conv}(S)$.



OBR. 3.43. Vloženie p_i do CH_{i-1}.



OBR. 3.44. Vľavo je znázornená množina úsečiek S s jej konvexným obalom. Vpravo množina koncových bodov úsečiek z S spolu s ich konvexným obalom.

(1) Najskôr dokážeme, že $\text{conv}(M) \subseteq \text{conv}(S)$. Keďže $M \subseteq S$, tak $\text{conv}(M) \subseteq \text{conv}(S)$.

(2) $\text{conv}(S) \subseteq \text{conv}(M)$. Dokážeme to sporom. Ak $\text{conv}(S) \nsubseteq \text{conv}(M)$, tak $S \nsubseteq \text{conv}(M)$.

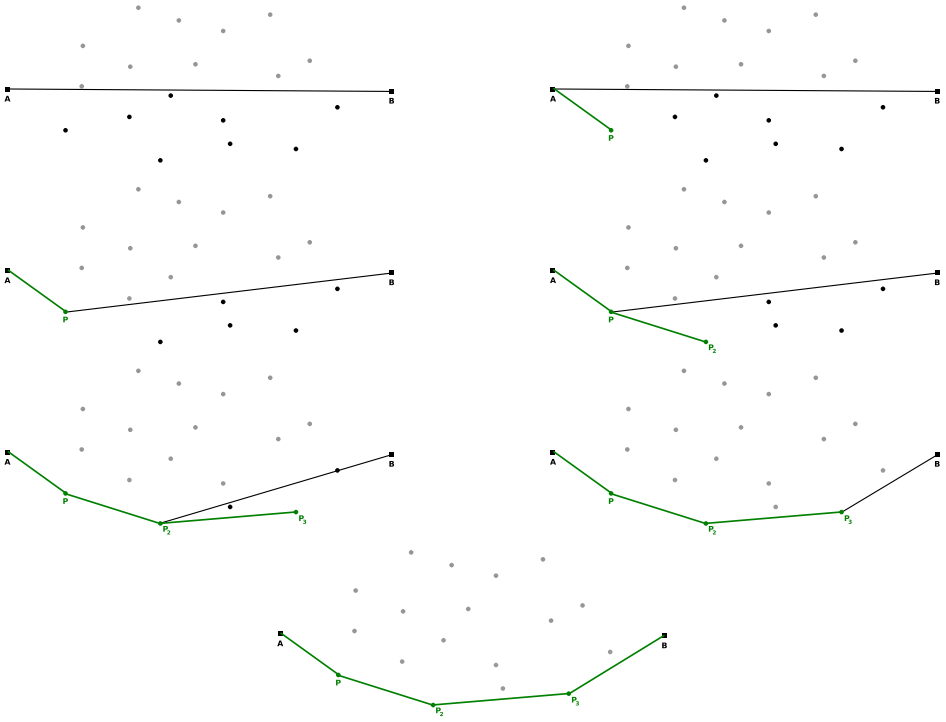
Potom pre nejakú úsečku $AB \in S$ platí: $A, B \in \text{conv}(M)$ (podľa definície $\text{conv}(M) \wedge AB \nsubseteq \text{conv}(M)$), čo je spor s predpokladom, že $\text{conv}(M)$ je konvexná množina.

Keďže platia obe inklúzie, platí aj rovnosť $\text{conv}(M) = \text{conv}(S)$. □

Cvičenie 3.10 (30 bodov). *Nech S je množina bodov a a, b sú dva body z doplnku S také, že a leží vľavo od S a b leží vpravo od S. Zostavte algoritmus, ktorý nájde najkratšiu cestu z a do b idúcu popod S.*

Riešenie (Dušan Pácal, 13.10.2008): Predpokladajme, že máme množinu bodov S v rovine a okrem nej ešte dva body - **a** a **b**, ktoré do nej nepatria. Bod **a** je naľavo od množiny a bod **b** je od nej napravo. Znamená to, že jeho x-ová súradnica je menšia (resp. väčšia) ako všetky ostatné x-ové súradnice bodov z množiny S.
Nech $A = (a_x, a_y), B = (b_x, b_y)$
Určíme rovnicu priamky, ktorá prechádza oboma bodmi.
Bude mať tvar: $ax + by + c = 0$ pričom:
$a = (b_y - a_y)$,
$b = -(b_x - a_x)$
$c = -a.a_x - b.a_y$
Keďže smerový vektor našej priamky je vektor $\mathbf{b} - \mathbf{a}$ a hľadáme body, ktoré sú pod priamkou, budú to body napravo od polpriamky $\overrightarrow{\mathbf{ab}}$, teda tie body, pre ktoré po dosadení do rovnice priamky dostaneme kladný výsledok.
Cestu budeme uchovávať v zozname (označme ho C), do ktorého najprv vložíme počiatočný bod - bod **a**.

(1) Do zoznamu (označme ho Z) si uložíme všetky body z S, ktoré sú pod, alebo na priamke, t.j. body, pre ktoré: $ax + by + c \geq 0$.
(2) Postupne prechádzame bodmi v zozname Z, aktuálny si označíme **p**. Hľadáme bod, pre ktorý bude uhol **bap** najväčší. Nájdený vrchol zaradíme do zoznamu C, ktorý bude určovať cestu.
(3) Rovnakým spôsobom ako na začiatku nájdeme rovnicu priamky, ktorá prechádza bodmi **p** a **b**.
(4) Určíme body, ktoré sú nad priamkou, dosadením do rovnice. Tieto body ďalej nebudeme potrebovať a môžeme ich vymazať zo zoznamu Z. Takisto z neho môžeme vymazať aj bod **p**, ktorý už na patrí ceste a je zbytočné ho znovu testovať.
(5) Opäť budeme hľadať bod zo zoznamu Z, označme ho $\mathbf{p}_2$, pre ktorý bude uhol $\mathbf{bpp}_2$ najväčší. Nájdený bod zaradíme do zoznamu C.
(6) Takto pokračujeme, pokiaľ nebude zoznam Z prázdny.
(7) Do zoznamu C pridáme bod **b**.



Obr. 3.45. Hľadanie spojnice "popod"množinu S.

V zozname C sa teraz nachádza cesta z bodu **a** do bodu **b**, ktorá je celá pod množinou S. Takto sme našli časť konvexného obalu množiny S (pozri obr. 3.45). □

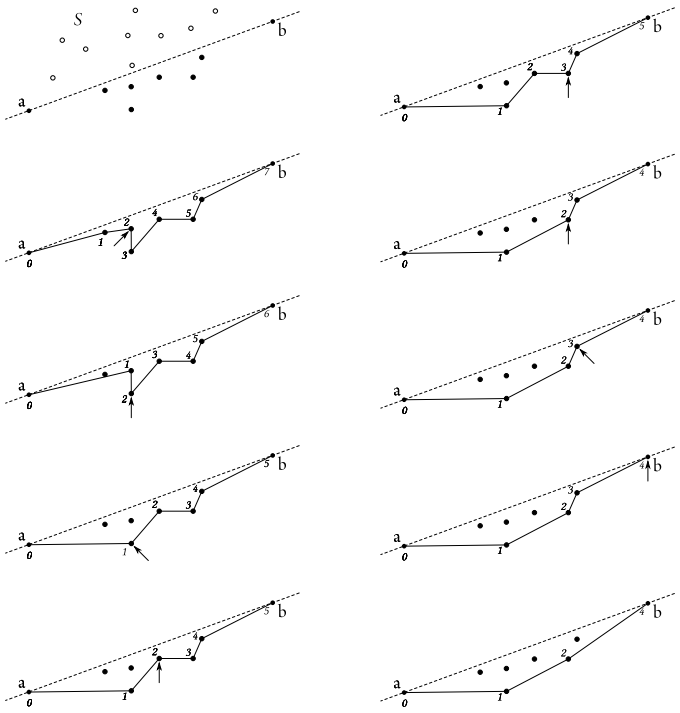
Riešenie (Martin Havala, 27.10.2009): Zrejme do riešenia budú patriť iba body, ktoré ležia pod priamkou $\overline{ab}$. V prvom kroku si teda vytvoríme podmnožinu S_1, v ktorej nebudú vrcholy nad priamkou $\overline{ab}$. Tento krok nám tak v priemernom prípade pomôže urýchliť výpočet cesty.

Algoritmus:

Na začiatku máme zoznam bodov $x_i \in S$. Označme si ho S.

(1) L // prázdny zoznam pre riešenie
(2) `do` L `vlož vrchol` a
(3) `pre všetky` $x_i \in S$ {
(4) `if` $(\angle_{or} bax_i \leq 0)$ {
(5) `do` L `vlož vrchol` x_i
(6) }}
(7) `lexikograficky usporiadaj` L `podľa súradníc` x, y
(8) `do` L `vlož vrchol` b
(9) `i=2`
(10) `while i < size(`L`)` {
(11) `while(i`≥ 2 `&&(` $\angle L_{i-2}L_{i-1}L_i$ `je konkávny` $\| L_i.x == L_{i-1}.x$ `)){`

(12) z L vyber vrchol L_{i-1}
(13) i−−
(14) }
(15) i++
(16) }
(17) return L



Obr. 3.46. Hľadanie dolnej cesty

Zložitosť tohoto algoritmu je $\mathcal{O}(n \log n)$, keďže v princípe ide o jednoduché triedenie a následne jeden prechod zoznamom. □

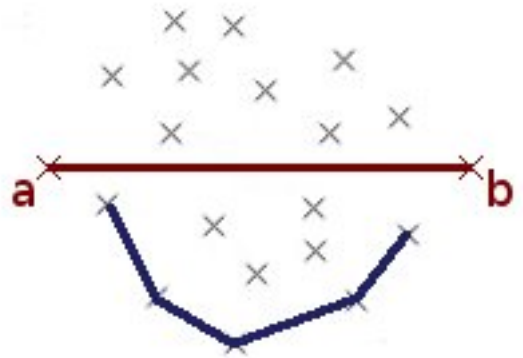
Riešenie (Viktória Bakurová, 5.11.2009): Pri hľadaní najkratšej cesty z bodu a do bodu b využijeme vlastnosti konvexného obalu zostrojeného na danej množine bodov.

Zostrojíme priamku aba zistíme, kde ležia body množiny S vzhľadom na túto priamku. Nech priamka abmá rovnicu $a_0x + b_0y + c_0 = 0$. Môžu nastať tieto prípady:

a) Všetky body množiny Sležia nad priamkou ab, to je po dosadení do rovnice priamky spĺňajú nerovnosť $a_0x + b_0y + c_0 < 0$. Najkratšia cesta z $a$do b vznikne priamym spojením bodov a a b.

b) Existujú body z množiny S,ktoré ležia pod priamkou ab, to sú body, ktoré po dosadení do rovnice priamky spĺňajú nerovnosť $a_0x + b_0y + c_0 \geq 0$. Označme si túto množinu bodov S_0.

Najkratšiu vzdialenosť z a do bhľadáme nasledovne:



Obr. 3.47. Dolný konvexný obal množiny S_0

1) Nájdeme dolný konvexný obal množiny S_0. Označme si tieto vrcholy tohoto obalu postupne $q_0, \ldots, q_n$.

2) Nájdeme opornú priamku k tomuto konvexnému obalu prechádzajúcu cez bod a. Takáto oporná priamka je práve jedna a pretne konvexný obal buď v nejakom vrchole obalu, nech je to vrchol q_i, alebo v hrane obalu, nech je to hrana $q_{i-1}q_i$.

3) Rovnako nájdeme opornú priamku ku konvexnému obalu prechádzajúcu cez bod b. Takáto oporná priamka je práve jedna a pretne konvexný obal buď v nejakom vrchole obalu, nech je to vrchol q_j, alebo v hrane obalu, nech je to hrana $q_{j-1}q_j$. Platí, že $i \leq j$.

4) Najkrašiu cestu z $a$do b získame spojením bodov $a, q_i, \ldots q_j, b$.

Algoritmus:

1) zostroj priamku ab s rovnicou $a_0x + b_0y + c_0 = 0$.

2) pre každý bod z S: ak $a_0x + b_0y + c_0 \geq 0$, zaraď bod do množiny S_0

3) ak $S_0 = \emptyset$, najkratšia cesta je úsečka ab

inak

a) nájdi opornú priamku k obalu prechádzajúcu bodom a a jej prienik s obalom, t.j. buď vrchol q_i alebo hranu $q_{i-1}q_i$

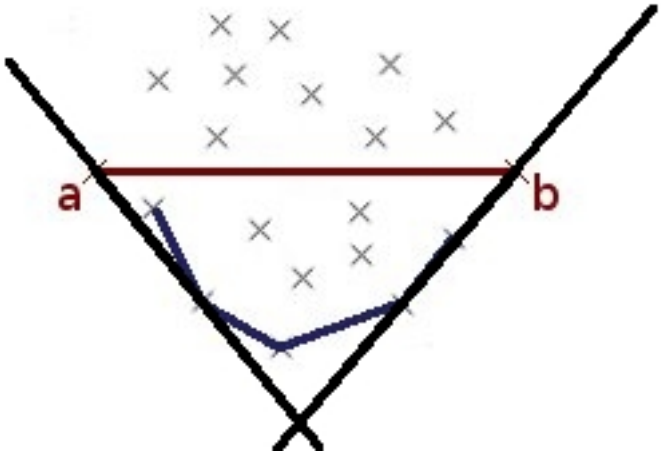
b) nájdi opornú priamku k obalu prechádzajúcu bodom b a jej prienik s obalom, t.j. buď vrchol q_j alebo hranu $q_{j-1}q_j$

4) spojením bodov $a, q_i, \ldots q_j, b$ máme najkratšiu cestu z a do $b$vedúcu popod S

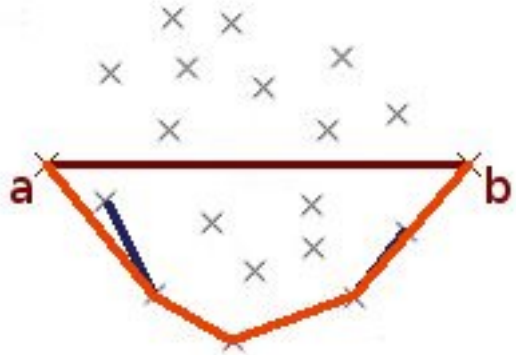
časová náročnosť:

1) Rozhodnutie pre každý bod z S, či patrí do S_0: $\mathcal{O}(n)$

2) Vytovrenie konvexného obalu: $\mathcal{O}(n \log n)$



Obr. 3.48. Oporné priamky ku konvexnému obalu prechádzajúce cez body aa b



Obr. 3.49. Najkratšia cesta z a do b vedúca popod S

3) Nájdenie oporných priamok ku konvexnému obalu: $\mathcal{O}(n)$
4) Spojenie nájdených bodov: $\mathcal{O}(1)$

Celková časová náročnosť: $\mathcal{O}(n \log n)$ □

Riešenie (Júlia Kučerová, 6.11.2010): Majme v rovine množinu bodov S a body a a b, kde a leží naľavo od množiny S a bod b leží napravo od množiny S.

Nech $a = [a_x, a_y]$ a $b = [b_x, b_y]$, potom:

$a_0 = (b_y - a_y)$
$b_0 = -(b_x - a_x)$
$c_0 = -a.a_x - b.a_y$

Pomocou týchto vyjadrení si určíme rovnicu priamky: $a_0x + b_0y + c_0 = 0$. Môžu nastať 3 situácie polohy bodov z množiny S vzhľadom na priamku ab:

(a) bod leží nad priamkou ab: $a_0x + b_0y + c_0 < 0$
(b) bod leží pod priamkou ab: $a_0x + b_0y + c_0 > 0$
(c) bod leží na priamke ab: $a_0x + b_0y + c_0 = 0$

Nás budú zaujímať 2 situácie:

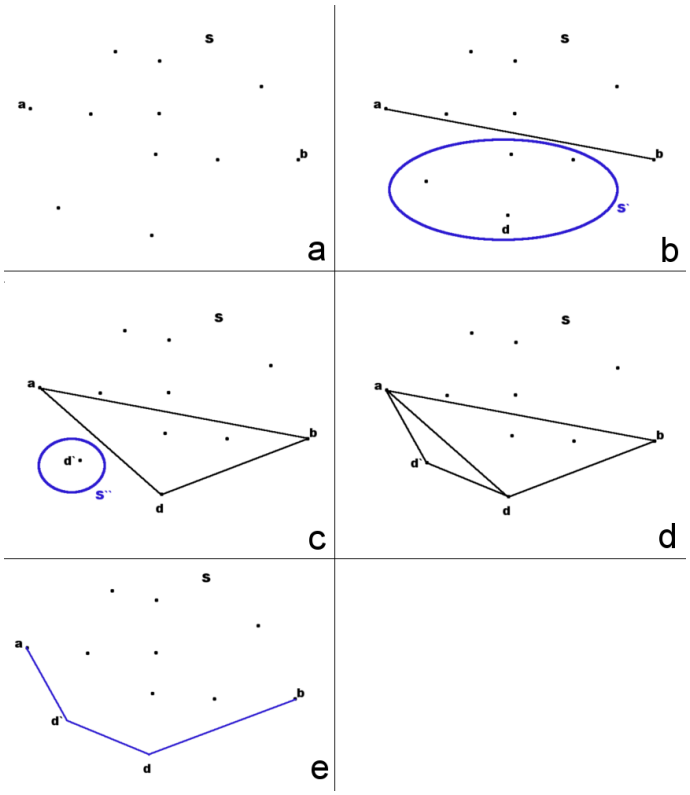
1) Bod leží nad priamkou, alebo na nej.
2) Bod leží pod priamkou.

V prípade, že všetky body z množiny S ležia nad priamkou ab, alebo na nej, vznikne najkratšia cesta priamym spojením bodov a a b.

Ak existujú body z množiny S, ktoré ležia pod priamkou ab zaradíme si ich do novej množiny S'.

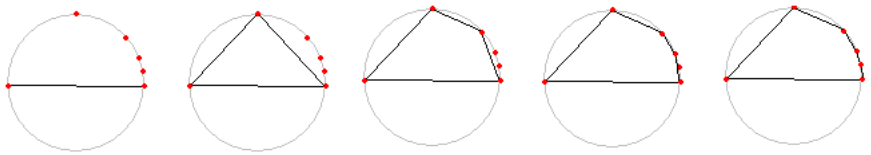
V tomto prípade najkratšiu cestu z bodu a do bodu b vyhľadáme nasledovne:

(1) Vytvoríme si zoznam Z, do ktorého budeme ukladať body cesty. Vložíme doň bod a, ako počiatočný bod cesty.
(2) Majme množinu S' v ktorej si nájdeme bod najvzdialenejší od priamky ab, označme ho d. Môže nastať situácia, kde takýchto bodov bude viac. V tomto prípade si z nich vyberieme bod, ktorý je najbližšie k bodu a.
(3) Body a a b si spojím s vybraným bodom d. Tým nám vzniknú dve nové priamky ad a db.
(4) Preskúmame polohu bodov z množiny S' vzhľadom na nové priamky. Zoberieme si priamku ad. Ak všetky body z množiny S' ležia nad danou priamkou, alebo na nej, môžeme bod d vložiť do zoznamu Z a pokračovať s priamkou db. V opačnom prípade si vytvoríme nový zoznam bodov S'', do ktorého si vložíme všetky body ležiace pod priamkou ad. Rekurzívne opakujeme pokiaľ nenastane situácia, kedy všetky body ležia nad priamkou.



Príklad:
(obrázok a): Máme zadanú množinu S a body a a b. Do zozmanu Z si vložíme bod a.
(obrázok b): Vytvoríme si množinu S' a nájdeme si bod d, ktorý je od priamky ab najvzdialenejší.
(obrázok c): Vytvoríme si priamky ad a db. Pokračujeme na priamke ad. Množina S'' nie je prázdna, čiže bod d nemôžme zatiaľ zaradiť do zoznamu Z.
(obrázok d): V množine S'' si nájdeme bod d' najvzdialenejší od ad. Vytvoríme si priamky ad', $d'd$. Preskúmame polohu bodov z množiny S'' vzhľadom na priamku ad'. Všetky body ležia na, alebo nad priamkou ad'. Do zoznamu Z môžme vložiť bod d' a následne d, keďže pod priamkou $d'd$ neleží žiaden bod. Pokračujeme s priamkou db. Žiaden bod z množiny S' neleží pod priamkou db, čo znamená, že do množiny môžeme vložiť bod b.
(obrázok e): Zostrojíme cestu z bodu a do bodu b postupným vykresľovaním bodov zoznamu Z.
Pseudokód:

```
nájdenie cesty (a, b, S)
    (1) pre každý bod patriaci množine S zisti, či patrí
        do množiny S‘
    (2) Ak je S’ prázdna do zoznamu Z pridaj bod b.
        inak
```



Obr. 3.50. Najhorší prípad konvexného obalu pre algoritmus QuickHull.

```
(a) z množiny S' vyber bod d
(b) nájdenie cesty(a, d, S')
(c) nájdenie cesty(d, b, S')

Algoritmus:
    (1) do zoznamu Z pridaj bod a
    (2) nájdenie cesty(a, b, S)
    (3) vykresli cestu spájaním bodov zo zoznamu Z
```

Analýza zložitosti: V najhoršom možnom prípade bude zložitosť $\mathcal{O}(n^2)$. □

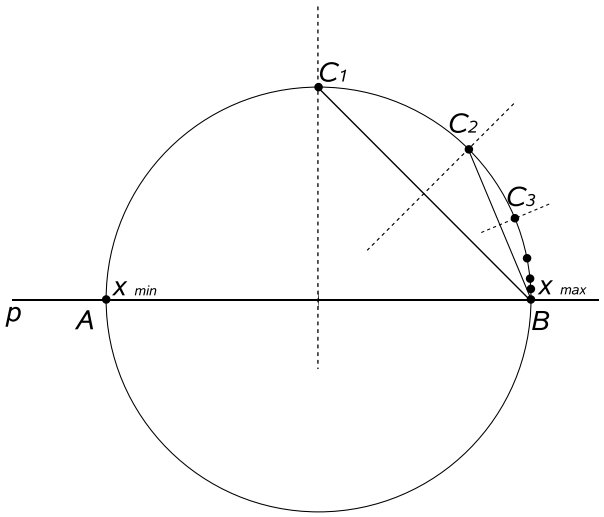
Cvičenie 3.11 (15 bodov). *Nájdite množinu bodov v rovine, na ktorej algoritmus QuickHull dosiahne hornú hranicu svojej časovej zložitosti.*

Riešenie (Lukáš Apalovič, 14.10.2008): Algoritmus QuickHull dosiahne hornú hranicu časovej zložitosti, ak v každej iterácii musí prehľadávať všetky zvyšné body množiny (pozri obr. 3.50). □

Riešenie (Elena Dušková, 19.10.2008): Algoritmus QuickHull funguje nasledovne: V prvom kroku vyhľadáme extremálne body ($X_{\min}$ a $X_{\max}$), cez ktoré vedieme priamku p. Potom nájdeme najvzdialenejší bod B od priamky p (idúcej extremálnymi bodmi) a spojíme ho úsečkami s $X_{\min}$ a $X_{\max}$. Zostavíme množiny bodov P_1, P_2 vytvorené polrovinami $X_{\min B}$ a $X_{\max B}$, ktoré neobsahujú úsečku $X_{\min}X_{\max}$ a algoritmus rekurzívne aplikujeme na tieto množiny.

QuickHull má výhody aj nevýhody QuickSortu prenesené do dvoch rozmerov. Pre väčšinu vstupov dosahuje lepší čas, najmä vtedy, keď veľká časť bodov sa nachádza vo vnútri konvexného obalu a nie na jeho hranici. V najhoršom prípade je časová zložitosť tohto algoritmu $\mathcal{O}(n^2)$. Teda v každom kroku sa algoritmus „zbaví“ len jedného bodu (v prvom kroku dvoch). Algoritmus beží n-krát (resp. $(n-1)$-krát) a pre každý beh treba zo zvyšných bodov nájsť najvzdialenejší od priamky. Odtiaľ $\mathcal{O}(n^2)$.

Rozložíme vrcholy na kružnicu, teda určite všetky budú patriť len hranici konvexného obalu. Prvé dva vrcholy, cez ktoré vedieme priamku



OBR. 3.51. Najhorší prípad konvexného obalu pre algoritmus QUICKHULL.

p, budú ležať na priemere kružnice rovnobežnom s osou x (vrcholy A, B, pozri obr. 3.51). Ďalší bod C_1 (najvzdialenejší od priamky p) bude rovnako vzdialený od $X_{\min}$ a $X_{\max}$ a bude ležať na kružnici. Bod C_1 a jeden extremálny body ($X_{\max}$) z predchádzajúceho kroku spojíme úsečkou. Opäť nájdeme bod na kružnici rovnako vzdialený od bodov B a C_1. Také body sú 2, vyberieme si ten, ktorý leží v polrovine ABC_1. Ďalej postupujeme rekurzívne až po rozmiestnenie všetkých n bodov. Vždy sa vnárame len do jednej („pravej“) časti kružnice po rozdelení algoritmom na 2 podmnožiny. Body C_i sa limitne blížia k bodu B. □

Riešenie (Pavol Beluško, 31.10.2008): Najhorší prípad nastáva, keď sa v každom rekurzívnom vnorení body rozdelia na jeden a zvyšné. Kardinality množín zvyšných bodov tvoria po krokoch lineárne klesajúcu postupnosť prirodzených čísel:
1. vnorenie: spracuje sa $n-1$ bodov
2. vnorenie: spracuje sa $n-2$ bodov
3. vnorenie: spracuje sa $n-3$ bodov
..
$(n-1)$. vnorenie: spracuje sa 1 bod

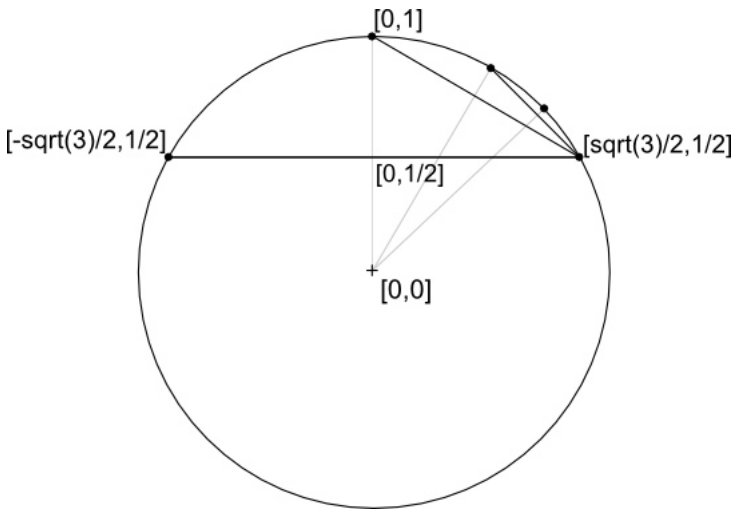
Súčet krokov je teda $(n-1)+(n-2)+\cdots+1$, čiže podľa vzorca na súčet prvých n prirodzených čísel $\frac{(n-1)n}{2} = \frac{n^2-n}{2}$. V polynóme figuruje monóm stupňa 2, čiže algoritmus potrebuje $\mathcal{O}(n^2)$ krokov, čo zodpovedá najhoršiemu prípadu.

Zostáva dokázať, že takáto množina bodov existuje. QUICKHULL hľadá extremálne body podľa x-ovej súradnice (x_{min}, x_{max}, pridáme ich do konvexného obalu), ich spojnicou body rozdelí a rozdelí tak problém

na hľadanie horného a dolného konvexného obalu. V "hornej"časti identifikujeme bod, ktorý je od spojnice x_{min} a x_{max} najvzdialenejší. Tento pridáme do konvexného obalu, spojíme s bodmi x_{min} a x_{max}. Obe spojnice opäť rozdelia body. Novovzniknuté oblasti sú 2 alebo 3, v každom prípade len dve obsahujú kandidátov na príslušnosť k hornému konvexnému obalu (tretia oblasť, ak existuje, obsahuje body, ktoré sú vo vnútri konvexného obalu, preto ju neprehľadávame). Na týchto dvoch oblastiach algoritmus pokračuje rekurzívne ďalej. Obdobné platí pre dolný konvexný obal.

Naším cieľom je rozdeliť spomínaným delením body na oblasti tak, aby sme body rozdelili na jeden a všetky ostatné (podmienka a), pričom aj úvodné delenie pomocou spojnice x_{min} a x_{max} musí zaradiť všetky body do jednej oblasti (podmienka b). Takto usporiadané body možno generovať napr. pomocou nasledujúcej procedúry Generuj. **Pseudokód:**

```
KolekciaBodov SkodneBody;

void RekurzivneGeneruj(Bod a, Bod b, int n)
{
  if (n == 0)
    Return;

  Priamka ab = new Priamka(a, b);
  //Vygeneruj "najvzdialenejší", tj. deliaci bod.
  a = Bod.Nula + ab.Normala / ab.Normala.Length;
  SkodneBody.Pridaj(a);

  //ponechaním bodu b docielime, že sa budú všetky dalšie body
  //generovať len v jednej z dvoch kandidujúcich oblastí
  //Týmto rekurentne platným pravidlom splníme podmienku (a)
  RekurzivneGeneruj(a, b, n-1);
}

//Procedúra vytvorí množinu n bodov, na ktorých
//dosiahne QuickHull najhorší čas behu
void Generuj(n)
{
  if (n < 1)
    Return;

  if (n == 1)
  {
    SkodneBody.Pridaj(Bod.Nula);
    Return;
  }
```



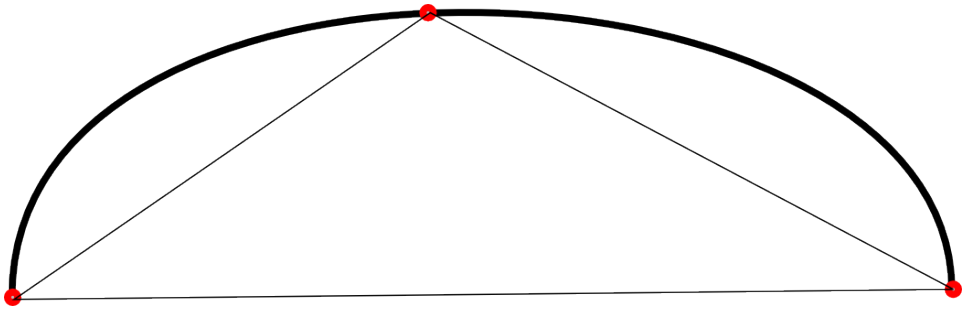
Obr. 3.52. Body vygenerované procedúrou Generuj pre n = 5.

```
  //Vytvor body x_min a x_max. Ďalšie budú generované nad nimi
  //Tým splníme podmienku (b)
  a = new Bod(-1, 0.5);
  b = new Bod( 1, 0.5);

  SkodneBody.Pridaj(a);
  SkodneBody.Pridaj(b);

  RekurzivneGeneruj(a, b, n-2);
}
```

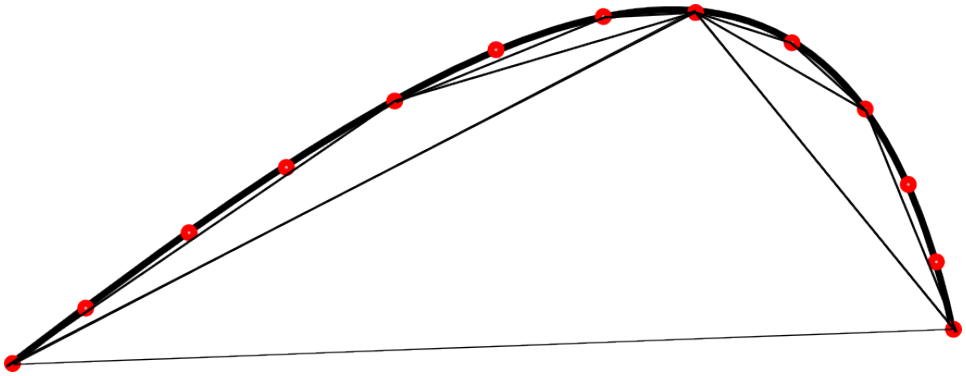
Pozn.: Táto procedúra generuje body na jednotkovej kružnici. □

Riešenie (Jakub Mišún, 18.11.2011): Podstatou algoritmu `QuickHull` je vyraďovanie bodov, zo zoznamu potenciálnych kandidátov na bod konvexného obalu. Vyraďujeme body ktoré sa nachádzajú v aktuálnom trojuholníku, preto ak chceme dosiahnuť hornú hranicu zložitosti, musíme pracovať s takou množinou bodov, z ktorej pri žiadnom kroku algoritmu nenájdeme bod v trojuholníku. Takouto množinou ju napríklad kružnica, ako ukázali riešenia mojich kolegov. Avšak kružnica nie je jediným príkladom. Hornú hranicu zložitosti dosahujú body rozložené po elipse, časti sínusoidy, hyperboly a mnohých iných. Vo všeobecnosti sa dá povedať, že akákoľvek množina bodov, ktorá je rozložená po konkávnej alebo konvexnej krivke, ktorá nemá inflexný bod, dosahujú túto zložitosť (samozrejme krivka musí byť spojitá). Práve konkávnosť (konvexnosť) nám zaisťuje potrebné vlastnosti množiny. Vďaka nej trojuholník pretína krivku v práve troch bodoch.

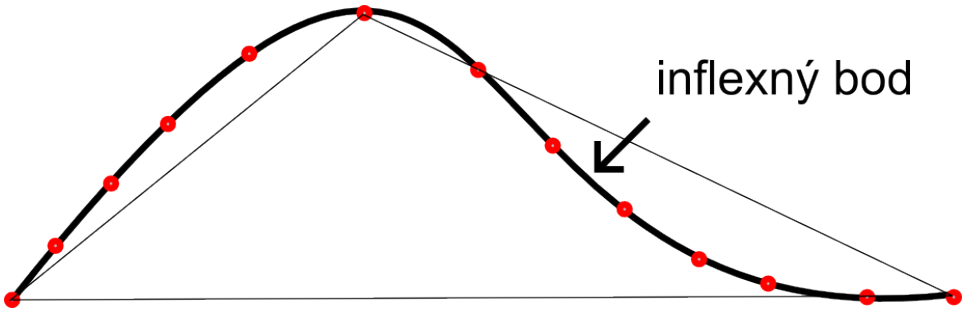
□



Obr. 3.53. Konkávnosť zabezpečuje, že žiaden bod nebude v trojuholníku



Obr. 3.54. Algoritmus musí prejsť všetky body



Obr. 3.55. Existencia inflexného bodu nám nevyhovuje, pretože vylučuje niektoré body množiny

Cvičenie 3.12 (30 bodov). *Zostavte algoritmus konštrukcie konvexného obalu monotónneho mnohouholníka (vzhľadom na vertikálnu os), ktorého zložitosť je lineárna.*

Riešenie (Pavol Beluško, 3.11.2008): Mnohouholník M sa nazýva monotónny vzhľadom na nejakú priamku p, ak každá priamka kolmá na p pretne mnohouholník M najviac dvakrát.

Hľadajme horný konvexný obal (problém je symetrický). Nájdeme bod s maximálnou ($X_{\max}$) a minimálnou ($X_{\min}$) x-ovou súradnicou. Z

monotónnosti mnohouholníka vyplýva, že ak budeme po hornej hranici postupovať od bodu $X_{\max}$ k $X_{\min}$, každý ďalší bod bude mať menšiu hodnotu súradnice x. Teda postupnosť bodov na hornej hranici určuje usporiadanie, ktoré možno chápať ako usporiadanie podľa polárnych súradníc, keď stred sústavy polárnych súradníc je v nevlastnom bode osi y. Z toho vyplýva, že na ne možno použiť Grahamov prechod a získať tak horný konvexný obal. Pri dolnom konvexnom obale postupujeme analogicky.

Analýza časovej zložitosti:

(1) Hľadanie bodov $X_{\min}$ a $X_{\max}$ – $\mathcal{O}(n)$
(2) Grahamov prechod po hranici mnohouholníka – $\mathcal{O}(n)$

Celková časová zložitosť algoritmu je teda $\mathcal{O}(n)$. □

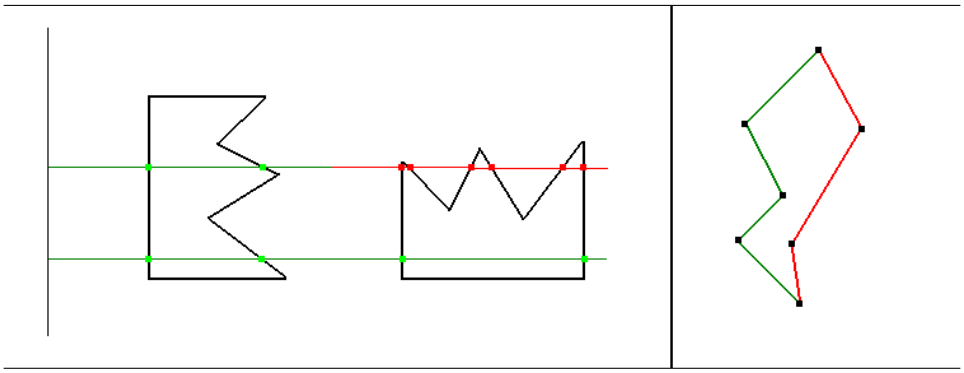
Riešenie (Jana Hlinková, 31.10.2008): Mnohouholník P je monotónny vzhľadom na vertikálnu os, ak jeho hranicu δP vieme rozdeliť na dve lomené čiary, ktoré sú monotónne vzhľadom na vertikálnu os. Lomená čiara je monotónna vzhľadom na vertikálnu os, ak jej prienikom s ľubovoľnou horizontálnou priamkou je maximálne jeden súvislý komponent (prázdna množina, jeden bod alebo úsečka). V poňatí kartézskych súradníc, monotónna lomená čiara sa nikdy nevráti na y-ovú súradnicu, ktorou už raz prešla a ktorú už opustila. Z toho vyplýva, že monotónna lomená čiara je aj jednoduchá. Ak by nebola jednoduchá, musela by sama seba niekde pretínať. Aby sa pretínala, musela by sa znovu vrátiť na niektorú y-ovú súradnicu, čo ale nemôže, keďže je monotónna.

Monotónny mnohouholník teda vieme rozdeliť na dve jednoduché lomené čiary. Môžeme postupovať takto: nájdeme si bod s najmenšou y-ovou súradnicou (najnižší bod) a bod s najväčšou y-ovou súradnicou (najvyšší bod). Z najnižšieho bodu postupne traverzujeme mnohouholníkom, až kým neprídeme do najvyššieho bodu. Pretraverzovaná časť tvorí jednu lomenú čiaru, zvyšná časť druhú (obrázok 3.56).

Pre jednoduchý mnohouholník vieme v čase $\mathcal{O}(n)$ zostrojiť konvexný obal (viď prednáška). Teda vieme v čase $\mathcal{O}(n)$ zostrojiť aj konvexný obal jednoduchej lomenej čiary. Stačí teda nájsť konvexné obaly našich dvoch jednoduchých monotónnych lomených čiar a spojiť ich.

Analýza časovej zložitosti:

- nájdenie najnižšieho a najvyššieho vrcholu vieme urobiť na jeden prechod poľom: $\mathcal{O}(n)$
- rozdelenie hranice mnohouholníka na dve monotónne lomené čiary je v prípade reprezentácie mnohouholníka v "cyklickom"poli `P`, kde `P[i-1]` a `P[i+1]` sú susedné vrcholy hranice vrholu `P[i]` veľmi jednoduché, pole sa rozdelí na dve časti pri najnižšom a najvyššom vrchole: $\mathcal{O}(1)$



OBR. 3.56. Monotónny mnohouholník a nemonotónny mnohouholník, vzhľadom na vertikálnu os. Rozdelenie monotónneho mnohouholníka na dve monotónne lomené čiary, s vyznačeným najnižším a najvyšším bodom.

- nájdenie konvexných obalov jednoduchých monotónnych lomených čiar, ako bolo vyššie spomenuté, vieme realizovať v lineárnom čase: $\mathcal{O}(n)$
- spojenie dvoch konvexných obalov: $\mathcal{O}(n)$

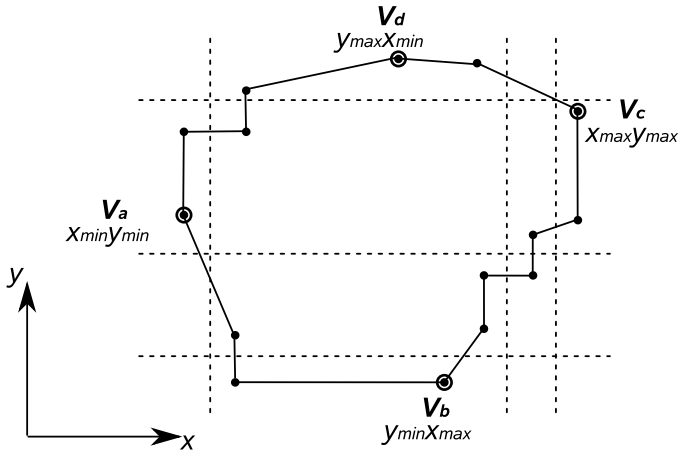
□

CVIČENIE 3.13 (40 bodov). *Mnohouholník sa nazýva ortogonálne konvexný, ak prienik ľubovolnej horizontálnej(vertikálnej) priamky je najviac jeden súvislý komponent. Charakterizujte ortogonálne konvexné mnohouholníky a pokúste sa zostrojiť algoritmus na otestovanie daného mnohouholníka.*

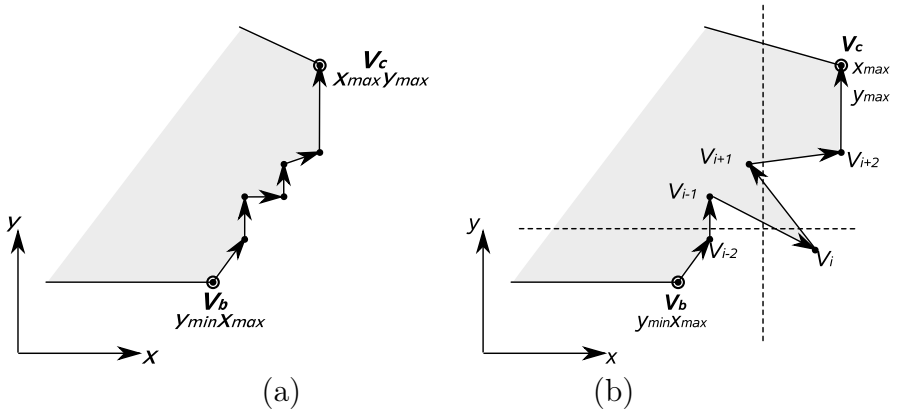
Riešenie (Elena Dušková, 19.10.2008): Predpokladajme reprezentáciu, pri ktorej si pamätáme poradie vrcholov mnohouholníka, prvý vrchol je nasledovníkom posledného (cyklickosť) a vieme túto postupnosť jednoducho rozdeliť na viacej častí. Nájdeme 4 význačné vrcholy (pozri obr. 3.57).

V_a**:** vrchol s minimálnou x-ovou súradnicou a z vrcholov spĺňajúcich túto podmienku vrchol s minimálnou y-ovou súradnicou
V_b**:** vrchol s minimálnou y-ovou súradnicou a z vrcholov spĺňajúcich túto podmienku vrchol s maximálnou x-ovou súradnicou
V_c**:** vrchol s maximálnou x-ovou súradnicou a z vrcholov spĺňajúcich túto podmienku vrchol s maximálnou y-ovou súradnicou
V_d**:** vrchol s maximálnou y-ovou súradnicou a z vrcholov spĺňajúcich túto podmienku vrchol s minimálnou x-ovou súradnicou

Takto sa nám obvod mnohouholníka rozdelí na štyri postupnosti vrcholov, pričom vrcholy V_a, V_b, V_c, V_d budú patriť vždy do 2 postupností



Obr. 3.57. Príklad ortogonálne konvexného mnohouholníka.



Obr. 3.58. Postupnosť V_b až V_c (a) spĺňajúca a (b) nespĺňajúca podmienku ortogonálnej konvexnosti.

(raz ako začiatočný, potom ako koncový bod). Ak má byť mnohouholník ortogonálne konvexný, tak musia tieto postupnosti spĺňať určité podmienky.

Keď si zoberieme postupnosť V_b až V_c (pozri obr.), tak sa musíme pohybovať len smermi „doprava až hore“, teda súradnice v smere x neklesajú ($x_{i+1} \geq x_i$) a taktiež súradnice v smere y ($y_{i+1} \geq y_i$). Ak by sme sa medzi vrcholmi V_i a V_{i+1} posunuli iným smerom, už to nebude ortogonálne konvexný mnohouholník (pozri obr.). Obdobné pravidlá platia aj pre ostatné sekvencie.

Aby to bol ortogonálne konvexný mnohouholník musí platiť:

medzi V_a **a** V_b**:** $x_{i+1} \geq x_i$ a $y_{i+1} \leq y_i$
medzi V_b **a** V_c**:** $x_{i+1} \geq x_i$ a $y_{i+1} \geq y_i$
medzi V_c **a** V_d**:** $x_{i+1} \leq x_i$ a $y_{i+1} \geq y_i$
medzi V_d **a** V_a**:** $x_{i+1} \leq x_i$ a $y_{i+1} \leq y_i$

Algoritmus na otestovanie ortogonálnej konvexnosti:

```
For (int i = 1; i < n; i++) Begin
  //pre všetkých n vrcholov v úseku  Vb-Vc
  //Vb = V1, Vc = Vn
  If ((Vi+1.x >= Vi.x) and (Vi+1.y >= Vi.y))
       Ortogon_konvex = ,,TRUE‘‘
  Else Begin
       Ortogon_konvex = ,,FALSE‘‘
       Exit
  End
End
```

Obdobne (podľa podmienok uvedených vyššie) sa dajú algoritmizovať aj ostatné 3 podmnožiny vrcholov. □

Riešenie (Lukáš Tencer, 30.10.2008): Aby sme mohli korektne charakterizovať ortogonálne konvexné mnohouholníky, uvedieme si ešte niektoré pojmy.

Monotónny mnohouholník: Polygonálna reťaz C je striktne monotónna vzhľadom k priamke L ak ľubovoľná priamka ortogonálna ku L pretína C najviac v jednom bode. Reťaz C je monotónna vzhľadom k priamke L, pokiaľ ľubovoľná priamka ortogonálna ku L pretína C v jednom súvislom komponente. A teda nemusí ju pretínať vôbec, alebo ju pretína v jednom bode, alebo pozdĺž jednej úsečky. Polygón P je monotónny vzhľadom k úsečke P, ak jeho hranica $d(P)$ môže byť rozdelená na dve reťaze, pričom každá z nich je monotónna ku L.

Použijúc predchádzajúce definície, môžme teraz charakterizovať ortogonálne konvexné mnohouholníky ako mnohouholníky, ktoré sú monotónne ku vertinkálnej (y-ovej) a horizontálnej (x-ovej) osi.

Algoritmus na testovanie bude spočívať v overení, či je možné mnohouholník rozložiť na dve reťaze monotónne k vertikálnej osi a dve reťaze monotónne k horizontálnej osi. Najprv overujme monotónnosť ku horizontálnej osi. Nájdeme bod p_{xmin} s minimálnou x-ovou súradnicou a bod p_{xmax} s maximálnou x-ovou súradnicou. Teraz prechádzame dvomi vetvami od bodu p_{xmin} ku bodu p_{xmax}. Pokiaľ máme uložený polygón v dvojsmerne spájanom zozname, tak jedna z vetiev pôjde po $p_{xmin}.predchodca$ a bude sa ďalej vnárať do predchodcov, kým nedosiahne p_{xmax} a druhá obdobne akurát pôjde po $p_{xmin}.nasledovnik$ a bude sa vnárať do nasledovníkov. Pri každom postupe k ďalšiemu vrcholu v reťazi p_{i+1} musí byť splnené pravidlo, že $p_{i+1}.x \geq p_i.x$. Pokiaľ splnené nie je, prehlásime, že mnohouholník nie je ortogonálne konvexný.

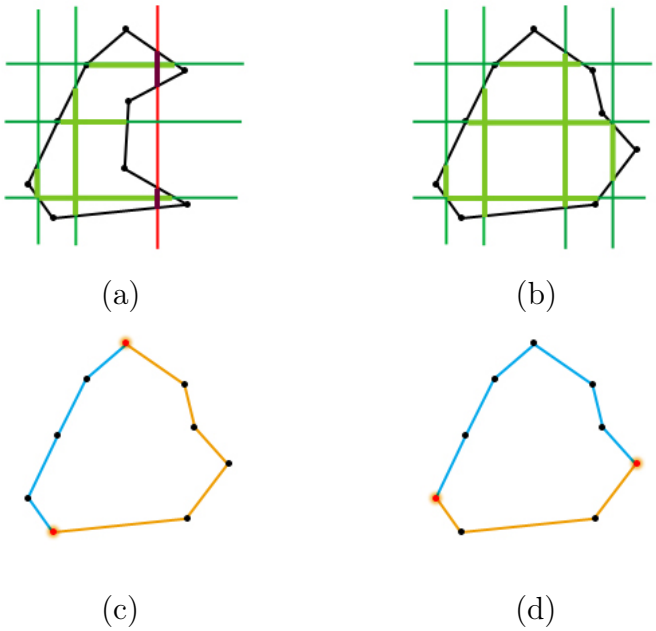
Ekvivalentný krok urobíme i na overenie monotónnosti ku vertikálnej osi, avšak budeme používať body s minimálnou a maximálnou y-ovou súradnicou a taktiež podmienka sa zmení na: $p_{i+1}.y \geq p_i.y$.

Pokiaľ by sme pri vyberaní min/max bodu našli viaceré s rovnakou súradnicou vybereme jeden ľubovoľný z nich.

```
boolean jeOrtoKonvex(poly[]) {
  pmin;qmax;r
  najdiXMinMax(poly[], &pmin, &qmax) {
  r=pmin;
  while(r!=qmax){
    if(r.nasledovnik.x < r.x){
      return FALSE;
    }else{
      r=r.nasledovnik;
    }
  }

  r=pmin;
  while(r!=qmax){
    if(r.predchodca.x < r.x){
      return FALSE;
    }else{
      r=r.predchodca;
    }
  }

  najdiYMinMax(poly[], &pmin, &qmax) {
  r=pmin;
  while(r!=qmax){
    if(r.nasledovnik.y < r.y){
      return FALSE;
    }else{
      r=r.nasledovnik;
    }
  }

  r=pmin;
  while(r!=qmax){
    if(r.predchodca.y < r.y){
      return FALSE;
    }else{
      r=r.predchodca;
    }
  }
}
```

□

CVIČENIE 3.14 (50 bodov). *Nech P je axiálny mnohouholník(t. j. jeho strany sú rovnobežné s osami). Ortogonálny konvexný obal mnohouholníka P tvorí najmenší(v zmysle inklúzie) ortogonálne konvexný mnohouholník obsahujúci P. Zostavte algoritmus na výpočet ortogonálneho konvexného obalu daného axiálneho mnohouholníka.*



(a) (b)

(c) (d)

OBR. 3.59. (a) Mnohouholník ktorý nie je ortogonálne konvexný. (b) Mnohouholník Q ktorý je ortogonálne konvexný. (c) Monotónne reťaze Q vzhľadom na vertikálnu os. (d) Monotónne reťaze Q vzhľadom na horizontálnu os.

Riešenie (Martina Bátorová, 26.10.2008): Uvažovaný mnohouholník je axiálny, teda hrany sú buď horizontálne alebo vertikálne. Množinu vrcholov V utrieďme podľa súradnice x a vrámci bodov s rovnakou súradnicou x podľa súradnice y.
Spomedzi V teraz vyberme nasledujúce význačné body najprv podľa súradnice x a potom z bodov s rovnakou súradnicou x podľa súradnice y:

(1) Bod $v_0 = (x, y)$ najviac vľavo dole: $x = x_{min}$, $y = y_{min}$
(2) Bod $v_1 = (x, y)$ najviac vľavo hore: $x = x_{min}$, $y = y_{max}$
(3) Bod $v_2 = (x, y)$ najviac vpravo hore: $x = x_{max}$, $y = y_{max}$
(4) Bod $v_3 = (x, y)$ najviac vpravo dole: $x = x_{max}$, $y = y_{min}$

Týmito vrcholmi môžeme množinu V rozdeliť na štyri množiny - orientované lomené čiary (orientované v smere hodinových ručičiek):

(1) $\Pi_0 = \{v_0, \ldots, v_1\} : (v_0.x \leq v_1.x), (v_0.y \leq v_1.y)$
(2) $\Pi_1 = \{v_1, \ldots, v_2\} : (v_1.x \leq v_2.x), (v_1.y \geq v_2.y)$
(3) $\Pi_2 = \{v_2, \ldots, v_3\} : (v_2.x \geq v_3.x), (v_2.y \geq v_3.y)$
(4) $\Pi_3 = \{v_3, \ldots, v_4\} : (v_3.x \geq v_4.x), (v_3.y \leq v_4.y)$

Ortogonálny konvexný obal $OCH(P)$ zostavíme zo štyroch orientovaných lomených čiar $\Phi_0, \ldots, \Phi_3$, ktoré budeme konštruovať z čiar $\Pi_0, \ldots, \Pi_3$.

Budujme napr. Φ_0 nasledovne:

(1) **Neformálne povedané:**

(a) Do konštruovanej čiary Φ_0 vložíme vrchol v_0. Tento je naším východzím vrcholom, označme ho v.

(b) Z bodov lomenej čiary Π_0 vyberieme dva body p_1 a p_2. Bod p_1 je nasledovníkom vrchola v v Π_0. Druhý vrchol p_2 vyberieme tak, že je k v najbližší v smere nahor-doprava, teda bod kde $(p_2.x \geq v.x) \wedge (p_2.y \geq v.y) \wedge (p_2.x, p_2.y$ sú minimálne možné). Predchádzajúca séria podmienok sa aplikuje nasledovne: najprv vyberieme body, ktoré spĺňajú prvú podmienku. Z množiny týchto bodov vyberieme len tie, ktoré spĺňajú druhú podmienku, z tejto množiny body ktoré spĺňajú tretiu podmienku a napokon pomocou poslednej podmienky identifikujeme hľadaný bod p_2.
Môže sa stať, že $p_1 = p_2$.

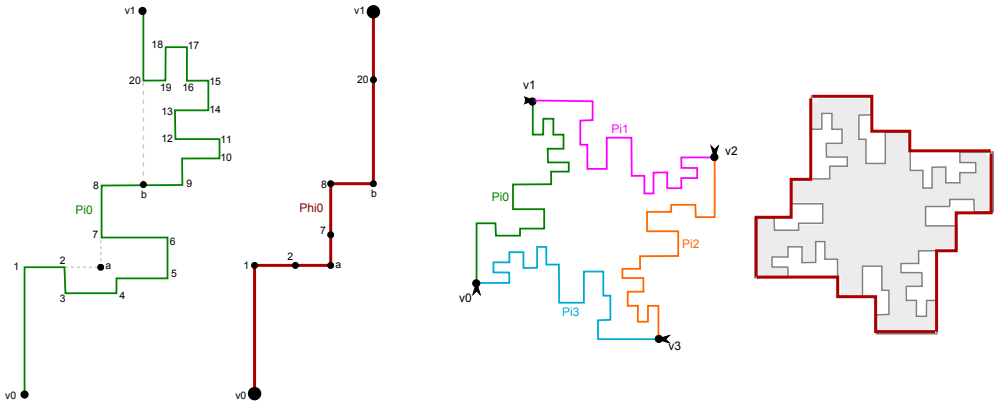
(c) Zo súradníc vrcholov p_1, p_2, v skonštruujeme bod $p_3 = (x, y)$ tak, že sa v smere x neposúvame naľavo a v smere y nadol:
$x = max\{min\{p_1.x, v.x\}, p_2.x\}, y = min\{p_2.y, v.y\}$.

(d) Do Φ_0 vložíme vrchol p_3 (je možné, že je zhodný s jedným z vrcholov p_1, p_2). Ak p_3 nepatril do Π_0, do Φ_0 vložíme aj p_2.
V ďalšom kroku naposledy vložený vrchol považujeme za východzí vrchol v.

(e) Kroky (b) - (d) opakujeme, kým sa nedostaneme do vrchola v_1.

(2) **Algoritmus v pseudokóde:**

```
procedure BudujPhi0(LomenaCiara Pi0){
  LomenaCiara Phi0;
  Phi0[0] := Pi0[0];
  Bod v   := Pi0[0];

  while (v  != Pi0[Pi0.max]){
    Bod p1 := v.next; // v mnozine Pi0
    Bod p2 := Pi0[j]:
      Pi0[j].x - v.x >= 0 min. mozne;
      Pi0[j].y - v.y >= 0 min. mozne v ramci bodov
                          so suradnicou Pi0[j].x;
    Bod p3:
      p3.x := max( min(p1.x, v.x), p2.x );
      p3.y := min( p2.y, v.y );
    Phi0.add(p3);
    if ( (p3 != p1) && (p3 != p2) )
      Phi0.add(p2);
```



OBR. 3.60. Obrázky vľavo popisujú vytváranie $\Phi_0 = \{v_0, 1, 2, a, 7, 8, b, 20, v_1\}$ z $\Pi_0 = \{v_0, 1, \ldots, 20, v_1\}$. Body a, b sú body pridané počas tvorby Φ_0 také, že nepatria pôvodnej množine vrcholov. Na obrázkoch vpravo je pôvodný mnohouholník s vyznačenými orientovanými lomenými čiarami $\Pi_0, \ldots, \Pi_3$ a výsledný $OCH(P)$.

```
        v := Phi0[Phi0.max];
      }
      return Phi0;
    }
```

Veľmi podobným spôsobom by sme zostavili aj ostatné množiny Φ_1, Φ_2, Φ_3 - jediný rozdiel by bol pri podmienkach (aplikovaných vyššie popísaným spôsobom), ktoré by museli spĺňať body p_2 a p_3:

(1) Φ_1 :
  (a) $p_2 : (p_2.x \geq v.x) \wedge (p_2.y \leq v.y) \wedge (p_2.x$ min., $p_2.y$ max.)
  (b) p_3 : v smere x sa neposúvame naľavo a v smere y nahor:
  $x = min\{max\{p_1.x, v.x\}, p_2.x\}, y = min\{p_2.y, v.y\}$.

(2) Φ_2 :
  (a) $p_2 : (p_2.x \leq v.x) \wedge (p_2.y \leq v.y) \wedge (p_2.x$ max., $p_2.y$ min.)
  (b) p_3 : v smere x sa neposúvame napravo a v smere y nahor:
  $x = min\{max\{p_1.x, v.x\}, p_2.x\}, y = max\{p_2.y, v.y\}$.

(3) Φ_3 :
  (a) $p_2 : (p_2.x \leq v.x) \wedge (p_2.y \geq v.y) \wedge (p_2.x$ max., $p_2.y$ min.)
  (b) p_3 : v smere x sa neposúvame napravo a v smere y nadol:
  $y = x = max\{min\{p_1.x, v.x\}, p_2.x\}, max\{p_2.y, v.y\}$.

□

Riešenie (Lenka Mezeiová, 17.11.2009): Množinu všetkých vrcholov axiálneho mnohouholníka označíme $\mathcal{B} = \{P_1, \ldots, P_n\}$. Z $\mathcal{B}$ vyberieme body s najmenšími y súradnicami, množinu týchto bodov označme

$\mathcal{T}_1 = \{P_1', \ldots, P_k'\}$. Označme $\mathcal{B}_1 = \mathcal{B} - \mathcal{T}_1$, teda $\mathcal{B}_1$ obsahuje všetky vrcholy mnohouholníka, ktorých y súradnica je väčšia ako y súradnica bodov z $\mathcal{T}_1$.

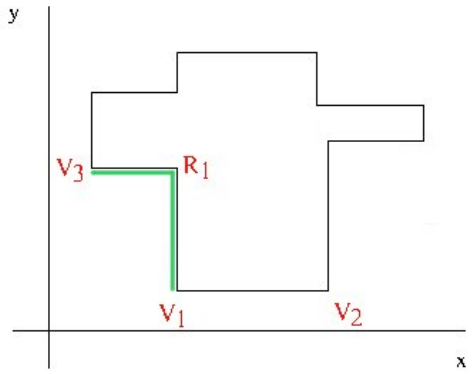
Z $\mathcal{T}_1$ vyberieme bod s najväčšou a najmenšou x súradnicou; bod s najmenšou x súradnicou označme $V_1 = (a, u)$, bod s najväčšou x súradnicou označme $V_2 = (b, u)$.

Body V_1, V_2 spojíme, vzniknutá hrana V_1V_2 je prvou hranou hľadaného konvexného obalu mnohouholníka.

V množine $\mathcal{B}_1$ zostávajúcich vrcholov mnohouholníka opäť nájdeme tie, ktoré majú minimálnu y súradnicu. Množinu nájdených bodov označíme $\mathcal{T}_2$ a zároveň označme $\mathcal{B}_2 = \mathcal{B}_1 - \mathcal{T}_2$. V $\mathcal{T}_2$ nájdeme body s najmenšou a najväčšou možnou x súradnicou. Označme $V_3 = (c, v)$ bod s minimálnou x súradnicou a $V_4 = (d, v)$ bod s maximálnou možnou x súradnicou. Určíme body R_1 a R_2; nech $R_1 = (a, v)$ a $R_2 = (b, v)$ (t.j. x súradnica bodu R_1 sa zhoduje s x súradnicou bodu V_1 z predchádzajúceho kroku, y súradnica je zhodná s y súradnicou bodov z $\mathcal{T}_2$. Analogicky sme určili súradnice bodu R_2).

Testujeme vzájomnú polohu bodov $R_1 = (a, v)$ a $V_3 = (c, v)$:

1. Ak $a > c$ pridáme hrany V_1R_1 a R_1V_3. [Obr. 1].

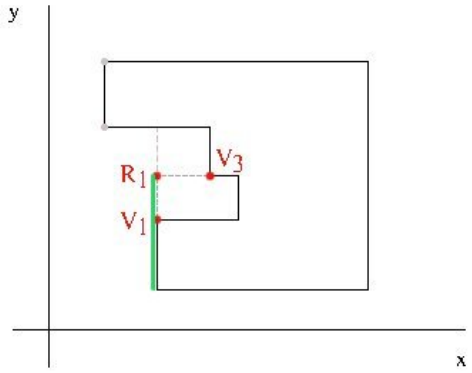


Obr. 3.61. Pridanie hrán V_1R_1 a R_1V_3 v prípade, ak x súradnica bodu R_1 je väčšia ako x súradnica bodu V_3.

2. Ak $a < c$ v množine $\mathcal{B}_2$ vyhľadáme body, ktoré majú x súradnicu menšiu alebo rovnú ako je a. Ak takýto bod nájdeme pridáme hranu V_1R_1 [Obr. 2].

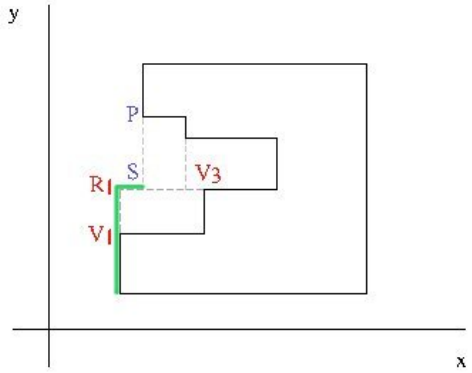
V prípade, že sa takýto bod v zozname $\mathcal{B}_2$ ostávajúcich vrcholov mnohouholníka nenachádza budeme v $\mathcal{B}_2$ hľadať body, ktoré sa nachádzajú medzi bodmi R_1 a V_3 vzhľadom na x súradnicu (t.j. body, ktorých x súradnica je väčšia ako a a menšia ako c).

Ak sa v $\mathcal{B}_2$ nenachádza ani jeden vrchol s uvedenou vlastnosťou, pridáme hrany V_1R_1 a R_1V_3. Ak takéto body nájdeme, vyberieme z nich ten, ktorý je najbližšie k bodu R_1 vzhľadom na x súradnicu, označme



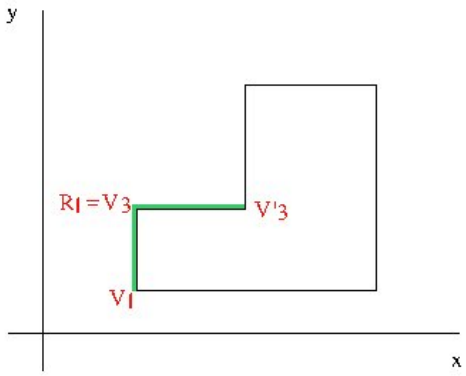
OBR. 3.62. V $\mathcal{B}_2$ sme našli vrcholy, ktorých x súradnica je menšia ako x súradnica bodu R_1, preto pridávame iba hranu V_1R_1.

ho $P = (x, y)$. Určíme bod S, ktorý bude mať súradnice $S = (x, v)$. Pridáme hranu V_1R_1 a R_1S. [Obr. 3].



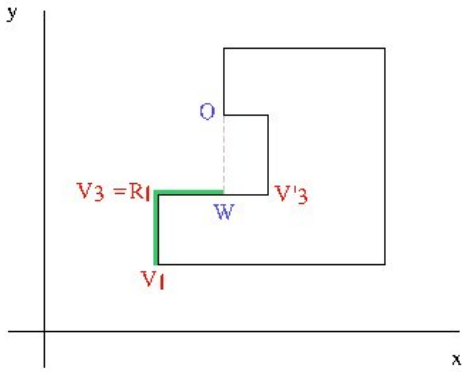
OBR. 3.63. Prípad, keď sa v $\mathcal{B}_2$ nachádzajú body, ktoré (vzhľadom na x súradnicu) ležia medzi bodmi R_1 a V_3.

3. Ak $a = c$ postupujeme podobne ako v 2.: V $\mathcal{B}_2$ hľadáme body s menšou alebo rovnakou x súradnicou ako má bod V_3 ($=R_1$). Ak takýto bod nájdeme, pridáme iba hranu V_1V_3. Ak sa takýto bod v $\mathcal{B}_2$ nevyskytuje, v T_2 nájdeme vrchol, ktorý je najbližšie ku V_3 vzhľadom na x súradnicu, označme ho $V_3' = (q, v)$. V $\mathcal{B}_2$ teraz budeme hľadať vrcholy, ktoré ležia medzi bodmi V_3 a V_3' vzhľadom na súradnicu x. Ak nenájdeme ani jeden bod, ktorý spĺňa túto podmienku, pridáme hrany V_1V_3 a V_3V_3'. [Obr. 4].
Ak takéto body existujú, vyberieme z nich ten, ktorého x súradnica je najbližšie k x súradnici bodu V_3, označme ho $Q = (\overline{x}, \overline{y})$. Určíme bod



Obr. 3.64. Pridanie hrán V_1V_3 a V_3V_3'.

W, ktorý bude mať súradnice $W = (\overline{x}, v)$. Pridáme hrany V_1V_3 a V_3W [Obr. 5].



Obr. 3.65. Pridanie hrán V_1V_3 a V_3W.

Po určení vzájomnej polohy bodov V_1, R_1 a pridaní hrán, pokračujeme v postupe hľadania konvexného obalu analogicky, t.j. opäť vyberieme z $\mathcal{B}_2$ body s minimálnymi y súradnicami, zaradíme ich do T_3 a určíme $\mathcal{B}_3 = \mathcal{B}_2 - T_3$. Novým bodom V_1 (resp. V_2) bude koncový bod poslednej pridávanej hrany.

Analogicky skúmame vzájomnú polohu bodov $R_2 = (b, v)$ a $V_4 = (d, v)$, pričom v tomto prípade budeme vyhľadávať a brať do úvahy vždy body s maximálnymi x súradnicami a budú platiť opačné nerovnosti ako pri predchádzajúcich pozorovaniach.

Algoritmus končí, keď množina $\mathcal{B}_i$ bude obsahovať už len vrcholy s rovnakými y súradnicami. Spojíme dva krajné a tým uzavrieme hľadaný konvexný obal mnohouholníka.

Algoritmus (pseudokód):

1. do B_{MAX} ulož všetky vrcholy mnohouholníka M;
2. nájdi v B_{MAX} body s minimálnou a maximálnou y súradnicou, minimálnu y súradnicu označ u, maximálnu max;
3. body s minimálnou y súradnicou zaraď do poľa T_1;
4. $B_1 =: B_{MAX} - T_1$;
5. z T_1 vyber bod s minimálnou x súradnicou, označ ho $V_0 = (V[0].x, V[0].y)$, kde $V[0].x = a$, $V[0].y = u$;
6. z T_1 vyber bod s maximálnou x súradnicou, označ ho $D_0 = (D[0].x, D[0].y)$, kde $D[0].x = b$, $D[0].y = u$;
7. pridaj hranu V_0D_0;
//ľavá strana:
8. for (int $i = 1$; $i < max$; $i++$){
    9. nájdi v B_i body s minimálnou y súradnicou, označ ju y_i;
    10. nájdené body zaraď do T_{i+1};
    11. $B_{i+1} =: B_i - T_{i+1}$;
    12. v T_{i+1} nájdi bod s minimálnou x súradnicou, označ ho $V_i = (V[i].x, V[i].y)$;
    13. $R =: (R.x, R.y)$, kde $R.x = V[i-1].x$, $R.y = y_i$;
    14. if($R.x > V[i].x$){
    15. pridaj hrany $V[i-1]R$ a $RV[i]$;
    }
    16. if($R.x <= V[i].x$){
    17. v B_{i+1} nájdi body s x súradnicou menšou alebo rovnou ako $R.x$;
    20. ak takýto bod nájdeš{
    21. pridaj hranu $V[i-1]R$;
    22. $V[i].x == R.x$;
    }
    23. inak{
    24. v B_{i+1} nájdi bod s x súradnicou, pre ktorú platí $R.x < x < V[i].x$;
    25. ak takéto body nájdeš{
    26. vyber ten, ktorého x súradnica je najmenšia, označ ju X;
    27. $S =: (S.x, S.y)$, kde $S.x = X, S.y = y_i$;
    28. pridaj hrany $V[i-1]R$ a RS;
    29. $V[i].x == S.x$;
    }
    30. inak{ //v tomto prípade platí rovnosť $R.x == V[i].x$
    31. v T_{i+1} nájdi bod s x súradnicou, ktorá je najbližšie ku $V[i].x$,
    označ ho $W = (W.x, W.y)$, kde $W.x = \overline{X}, W.y = y_i$;
    32. v B_{i+1} nájdi bod s x súradnicou, pre ktorú platí:
    $V[i].x < x < W.x$;
    33. ak takéto body nájdeš{

34. vyber ten, ktorého x súradnica je najmenšia, označ ju XX;
35. $Z =: (Z.x, Z.y)$, kde $Z.x = XX, Z.y = y_i$;
36. pridaj hrany $V[i-1]V[i]$ a $V[i]Z$;
37. $V[i].x == Z.x$;
}
38. inak{
39. pridaj hranu $V[i-1]V[i]$, $V[i]W$;
40. $V[i].x == W.x$;
}
}
}
}

// podobný cyklus (so začiatočným bodom $D_0 = (D[0].x, D[0].y)$) zostavíme aj pre pravú stranu, pričom budú platiť opačné nerovnosti a namiesto minimálnych x súradníc budeme hľadať a porovnávať vždy tie maximálne.

//oba cykly budú prebiehať pokým sa dostanú ku B_{MAX-1}, ostane nám pole B_{MAX}, ktoré obsahuje len body s maximálnymi možnými y súradnicami; tieto už stačí len spojiť hranou, čím uzavrieme hľadaný konvexný obal.

Zložitosť algoritmu:

Na utriedenie n bodov podľa súradnice y potrebujeme čas $O(n \log n)$, rovnako dlho nám bude trvať utriediť tieto body podľa súradnice x. Prehľadávanie bodov v utriedených zoznamoch (binárne vyhľadávanie) má časovú zložitosť $O(\log k)$, kde $k < n$. Teda celková časová zložitosť algoritmu bude $T : O(n \log n)$. □

Cvičenie 3.15 (50 bodov). *Predpokladajme, že množina n bodov $P \subseteq \mathbb{E}^2$ je taká, že náhodne vybraná podmnožina veľkosti r obsahuje $\mathcal{O}(r^\alpha)$ vrcholov konvexného obalu pre nejaké $\alpha < 1$. Potom vieme zostrojiť algoritmus konštrukcie konvexného obalu bežiaci v čase $\mathcal{O}(n)$.*

Riešenie (Michal Ferko, 20.10.2011): Analyzujme algoritmus rozdeľuj a panuj pre tvorbu konvexného obalu v $\mathbb{E}^2$. Pre vytvorenie konvexného obalu n prvkov sa vytvoria konvexné obaly pre množiny P_1 a P_2, každá z nich bude mať $\frac{n}{2}$ prvkov. Následne sa v čase $\mathcal{O}(p+q)$ (kde p je počet vrcholov $conv(P_1)$ a q je počet vrcholov $conv(P_2)$) tieto množiny spoja do konvexného obalu P nájdením oporných priamok.

Vzhľadom na to, že tento algoritmus funguje rekurzívne, vďaka jeho štruktúre sa dá jeho zložitosť zapísať ako rekurentný vzťah:

$$T(n) = 2T\left(\frac{n}{2}\right) + \mathcal{O}(n)$$

Pričom prvá časť pravej strany hovorí o tom, že rozdelíme problém na 2 menšie a rátame tento problém pre polovičné podproblémy. Druhá časť pravej strany hovorí, že na každej úrovni algoritmu sa vykoná $\mathcal{O}(n)$ operácii na vytvorenie finálneho riešenia (toto je práve spájanie dvoch konvexných obalov do jedného).

Náš predpoklad hovorí, že náš výsledný konvexný obal nebude mať príliš veľa vrcholov, teda že existuje $\alpha < 1$ také, že počet prvkov konvexného obalu je $\mathcal{O}(r^\alpha)$. To znamená, že v každom kroku spájania dvoch konvexných obalov do jedného vykonáme menej ako $\mathcal{O}(n)$ krokov, lebo pre obe časti P_1 aj P_2 sa vykoná $\mathcal{O}(r^{\alpha_1})$ resp. $\mathcal{O}(r^{\alpha_2})$ operácii, a teda celkovo $\mathcal{O}(r^{\alpha_3})$ kde $\alpha_3 = max(\alpha_1, \alpha_2)$.

Veta 3.1 (Hlavná veta - Master theorem). *Nech $a \geq 1$, $b > 1$ sú konštanty a nech $f(n)$ je funkcia. Nech $T(n)$ je definovaná rekurentne ako nezáporná:*

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

Potom:

1. *Ak $f(n) = \mathcal{O}(n^{\log_b a - \varepsilon})$ pre $\varepsilon > 0$, tak $T(n) = \Theta(n^{\log_b a})$.*
2. *Ak $f(n) = \mathcal{O}(n^{\log_b a})$, tak $T(n) = \Theta(n^{\log_b a} \log n)$.*
3. *Ak $f(n) = \Omega(n^{\log_b a + \varepsilon})$ pre $\varepsilon > 0$ a ak $af(\frac{n}{b}) \leq cf(n)$ pre $c < 1$ a $\forall n > n_0$, tak $T(n) = \Theta(f(n))$.*

Dôkaz: Dôkaz je príliš dlhý a preto ho neuvedieme. Nachádza sa napríklad v knihe:
Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms, Third Edition (3rd ed.)*. The MIT Press.
Sekcia 4.6 - Proof of the master theorem, strany 97 - 106 □

Dosadením $a = 2, b = 2$ do rekurentného vzťahu z Master theorem dostávame:

$$T(n) = 2T\left(\frac{n}{2}\right) + f(n)$$

Ak $f(n) = \mathcal{O}(n)$, tak $T(n) = \Theta(n \log n)$ (podľa bodu 2 v Master theorem). Toto je prípad všeobecného algoritmu. Ak je ale splnený predpoklad zo zadania, tak $f(n) = \mathcal{O}(n^\alpha)$ pre $\alpha < 1$, teda existuje $\varepsilon > 0$ také, že $f(n) = \mathcal{O}(n^{1-\varepsilon})$. Z toho vyplýva (podľa bodu 1 v Master theorem), že $T(n) = \Theta(n)$, teda $T(n) = \mathcal{O}(n)$.

Algoritmus rozdeľuj a panuj pre konštrukciu konvexného obalu v $\mathbb{E}^2$ teda naozaj bude bežať v čase $\mathcal{O}(n)$, ak je splnená podmienka zo zadania. □

Cvičenie 3.16 (20 bodov). *Nech $Q \subset \mathbb{E}^3$ je jednoduchý mnohosten(uzavretá podmnožina $\mathbb{E}^3$ homeomorfná guli, ktorej hranicou je*

zjednotenie konečného počtu mnohouholníkov) s n vrcholmi. Opíšte algoritmus zložitosti $\mathcal{O}(n)$ na testovanie, či daný bod je z vnútra takéhoto mnohostena.

Riešenie (Michal Ferko, 21.10.2011): Keďže Q je homeomorfná guli, tak vieme všetky vrcholy týmto homeomorfizmom zobraziť na guľu. Tieto vrcholy (spolu s hranami a stenami) možno následne pomocou vhodnej voľby "severného pólu" na tejto guli rozbaliť do roviny. Vznikne tak rovinný graf: vrcholy grafu sú vrcholy nášho mnohostena, hrany sú hrany tohto mnohostena a steny sú steny mnohostena. O rovinnom grafe pritom vieme (pomocou Eulerovho vzorca), že $V - E + F = 2$, kde V je počet vrcholov grafu, E počet hrán grafu a F počet stien.

V našom grafe bude najviac hrán ak všetky steny sú trojuholníky. Každá hrana patrí dvom stenám a každá stena má tri hrany. Preto počet stien je $\mathcal{O}(n)$, a po úprave Eulerovho vzorca do tvaru $V + F = E + 2$ dostávame, že hrán je taktiež $\mathcal{O}(n)$. Preto ak algoritmus bude operovať nad zoznamom všetkých hrán alebo stien a pri každej vykoná $\mathcal{O}(1)$ operácii, tak zložitosť celého algoritmu bude $\mathcal{O}(n)$.

Teraz popíšeme takýto algoritmus. Tento algoritmus je analógiou algoritmu, ktorý testuje, či sa bod nachádza v polygóne v $\mathbb{E}^2$ rozšírený do $\mathbb{E}^3$. Nech a je náš bod, pre ktorý chceme zistiť, či sa nachádza v Q. Postup bude nasledovný:

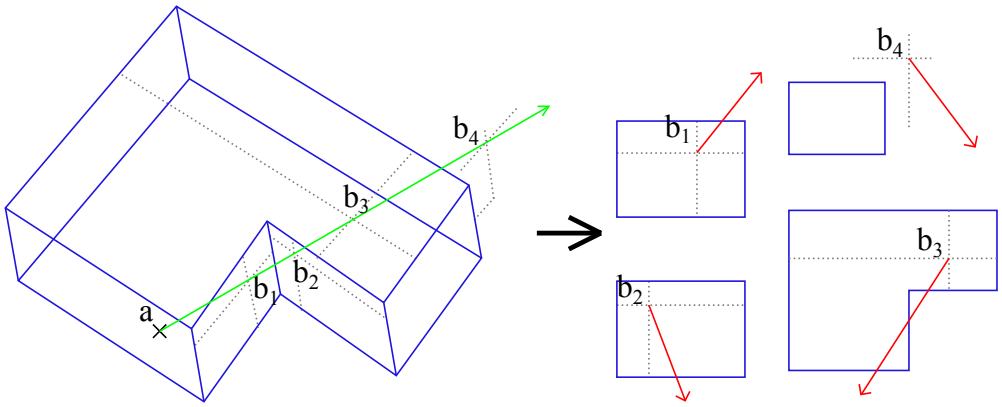
(1) Určíme si náhodný vektor z bodu a, ktorý s bodom a určujú polpriamku. Aby sme sa vyhli prípadom, keď táto polpriamka trafí vrchol, môžeme vygenerovať $\mathcal{O}(1)$ náhodných vektorov a vybrať najčastejší výsledok v kroku 3.
(2) Prejdeme zoznam stien a zistíme, ktoré z týchto stien polpriamka pretína.
(3) Zrátame počet týchto pretnutí a ak ich počet je nepárny, tak bod a je vo vnútri.

Na zistenie, či polpriamka pretína stenu, použijeme práve algoritmus na kontrolu či bod sa nachádza v polygóne v $\mathbb{E}^2$. Najprv skonštruujeme prienik polpriamky a roviny, v ktorej stena leží. Tento bod b následne testujeme rovnakým spôsobom, či je vnútri:

(1) Určíme si náhodný vektor (v $\mathbb{E}^2$) a spolu s bodom b určujú polpriamku.
(2) Zrátame počet prienikov s hranami polygónu. Ak je počet nepárny, bod b je vo vnútri polygónu.

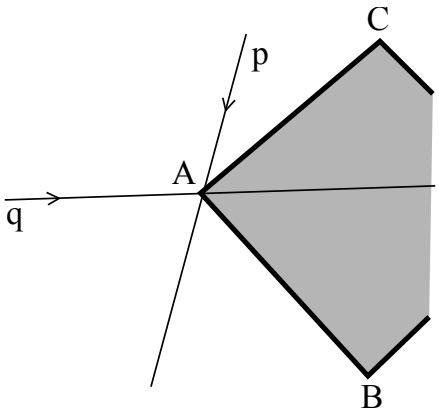
Pre jednoduchšie pochopenie je takáto situácia znázornená na obrázku 3.66.

Treba si uvedomiť, že tak ako v $\mathbb{E}^3$ sme mali problém ak polpriamka trafí bod, tak v $\mathbb{E}^2$ sa môžeme tomuto problému vyhnúť. Vždy, keď polpriamka trafí bod stačí skontrolovať, či susedné body tohto bodu sú v rovnakom polpriestore určenom polpriamkou, alebo ich polpriamka



OBR. 3.66. Príklad nášho algoritmu. Z bodu a sa vystrelí polpriamka (zelená), ktorá pretne 4 roviny prislúchajúce stenám v bodoch b_1, b_2, b_3, b_4. Následne sa testuje či body ležia v stenách a to vystrelením náhodných polpriamok (červených) v rovinách určených stenami. Po zrátaní prienikov v prípade stien zistíme, že b_1, b_2, b_3 sú v stenách, ale b_4 nie. Dokopy máme 3 prieniky stien, a teda bod a je vo vnútri nášho mnohostena.

oddeľuje. Ak ich oddeľuje, zarátame tento prienik ako jeden prienik, inak ako dva. Tieto prípady sú aj zobrazené na obrázku 3.67.



OBR. 3.67. Ošetrenie, keď polpriamka trafí bod mnohouholníka pri teste, či sa v ňom nachádza bod. Polpriamky p aj q pretínajú bod A. Ale pre p sú body B a C na rovnakej strane, teda tento prienik zarátame ako 2 prieniky s hranami mnohouholníka. V prípade polpriamky q zarátame iba jeden prienik.

Dôležité je ukázať, že algoritmus naozaj beží v čase $\mathcal{O}(n)$. Na prvý pohľad krok na testovanie, či bod je v stene, trvá $\mathcal{O}(n)$, lebo počet hrán mnohouholníka je ohraničený $\mathcal{O}(n)$ a test, či polpriamka pretína úsečku

v $\mathbb{E}^2$ je v čase $\mathcal{O}(1)$. Krok iterujúci cez všetky steny mnohostena sa dá analyzovať rovnako, akurát krok na test prieniku polpriamky a steny je v čase $\mathcal{O}(n)$. To by znamenalo, že celkovo bude mať algoritmus zložitosť $\mathcal{O}(n^2)$. Kvôli riadkom (1) a (2) v pseudokóde treba mať rozumnú dátovú štruktúru, pri ktorej vieme získať ľubovoľnú stenu z Q v čase $\mathcal{O}(1)$ a taktiež ľubovoľnú hranu prislúchajúcu stene v čase $\mathcal{O}(1)$.

Ďalšou analýzou nasledujúceho pseudokódu ukážeme, že takýto algoritmus bude naozaj lineárny.

```
a - bod, ktorý testujeme, Q - mnohosten, ktorý testujeme
v = náhodný smer v 3D
i = 0
pre každú stenu S mnohostena Q                          (1)
    L = rovina, v ktorej leží S
    b = prienik polpriamky [a, v> a roviny L
    ak existuje také b
        w = náhodný smer v rovine L
        j = 0
        pre každú hranu H steny S                       (2)
            c = prienik H a polpriamky [b, w>           (3)
            ak existuje také c
                j = j + 1
        ak j je nepárne
            i = i + 1
            ak bod a = bod b
                vráť NA STENE
ak i je nepárne, vráť VNÚTRI
inak vráť VONKU
```

Treba si uvedomiť, že počet hrán je $\mathcal{O}(n)$ a každá hrana má práve dve steny, ktoré jej prislúchajú. Preto test či polpriamka pretína úsečku bude vykonaný presne $2E$ krát. Teda celková zložitosť bude $\mathcal{O}(n)$, lebo riadok (3) sa spustí $\mathcal{O}(n)$ krát. □

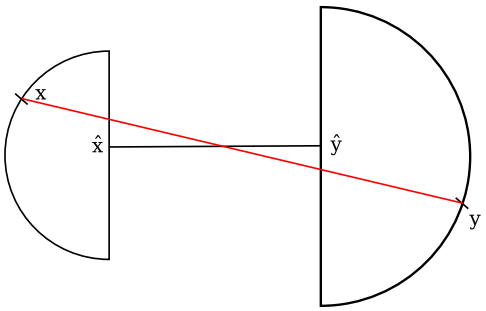
Cvičenie 3.17 (45 bodov). *Nech $P, Q \subset \mathbb{E}^2$ sú jednoduché mnohouholníky také, že $\operatorname{int} P \cap \operatorname{int} Q = \emptyset$, Q je konvexný a $Q \subset \operatorname{conv}(P)$. Nájdite kritérium, či existuje taký spojitý jednoparametrický systém $\phi : [0,1] \to \mathbb{E}^2$ zhodností mnohouholníka Q (pohyb mnohouholníka Q), že*

- $\phi(0) = \mathrm{id}_{\mathbb{E}^2}$,
- $\operatorname{int} \phi(t)(Q) \cap \operatorname{int} P = \emptyset$, $t \in [0,1]$,
- $\operatorname{int} \phi(1)(Q) \cap \operatorname{int} \operatorname{conv}(P) = \emptyset$.

Chceme zistiť, či sa dá konvexný mnohouholník Q „vyvliecť" z mnohouholníka P.

CVIČENIE 3.18 (30 bodov). *Nech $K \subseteq \mathbb{E}^d$ je d-rozmerný konvexný mnohosten a zobrazenie $\phi\colon \mathbb{E}^d \to K$ také, že $\phi(\mathbf{x}) = \arg\min_{\mathbf{y}\in K} \|\mathbf{x} - \mathbf{y}\|$. Ukážte, že funkcia ϕ je dobre definovaná a spojitá.*

Riešenie (Martin Manduch, 22.11.2009): Najprv dokážeme nerovnosť $\|\phi(\mathbf{x}) - \phi(\mathbf{y})\| \leq \|\mathbf{x} - \mathbf{y}\|$, a potom ukážeme, ako z toho vyplýva, že zobrazenie ϕ je dobre definované a spojité.



OBR. 3.68. Ukážka situácie v $\mathbb{E}^2$.

LEMA 3.1. *Pre všetky $\mathbf{x}, \mathbf{y} \in \mathbb{E}^d$ platí: $\|\phi(\mathbf{x}) - \phi(\mathbf{y})\| \leq \|\mathbf{x} - \mathbf{y}\|$.*

Dôkaz: Majme body $\mathbf{x}$, $\mathbf{y}$, $\hat{\mathbf{x}} = \phi(\mathbf{x})$ a $\hat{\mathbf{y}} = \phi(\mathbf{y})$. Označme $r_1 = \|\mathbf{x} - \hat{\mathbf{x}}\|$ a $r_2 = \|\mathbf{y} - \hat{\mathbf{y}}\|$. Nech G_1, G_2 sú gule s polomermi r_1, r_2 a stredmi $\hat{\mathbf{x}}, \hat{\mathbf{y}}$. Bod $\mathbf{x}$ leží na G_1 a v polpriestore L_1 určenom nerovnosťou $(\mathbf{x} - \phi(\mathbf{x}))(\phi(\mathbf{y}) - \phi(\mathbf{x})) \leq 0$. Kebyže $\mathbf{x}$ leží v opačnom otvorenom polpriestore mal by iný obraz ako $\hat{\mathbf{x}}$, lebo celá úsečka $\hat{\mathbf{x}}\hat{\mathbf{y}}$ leží v K. Z takého istého dôvodu leží $\mathbf{y}$ na G_2 a v polpriestore L_2 určenom nerovnosťou $(\mathbf{y} - \phi(\mathbf{y}))(\phi(\mathbf{x}) - \phi(\mathbf{y})) \leq 0$. Vzdialenosť $\mathbf{x}, \mathbf{y}$ je väčšia ako vzdialenosť L_1L_2 ktorá je $\|\phi(\mathbf{x}) - \phi(\mathbf{y})\|$. □

Ukážeme, že ϕ je dobre definované. Nech existujú $\mathbf{u}, \mathbf{v}$ také, že $\phi(\mathbf{x}) = \mathbf{u}$ a tiež $\phi(\mathbf{x}) = \mathbf{v}$. Použitím lemy 3.1 dostaneme: $\|\mathbf{u} - \mathbf{v}\| \leq \|\mathbf{x} - \mathbf{x}\| = 0$. Z toho je zrejmé, že $\mathbf{u} = \mathbf{v}$.

Zostalo nám ešte dokázať spojitosť ϕ. Ak $\|\mathbf{x} - \mathbf{y}\| \leq \epsilon$, tak z lemy 3.1 vyplýva, že aj $\|\phi(\mathbf{x}) - \phi(\mathbf{y})\| \leq \epsilon$. □

Riešenie (Alojz Kováčik, 14.11.2009): Ide o funkciu, ktorá bodu $\mathbf{x}$ priradí taký bod $\mathbf{f}(\mathbf{x}) \in K$, že

$$\|\mathbf{x} - f(\mathbf{x})\| = \operatorname{dist}(\mathbf{x}, K).$$

Nech bod $\mathbf{x} \in \mathbb{E}^d$. Ak $\mathbf{x} \in K$, potom $\mathbf{x}$ je zároveň jedinečným najbližším bodom samého seba.

Nech $\mathbf{x} \notin K$. Keďže K je neprázdna a uzavretá množina, existuje aspoň jeden bod $\mathbf{a} \in K$ taký, že $\phi(\mathbf{x}) = \mathbf{a}$.

Sporom: Nech existujú takéto body dva. Platí teda $\phi(\mathbf{x}) = \mathbf{a_1}$ a $\phi(\mathbf{x}) = \mathbf{a_2}$. Zoberme bod $\mathbf{c} = \frac{1}{2}(\mathbf{a_1} + \mathbf{a_2})$. Dostávame rovnoramenný trojuholník $\triangle(\mathbf{a_1}, \mathbf{x}, \mathbf{a_2})$. Z konvexnosti množiny K vyplýva, že $\mathbf{c} \in K$.

Z rovnoramennosti trojuholníka dostávame $\|\mathbf{x}-\mathbf{c}\| < \|\mathbf{x}-\mathbf{a_1}\|$, čo je však spor s predpokladom, že $\|\mathbf{x}-\mathbf{a_1}\| = \text{dist}(\mathbf{x}, K)$.

Funkcia ϕ každému bodu $\mathbf{x}$ priradí jedinečný bod z množiny K, a teda je dobre definovaná.

Dôkaz spojitosti: Nech $\mathbf{x}, \mathbf{y} \in \mathbb{E}^d$. Označme $\mathbf{a} := \phi(\mathbf{x}), \mathbf{b} := \phi(\mathbf{y})$.

Chceme ukázať:

$$\|\mathbf{b}-\mathbf{a}\| \leq \|\mathbf{x}-\mathbf{y}\| \tag{7}$$

Pre $\mathbf{a} = \mathbf{b}$ je nerovnosť splnená. Uvažujme teda $\mathbf{a} \neq \mathbf{b}$. Označme vektor $\vec{v} = \mathbf{a}-\mathbf{b}$ a nadroviny H_a, H_b kolmé na $\vec{v}$, pričom $\mathbf{a} \in H_a, \mathbf{b} \in H_b$.

Nech je daná nadrovina H_0 kolmá na $(\mathbf{x}-\mathbf{a})$ a zároveň obsahuje bod $\mathbf{a}$. Dokážeme, že bod $\mathbf{x}$ leží v opačnom polpriestore určenom H_0 ako mnohosten K, čo ďalej použijeme pri dôkaze vzťahu (7).

Sporom: Nech ležia v rovnakom polpriestore. Potom existuje taký bod $\mathbf{d} \in K - \{\mathbf{a}\}$, že

$$0 < \angle(\tilde{\mathbf{u}}, \mathbf{d}-\mathbf{a}) < \frac{\pi}{2}, \tilde{\mathbf{u}} = \mathbf{x}-\mathbf{a}. \tag{8}$$

Z toho, že $\|\mathbf{x}-\mathbf{a}\| = \text{dist}(\mathbf{x}, K)$ dostaneme $\|\mathbf{x}-\mathbf{a}\| < \|\mathbf{x}-\mathbf{d}\|$. V každom trojuholníku platí, že oproti väčšej strane leží väčší uhol, a teda

$$\angle(\mathbf{a}-\mathbf{d}, \mathbf{x}-\mathbf{d}) < \angle(\tilde{\mathbf{u}}, \mathbf{d}-\mathbf{a}) < \frac{\pi}{2}. \tag{9}$$

Z (9) vyplýva, že existuje taký bod $\mathbf{c}$ (bod v ktorom sa stretáva základňa s výškou), ktorý patrí základni $(\mathbf{a}, \mathbf{d})$ trojuholníka $\triangle(\mathbf{x}, \mathbf{a}, \mathbf{d})$. Kedže body $\mathbf{a}, \mathbf{d}$ patria K a K je konvexný mnohosten, platí, že bod $\mathbf{c}$ patrí K. Výška so základňou trojuholníka sú na seba kolmé, preto $\|\mathbf{x}-\mathbf{c}\| < \|\mathbf{x}-\mathbf{a}\|$, čo je však spor s predpokladom.

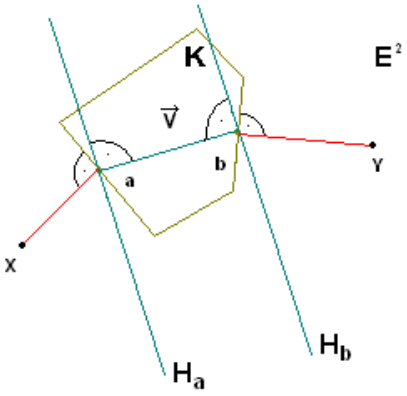
Vráťme sa späť k dôkazu vzťahu (7). Majme nadrovinu H_0 kolmú na vektor $(\mathbf{x}-\mathbf{a})$ a zároveň obsahujúcu bod $\mathbf{a}$ a nadrovinu H_1 kolmú na vektor $(\mathbf{y}-\mathbf{b})$ a zároveň obsahujúcu bod $\mathbf{b}$. Z predchádzajúceho dôkazu vyplýva, že bod $\mathbf{x}$ leží v opačnom polpriestore určenom H_0 ako bod $\mathbf{b}$ a tiež bod $\mathbf{y}$ leží v opačnom polpriestore určenom H_1 ako bod $\mathbf{a}$, a teda

$$\begin{gathered}(\mathbf{x}-\mathbf{a}) \cdot \vec{v} \geq 0 \\ (\mathbf{y}-\mathbf{b}) \cdot \vec{v} \leq 0 \\ \vec{v} = \mathbf{a}-\mathbf{b},\end{gathered}$$

z čoho vyplýva, že body $\mathbf{x}$ a $\mathbf{y}$ ležia na opačných stranách časti priestoru ohraničenej nadrovinami H_a a H_b (poz. obr. 3.69), čím sme dokázali vzťah (7).

Zo vzťahu (7) vyplýva, že funkcia je rovnomerne spojitá. Keďže z rovnomernej spojitosti vyplýva spojitosť, dôkaz je hotový. □

Cvičenie 3.19 (50 bodov). *Nech $K \subseteq \mathbb{E}^d$ je d-rozmerný konvexný mnohosten a zobrazenie $\phi\colon \mathbb{E}^d \to K$ také, že $\phi(\mathbf{x}) = \arg\min_{\mathbf{y}\in K} \|\mathbf{x}-$*

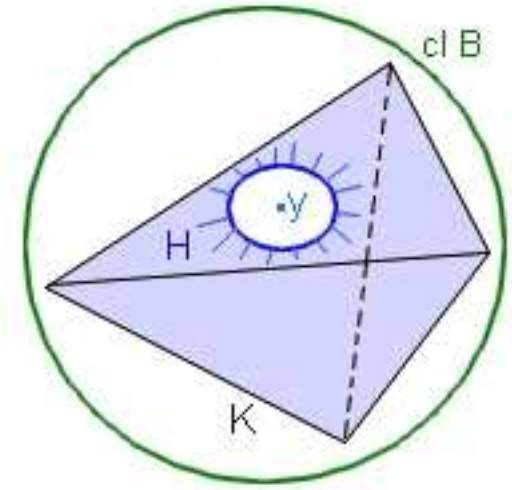


OBR. 3.69. Obrázok znázorňuje oddelenie bodov $\mathbf{x}, \mathbf{y}$ časťou priestoru vymedzenou nadrovinami H_a, H_b v $\mathbb{E}^2$.

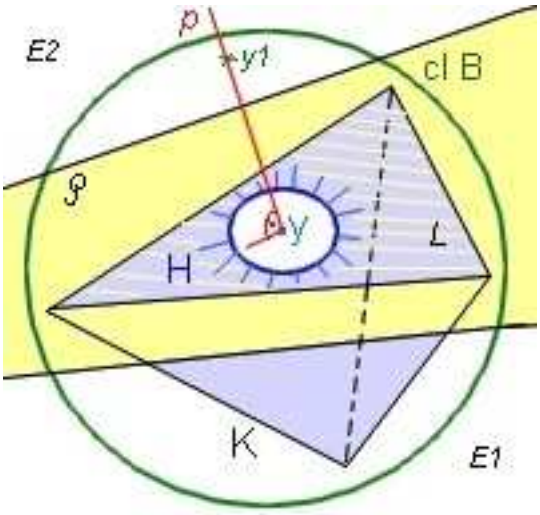
$\mathbf{y}\|$. *Ukážte, že zobrazenie ϕ zobrazuje hranicu guľovej nadplochy obaľujúcej K na hranicu ∂K. Predpokladajte, že zobrazenie ϕ je spojité (pozri cvičenie 3.18).*

Riešenie (Barbora Gallusová, 31.10.2009): Keďže ϕ je dobre definované, určite ku každému $x \in clB$ existuje $\phi(x) \in K$. Platí, že toto $\phi(x) \in \partial K$. Dôkaz sporom: Nech $\phi(x) \in intK$ potom úsečka $x\phi(x)$ obsahuje $x_0 \in \partial K$, ktorý leží medzi $\phi(x)$ a x (pretože $\phi(x) \in intK$ a $x \notin K$). Keďže x_0 leží medzi $\phi(x)$ a x, platí: $||x - x_0|| < ||x - \phi(x)||$ a to je spor, lebo $\phi(x) = argmin_{y \in K}||x - y||$. Ukázali sme teda, že $\phi(x) = argmin_{y \in K}||x - y|| = argmin_{y \in \partial K}||x - y||$. Čiže hranica guľovej plochy sa pomocou ϕ zobrazí na nejakú $H \subseteq \partial K$. Hranica guľovej plochy v E^d je kompaktná množina. Kompaktná množina sa spojitým zobrazením zobrazí na kompaktnú množinu. Takže $H \subseteq \partial K$ je kompaktná množina. Ak $H = \partial K$, je to v poriadku, lebo hranica d-rozmerného konvexného mnohostena je kompaktná množina. Ukážeme, že nemôže nastať prípad $H \subset \partial K$. Dôkaz sporom: Nech $H \subset \partial K$. Existuje teda bod $y \in \partial K$ taký, že $y \notin H$. Ak by sme získali H z ∂K len odobratím bodu y, H by nebola kompaktná (v ∂K z kompaktnosti $\exists\{x_i\}_{i=1}^n$, $x_i \in \partial K$: $lim_{i \to \infty} x_i = y$, každý bod $x_i \in H$, ale $y \notin H$). Ak teda $y \notin H$, potom ani otvorené $O_\varepsilon(y) \notin H$, poz. cvičenie 3.70. Nemôžeme vziať uzavreté okolie, lebo H je kompaktná a teda aj uzavretá.

Tento bod $y \in \partial K$, takže je súčasť aspoň jednej nadsteny K (pretože hranica konvexného K je tvorená nadstenami K). Vyberme jednu z týchto nadstien a označme ju L ($y \in L$). Keďže L je stena dimenzie (d-1), určuje nadrovinu ρ ($L \subset \rho$), ktorá rozdelí priestor E^d na dve disjunktné množiny E_1 a E_2 také, že $E_1 \cup E_2 = E^d$ a $\rho \subset E_1$. Z konvexnosti K vyplýva, že $K \subset E_1$. Zostrojme polpriamku p, ktorá je kolmá na L, vychádza z bodu y a smeruje do E_2. Existuje bod $y_1 \in p \cap bdB$, pretože guľová plocha obaľuje K, poz. 3.71. Hľadajme obraz $\phi(y_1)$. Najbližší bod k y_1 z množiny E_1 určite patrí do ρ (úsečka určená y_1



Obr. 3.70



Obr. 3.71

a ľubovoľným bodom z E_1 obsahuje bod z ρ). A najbližším bodom z ρ k bodu y_1 je $y \in K \subset E_1$, takže $\phi(y_1) = y$. Dôkaz sporom: Nech $\phi(y_1) \neq y$. Potom body $\phi(y_1)$, y, y_1 tvoria pravouhlý trojuholník, kde prepona je úsečka $y_1\phi(y_1)$. Platí teda $||y_1 - \phi(y_1)||^2 > ||y_1 - y||^2$. To je spor s definíciou ϕ. Naozaj teda $\phi(y_1) = y$.

Dostali sme situáciu, že bod $y_1 \in bdB$ sa zobrazením ϕ zobrazí na bod $y \notin H$. To je spor s tým, že ϕ je dobre definované.

□

Riešenie (Alojz Kováčik, 14.11.2009): Nech $B^d(\mathbf{a},\mathbf{r}), \mathbf{a} \in \partial K$ je guľa obsahujúca K vo svojom vnútri. Chceme dokázať $\phi(\partial B^d(\mathbf{a},\mathbf{r})) = \partial K$.

inklúzia "$\subset$":

Keďže guľa $B^d(\mathbf{a},\mathbf{r})$ obsahuje K vo svojom vnútri, je zrejmé, že k ľubovoľnému bodu $\mathbf{x} \in \partial B^d(\mathbf{a},\mathbf{r})$ je z bodov patriacich K najbližšie práve bod patriaci hranici ∂K. Teda $\phi(\partial B^d(\mathbf{s},\mathbf{r})) \subset \partial K$.

inklúzia "$\supset$":

Nech $\mathbf{a} \in \partial K$, $\mathbf{a}$ je hromadným bodom $\mathbb{E}^d \setminus K$, pretože ľubovoľná guľa so stredom v $\mathbf{a}$ obsahuje body množiny $\mathbb{E}^d \setminus K$ rôzne od $\mathbf{a}$. Preto existuje postupnosť $(\mathbf{x_k}), \mathbf{k} \in \mathbb{N}$, patriaca $\mathbb{E}^d \setminus K$, ktorá konverguje k $\mathbf{a}$. Položme $\mathbf{a_k} := \phi(\mathbf{x_k})$ pre $\mathbf{k} \in \mathbb{N}$. Z cvičenia 3.18 máme nerovnosť

$$0 \leq \|\mathbf{a} - \mathbf{a_k}\| = \|\phi(\mathbf{a}) - \phi(\mathbf{x_k})\| \leq \|\mathbf{a} - \mathbf{x_k}\|,$$

z ktorej vyplýva $\mathbf{a} = \lim_{k\to\infty} \mathbf{a_k}$.

$B^d(\mathbf{a},\mathbf{r})$ obsahuje K, a teda aj všetky členy postupnosti $\mathbf{a_k}, \mathbf{k} \in \mathbb{N}$. Nech ďalej P_k sú polpriamky vychádzajúce z bodov $\mathbf{a_k}$ a určené vektormi $(\mathbf{x_k} - \mathbf{a_k})$. Vektory $(\mathbf{x_k} - \mathbf{a_k}) \neq \vec{0}$, pretože $\mathbf{x_k} \in \mathbb{E}^d \setminus K$ a $\mathbf{a_k} \in K$. Dostávame postupnosť $\mathbf{y_k} \in P_k \cap \partial B^d(\mathbf{a},\mathbf{r}), \mathbf{k} \in \mathbb{N}$ a zrejme platí $\phi(\mathbf{y_k}) = \mathbf{a_k}$ pre $\mathbf{k} \in \mathbb{N}$. Z postupnosti $\mathbf{y_k}$ sa dá vybrať podpostupnosť $\mathbf{y_{k_i}}, \mathbf{i} \in \mathbb{N}$, ktorá konverguje k nejakému $\mathbf{y} \in \partial B^d(\mathbf{a},\mathbf{r})$.

Keďže ϕ je spojité zobrazenie,

$$\phi(\mathbf{y}) = \lim_{k\to\infty} \phi(\mathbf{y_{i_k}}) = \lim_{k\to\infty} \mathbf{a_{i_k}} = \mathbf{a}.$$

Teda $\phi(\partial B^d(\mathbf{s},\mathbf{r})) \supset \partial K$. □

Cvičenie 3.20 (50 bodov). *Odhadnite zhora počet k-stien d-rozmerného konvexného mnohostena s n vrcholmi pre $-1 \leq k \leq d$.*

Riešenie (Kristína Lidayová, 27.10.2011): Označme $f_k(P)$ počet k-stien v d-dimenzionálnom konvexnom mnohostene P, kde $k = 0, 1, ..., d$. Presná hodnota horného ohraničenia pre f_k, kde k nadobúda hodnoty $k = 0$ a $k = d$ sa dá určiť jednoducho. Počet 0-stien je rovný počtu vrcholov mnohostena a počet nevlastných d-stien je 1.

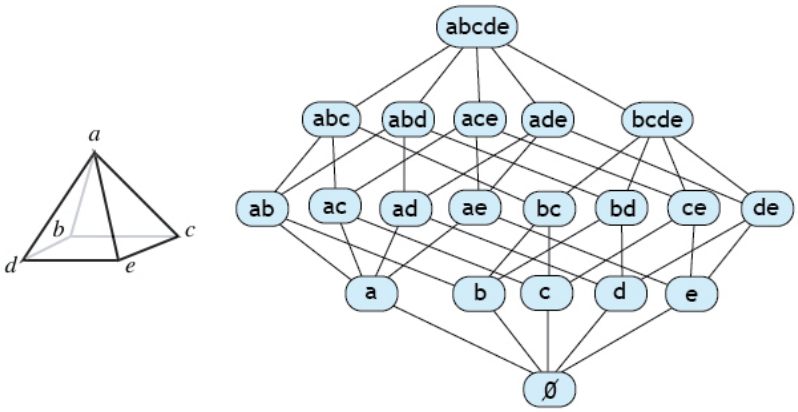
$$f_0(P) = v$$
$$f_d(P) = 1$$

Pre nevlastnú -1-stenu je jej počet tiež 1.

Pre $k = 1, 2, ..., d-1$ McMullenova *Upper Bound Theorem* [1970] dokazuje, že horné ohraničenie počtu $f_k(P)$ cez celý mnohosten P s n vrcholmi dosiahneme cyklickým mnohostenom.

Cyklickým mnohostenom $C(d,n)$ nazývame konvexný obal ľubovoľnej konečnej množiny bodov na momentovej krivke $M(t) = \{(t^1, t^2, ..., t^d)^T :$

$t \in R\}$ v R^d. Keďže kombinačná štruktúra (t.j stenová mriežka) je jednoznačne určená počtom vrcholov n a dimenziou d, ľubovoľný mnohosten kombinačne ekvivalentný cyklickému d-mnohostenu s n vrcholmi možme označiť tiež $C(d, n)$.



Príklad stenovej mriežky ihlanu

Veta 1: (Upper Bound Theorem) [1]
Pre ľubovoľnú dimenziu d, $n \geq d+1$ a $k = 1, 2, \ldots, d-1$ počet k-stien ľubovoľného d-dimenzionálneho konvexného mnohostena s n vrcholmi je nanajvýš toľko ako počet k-stien d-dimenzionálneho cyklického mnohostenu s n vrcholmi.

$$f_k(P) \leq f_k(C(d,n)), \forall k = 1, ..., d-1$$

Dôkaz: [2]
Na dôkaz použijeme nasledujúcu Lému.

Lema: Ľubovoľná podmnožina $k \leq (d+1)$ bodov na momentovej krivke je lineárne nezávislá.

Dôkaz: Nech d+1 bodov $\{M_0, M_1, ..., M_d\}$ na momentovej krivke má hodnoty parametrov $\{t_0, t_1, ..., t_d\}$. Determinant tvorený súradnicami týchto bodov

$$\begin{vmatrix} 1 & t_0 & t_0^2 & ... & t_0^d \\ 1 & t_1 & t_1^2 & ... & t_1^d \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & t_d & t_d^2 & ... & t_d^d \end{vmatrix} = \prod_{0 \leq i < j \leq d} (t_j - t_i)$$

nazývame Vandermondov determinant a nekráti sa, pokiaľ parametre t_i sú po dvojciach rôzne.

Dôsledok: Ľubovoľná nadrovina v E^d pretína momentovú krivku v maximálne d bodoch.

Ako sme už spomenuli cyklický mnohosten je konvexný obal $n \geq (d+1)$ bodov na momentovej krivke. Vlastnosťou cyklického mnohostenu je, že je simpliciálny.

Nech $C(d,n)$ je cyklický mnohosten v E^d s n vrcholmi, a teda konvexný obal n bodov $\{M_1, M_2, ..., M_n\}$ z momentovej krivky $M(t)$ s príslušnými parametrami $\{t_1, t_2, ..., t_n\}$. Nech $\mathcal{I}$ je podmnožina množiny indexov $\{1, 2, ..., n\}$, veľkosť množiny nech je $k \leq \frac{d}{2}$ a uvažujme polynóm

$$\pi_{\mathcal{I}}(t) = \prod_{i \in \mathcal{I}} (t - t_i)^2$$

Tento polynóm má stupeň $2k \leq d$ a existuje bod H^* v E^d a reálne číslo h_0, ktoré môžme použiť v nasledujúcom vzťahu

$$\pi_{\mathcal{I}}(t) = H^* \cdot M(t) - h_0$$

Polynóm $\pi_{\mathcal{I}}(t)$ je kladný a vykráti sa práve vtedy, keď $t = t_i, i \in \mathcal{I}$. Nadrovina H definovaná rovnicou

$$H^* \cdot X = h_0$$

je potom nadrovinou opornou k $C(d,n)$ pozdĺž $(k-1)$-steny $conv(\{M_i, i \in \mathcal{I}\})$. Konvexný obal ľubovoľnej podmnožiny $k \leq \frac{d}{2}$ vrcholov cyklického mnohostena $C(d,n)$ je stenou $C(d,n)$, čo dokazuje nasledujúcu vetu:

Veta: Cyklický mnohosten s n vrcholmi má $\binom{n}{k}(k-1)$-stien, pre každé $0 \leq k \leq \frac{d}{2}$.

Uvažujúc dualitu cyklických mnohostenov platí aj veta:

Veta: Pre ľubovoľné číslo $n \geq (d+1)$ existuje mnohosten v E^d majúci n $(d-1)$-stien a presne $\binom{n}{k}(d-k)$-stien, pre všetky $0 \leq k \leq \frac{d}{2}$.

Tým sme dokázali, že cyklický mnohosten $C(d,n)$ spĺňa horné ohraničenie počtu stien.

□

Počet k-stien cyklického mnohostena $C(d,n)$ vieme explicitne vyjadriť pomocou Vety 2. Vidíme, že horné ohraničenie počtu stien závisí len od počtu vrcholov n a dimenzie d.

Veta 2: [3]
Pre $d \geq 2$ a $0 \leq k \leq d-1$ platí:

$$f_k(C(d,n)) = \frac{n-\delta(n-k-2)}{n-k-1} \sum_{j=0}^{\lfloor d/2 \rfloor} \binom{n-1-j}{k+1-j} \binom{n-k-1}{2j-k-1+\delta}$$

kde $\delta = d - 2\lfloor d/2 \rfloor$.

Kokrétne pre $k = d-1$ platí:

$$f_{d-1}(C(d,n)) = \binom{n-\lfloor\frac{d+1}{2}\rfloor}{n-d} + \binom{n-\lfloor\frac{d+2}{2}\rfloor}{n-d}$$

Príklad:

P	f_0	f_1	f_2	f_3	f_4
$C(5,10)$	10	45	100	105	42
$C(5,20)$	20	190	580	680	272
$C(5,30)$	30	435	1460	1755	702

Zdroje:

[1] *P. McMullen. The maximum numbers of faces of a convex polytope. Mathematika 17:179-184, 1970.*

[2] *Jean-Daniel Boissonnat and Mariette Yvinec. Algorithmic Geometry . Cyclic polytopes on pages 153-154, 1998.*

[3] *Louis J. Billera and Anders Björner. Face Numbers Of Polytopes And Complexes. Theorem 15.3.4 on page 8, 1997.*

□

Cvičenie 3.21 (15 bodov). *Majme konečnú neprázdnu množinu bodov Q. Dokážte, že dva od seba najviac vzdialené body sú vrcholmi* $\mathrm{conv}(Q)$.

Riešenie (Kristína Lidayová, 28.10.2011):

Majme množinu bodov Q a v nej 2 body, ktoré sú od seba najviac vzdialené. Označme ich q_1 a q_2. Zostrojme konvexný obal množiny Q použitím algoritmu QUICKHULL. Keďže konvexný obal nie je závislý od voľby súradnicovej sústavy, zvoľme si takú, ktorej os x je rovnobežná s priamkou prechádzajúcou bodmi q_1 a q_2.

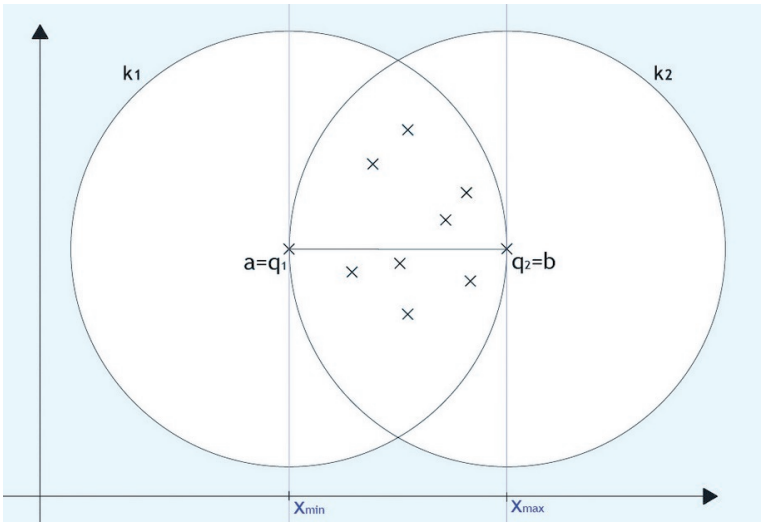
Keďže bod q_2 je od bodu q_1 najviac vzdialený neexistuje žiaden bod x patriaci množine Q, ktorý by bol od bodu q_1 ďalej. To znamená, že žiaden bod sa nenachádza mimo kruhu k_1 daného stredom q_1 a polomerom $|q_1q_2|$. Z rovnakého dôvodu sa žiaden bod $x \in Q$ nenachádza ani mimo kruhu k_2 daného stredom q_2 a polomerom $|q_1q_2|$ (poz. obr. 3.72).

Podľa algoritmu QUICKHULL nájdeme bod $a \in Q$ s minimálnou x-súradnicou a bod $b \in Q$ s maximálnou x-súradnicou. Je vidieť, že body a, b sa zhodujú s našimi bodmi q_1, q_2. Z algoritmu QUICKHULL vieme, že obidva tieto body a, b sú vrcholmi konvexného obalu. Dokázali sme teda, že aj body q_1, q_2, ktoré sú s nimi totožné sú vrcholmi konvexného obalu.

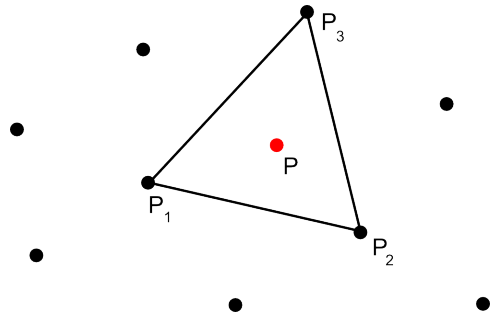
□

Cvičenie 3.22. *Dokážte, že bod P nie je extremálnym bodom konvexnej množiny $C \subset \mathbb{E}^2$ s neprázdnym vnútrom práve vtedy, ak leží v nejakom trojuholníku s vrcholmi ležiacimi v C, ale sám nie je vrcholom tohto trojuholníka.*

Riešenie (Jakub Mišún, 18.11.2011):



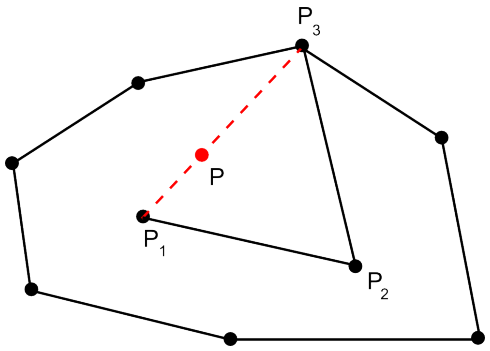
Obr. 3.72. Vymedzenie konvexného obalu kruhmi so stredmi v dvoch najvzdialeneších bodoch konvexného obalu a polomermi rovnými vzdialenosti stredov.



Obr. 3.73. Bod P nie je extremálny

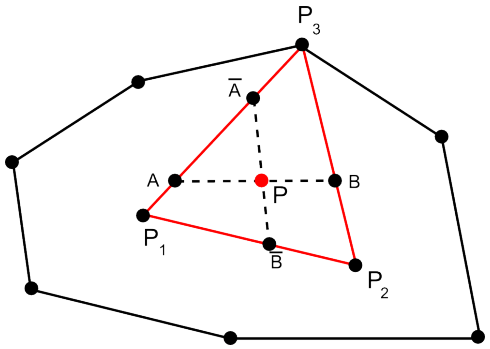
Bod P z konvexnej množiny C sa nazýva extremálnym, ak neexistujú žiadne dva rôzne body $A, B \in C$ také, že P leží na úsečke AB. To znamená, že nám stačí ukázať, že ak bod P leží v trojuholníku $P_1P_2P_3$ s vrcholmi ležiacimi v C, pričom sám nie je jeho vrcholom, tak potom nutne leží medzi dvoma bodmi z C. Bod P môže v trojuholníku nadobúdať 2 rôzne polohy. Môže ležať na jeho hrane alebo v jeho vnútri. Prípad keď leží na hrane je triviálny, pretože všetky jeho vrcholy P_1, P_2, P_3 ležia v C a P leží medzi dvoma z nich. Teda P triviálne nie je extremálnym bodom.

Čo ak sa bod P nachádza vo vnútri trojuholníka? Z vlastností konvexnej množiny C vieme, že ak nejaké dva body A, B patria množine C, tak jej patria i všetky body medzi nimi. Preto množine C patria všetky body z hrán trojuholníka $P_1P_2P_3$. Keďže P sa nachádza v trojuholníku $P_1P_2P_3$, leží nutne medzi nejakými dvoma bodmi A, B jeho



OBR. 3.74. Bod P leží medzi bodmi P_1 a P_3

hranice (takých dvojíc je nekonečne veľa). Potom P leží medzi dvoma bodmi z C a teda nie je extremálny.



OBR. 3.75. Bod P leží medzi dvoma bodmi z hranice trojuholníka $P_1P_2P_3$. Takáto dvojica nie je len jedna.

□

CVIČENIE 3.23 (35 bodov). *Napíšte podrobne procedúru vytvárania horného konvexného obalu pre algoritmus udržiavania konvexného obalu v* $\mathbb{E}^2$. *Nezabudnite na vyvažovanie stromu. Ilustrujte na príklade.*

CVIČENIE 3.24 (30 bodov). *Napíšte podrobne procedúru inkrementálneho vytvárania konvexného obalu v* $\mathbb{E}^2$, *ktorý je optimálny.*

CVIČENIE 3.25 (30 bodov). *Napíšte podrobne procedúru inkrementálneho vytvárania konvexného obalu v* $\mathbb{E}^3$.

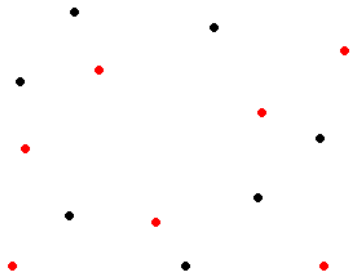
KAPITOLA 4

Proximita

CVIČENIE 4.1 (45 bodov). *Nech A a B sú množiny bodov v rovine, každá má n prvkov. Nájdite dva najbližšie body tak, že jeden je z A a druhý z B. Ukážte, že tento problém vyžaduje aspoň $\Omega(n \log n)$ operácií. Pouvažujte, či sa niečo zmení v prípade, ak A a B sú lineárne separovateľné.*

Riešenie (Dalibor Hrtánek, 19.11.2009):

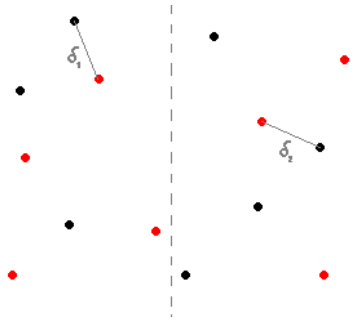
Máme množinu $\mathbf{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ a množinu $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ (pozri obr. 4.1). Úlohu vyriešime metódou typu *rozdeľuj a panuj*. Vytvoríme množinu $\mathbf{S} = \mathbf{A} \cup \mathbf{B} = \{\mathbf{a}_1, \ldots, \mathbf{a}_n, \mathbf{b}_1, \ldots, \mathbf{b}_n\}$ a utriedime ju lexikograficky podľa y-ovej súradnice bodov ($\mathcal{O}(n \log n)$). Množinu $\mathbf{S}$ potom rozdelíme vertikálnou priamkou l na dve polovice do množín $\mathbf{S_1}$ a $\mathbf{S_2}$ a v každej z nich rekurzívne nájdeme riešenie. Priamka l prechádza cez medián z x-ových súradníc množiny $\mathbf{S}$ ($\mathcal{O}(n)$).



OBR. 4.1. vstupná množina bodov

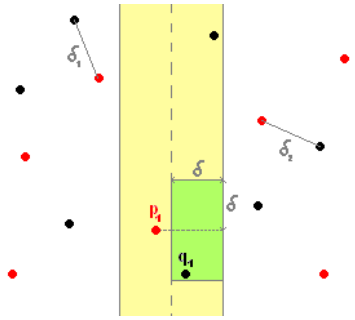
V množine $\mathbf{S_1}$ nájdeme dva najbližšie body tak, že jeden je z $\mathbf{A}$ a druhý z $\mathbf{B}$, ich vzdialenosť označíme δ_1. Ak $\mathbf{S_1}$ neobsahuje žiaden bod z $\mathbf{A}$ alebo žiaden bod z $\mathbf{B}$, tak $\delta_1 = \infty$, inak $\delta_1 = dist(\mathbf{p}, \mathbf{q})$, kde $\mathbf{p} \in \mathbf{A} \wedge \mathbf{q} \in \mathbf{B}$ sú také, že ich vzdialenosť je najmenšia spomedzi všetkých takých dvojíc. Rovnako v množine $\mathbf{S_2}$ nájdeme dva najbližšie body tak, že jeden je z $\mathbf{A}$ a druhý z $\mathbf{B}$ a ich vzdialenosť označíme δ_2 (pozri obr. 4.2). Najmenšiu z týchto vzdialeností označíme $\delta = \min\{\delta_1, \delta_2\}$.

V kroku spájania čiastkových riešení musíme zistiť, či existujú také dva body $\mathbf{s}_1 \in \mathbf{S_1} \cap \mathbf{A}$ a $\mathbf{s}_2 \in \mathbf{S_2} \cap \mathbf{B}$, ktorých vzdialenosť δ_3 by bola menšia ako δ, t.j. $\delta_3 = dist(\mathbf{s}_1, \mathbf{s}_2) < \delta$ (body na hranici). Budeme prechádzať body z $\mathbf{S_1} \cap \mathbf{A}$ a zisťovať ich vzdialenosť od bodov z $\mathbf{S_2} \cap \mathbf{B}$. Ak by sme museli prejsť všetky takéto body, tento krok by mal časovú

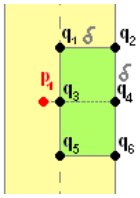


OBR. 4.2. rozdelenie roviny a čiastkové riešenia

zložitosť $\mathcal{O}(n^2)$. Vieme však, že najmenšia vzdialenosť už nemôže byť väčšia ako δ, takže pre každý bod $\mathbf{p}_i \in \mathbf{S_1} \cap \mathbf{A}$, ktorého x-ová súradnica je väčšia ako $l - \delta$, stačí kontrolovať maximálne 6 bodov $\mathbf{q}_j \in \mathbf{S_2} \cap \mathbf{B}$, ktorých x-ová súradnica je menšia ako $l + \delta$ a y-ová súradnica je menšia ako $p_{iy} + \delta$ a väčšia ako $p_{iy} - \delta$, kde p_{iy} je y-ová súradnica bodu $\mathbf{p}_i$ (pozri obr. 4.3,4.4). Sú to tie body, ktoré sú po priemete na deliacu priamku vo vzdialenosti menšej ako δ od priemetu bodu $\mathbf{p}_i$ a vieme ich získať v čase $\mathcal{O}(n)$.



OBR. 4.3. oblasť možných kandidátov $\mathbf{q}_j$



OBR. 4.4. maximálny počet kandidátov $\mathbf{q}_j$

Spomedzi týchto dvojíc bodov $\mathbf{p}_i$ a $\mathbf{q}_j$ nájdeme $\mathbf{s}_1$ a $\mathbf{s}_2$ také, že $\delta_3 = dist(\mathbf{s}_1, \mathbf{s}_2) < dist(\mathbf{p}_i, \mathbf{q}_j), \forall i, j$. Ak $\delta_3 < \delta$, tak $\mathbf{s}_1$ a $\mathbf{s}_2$ sú najbližšie také body, že $\mathbf{s}_1 \in \mathbf{A}$ a $\mathbf{s}_2 \in \mathbf{B}$ a ich vzdialenosť je δ_3. Ak neexistuje žiaden bod $\mathbf{p}_i$ alebo žiaden bod $\mathbf{q}_j$ alebo $\delta_3 > \delta$, tak najbližšie body sú body z čiastkového riešenia a ich vzdialenosť je δ.

Algoritmus rekurzívne rozdeľuje rovinu zvislými priamkami *v strede* množiny, až kým nedôjde k triviálnemu prípadu, kedy v podmnožine $\mathbf{S_i}$ zostanú len dva alebo tri body. Ak sú všetky body z $\mathbf{S_i}$ bodmi z $\mathbf{A}$ alebo všetky bodmi z $\mathbf{B}$, tak najmenšia vzdialenosť bodov z $\mathbf{S_i}$ takých, že jeden je z $\mathbf{A}$ a druhý je z $\mathbf{B}$, $\delta_i = \infty$. Ak jeden bod z $\mathbf{S_i}$ je bodom z $\mathbf{A}$, označme ho $\mathbf{a_0}$ a zvyšný jeden (alebo dva) body sú z $\mathbf{B}$, označme ich $\mathbf{b_1}$ (prípadne $\mathbf{b_1}, \mathbf{b_2}$), tak $\delta_i = dist(\mathbf{a_0}, \mathbf{b_1})$ ($\delta_i = \min\{dist(\mathbf{a_0}, \mathbf{b_1}), dist(\mathbf{a_0}, \mathbf{b_2})\}$). Analogicky, ak jeden bod z $\mathbf{S_i}$ je bodom z $\mathbf{B}$, označme ho $\mathbf{b_0}$ a zvyšný jeden (alebo dva) body sú z $\mathbf{A}$, označme ich $\mathbf{a_1}$ (prípadne $\mathbf{a_1}, \mathbf{a_2}$), tak $\delta_i = dist(\mathbf{b_0}, \mathbf{a_1})$ ($\delta_i = \min\{dist(\mathbf{b_0}, \mathbf{a_1}), dist(\mathbf{b_0}, \mathbf{a_2})\}$).

Analýza časovej zložitosti:

Rozdelenie prebieha v čase $\mathcal{O}(n)$. Spájanie čiastkových riešení vďaka utriedeniu bodov na začiatku algoritmu trvá čas $\mathcal{O}(n)$. Rekurentný vzťah pre počet krokov môžeme zapísať v tvare $T(n) = 2T(\frac{n}{2}) + \mathcal{O}(n)$. S využitím hlavnej vety dostávame $T(n) = \mathcal{O}(n \log n)$. Dolná hranica časovej zložitosti algoritmu je teda $\Omega(n \log n)$.

Ak sú množiny $\mathbf{A}$ a $\mathbf{B}$ lineárne separovateľné, tak existuje priamka, ktorá rozdeľuje rovinu na dve polroviny tak, že v jednej sa nachádzajú len body z $\mathbf{A}$ a v druhej len body z $\mathbf{B}$. Problém nájdenia dvoch najbližších bodov takých, že jeden je z $\mathbf{A}$ a druhý z $\mathbf{B}$ je v tomto prípade totožný s nájdením vzdialenosti množín $\mathbf{A}$ a $\mathbf{B}$. Aj pri tomto probléme je nutné kontrolovať každý bod z $\mathbf{A}$ s každým bodom z $\mathbf{B}$, čo pri použití triedenia a metódy typu *rozdeľuj a panuj* vedie opäť k dolnej hranici časovej zložitosti $\Omega(n \log n)$. □

Riešenie (Viktória Bakurová, 2.1.2010): Majme množinu A s prvkami $a_1, \dots a_n$ a množinu B s prvkami $b_1, \dots b_n$. Hľadáme takú dvojicu bodov a_i, b_j, ktoré sú najmenej vzdialené spomedzi všetkých možných dvojíc.

Najjednoduchší spôsob je nájsť pre každý bod z A najbližší bod z B a vybrať z týchto dvojíc dvojicu s najmenšou vzdialenosťou. Zložitosť takéhoto prístupu je $O(n^2)$. Táto zložitosť nie je optimálna. Hľadáme algoritmus s menšou časovou zložitosťou.

Pri riešení daného problému využijeme Voronoiove diagramy (VD) a jeho vlastnosti. Zostrojme VD množiny A.Zoberme si ľubovoľný prvok množiny B,nech je to prvok b_k. Hľadáme Voronoiovu oblasť, v ktorej tento bod leží. Nech je to $V(a_i)$. Potom najbližší bod množiny A k bodu b_k je generátor tejto oblasti, t.j. prvok a_i.

Ako nájsť oblasť, v ktorej leží b_k?

Zvolíme si prístup s časovou zložitosťou $O(\log n)$, kde n je počet prvkov množiny A. Majme zotrojený VD tejto množiny. Hľadáme najbližší bod z množiny A k ľubovoľnému bodu q, t.j. Voronoiovu oblasť, v ktorej tento bod leží.

Cez každý vrchol VD vedieme vodorovnú priamku. Táto priamka rozdelí jednotlivé regióny na trojuholníky a lichobežníky.

Vodorovnými priamkami sa rovina rozdelý na vodorovné "pásy". Okrem toho každý z týchto "pásov" pretína hrany Voronoiovho diagramu a tie sú vzhľadom na ne usporiadané.

Binárnym vyhľadávaním cez y-ovú súradnicu bodu q nájdeme teda "pás", v ktorom bod q leží a následne binárnym vyhľadávaním cez x-ovú súradnicu aj hrany VD, medzi ktorými bod q leží. V čase $\mathcal{O}(\log n)$ teda máme Voronoiovu oblasť, v ktorej bod q leží.

Týmto spôsobom nájdeme pre každý prvok množiny B Voronoiovu oblasť, v ktorej daný prvok leží a tým aj najbližší prvok k tomuto bodu z množiny A. Posledným krokom je nájsť minimálnu vzdialenosť spomedzi všetkých vzdialeností nájdených dvojíc.

Algoritmus:

1) zostroj VD množiny A

2) pre každý prvok množiny Bnájdi Voronoiovu oblasť, v ktorej leží

3) spočítaj vzdialenosť každého prvku z B a generátora oblasti, v ktorej prvok leží

4) z n vzdialeností vyber minimum

5) máme 2 najbližšie prvky také, že jeden je z A a druhý z B

časová zložitosť:

1) zostrojenie VD: $\mathcal{O}(n \log n)$

2) nájdenie oblasti pre jeden prvok: $\mathcal{O}(\log n)$
   nájdenie oblasti pre n prvkov: $\mathcal{O}(n \log n)$

3) spočítanie jednej vzdialenosti: $\mathcal{O}(1)$
   spočítanie n vzdialeností: $\mathcal{O}(n)$

4) výber minima: $\mathcal{O}(n)$

Celková časová zložitosť: $\mathcal{O}(n \log n)$

Chceme ukázať, že $\Omega(n \log n)$ je dolná časová zložitosť problému.

Najprv si ukážeme, že pre daných n bodov roviny určiť, ktorý je najbližšie k zadanému bodu q, si vyžaduje čas aspoň $\Omega(\log n)$. Je jasné, že z toho potom vyplýva, že riešenie nášho problému si vyžaduje čas aspoň $\Omega(n \log n)$.

Pri probléme vyhľadania najbližšieho suseda budeme vychádzať z binárneho vyhľadávania. Nech máme daných n reálnych čísel $x_1, \ldots, x_n$ a dané číslo x.Chceme nájsť prvok k nemu najbližší z danej množiny. Najjednoduchší prístup je použitie binárneho vyhľadávania, ktorého dolná časová zložitosť je $\Omega(\log n)$. Tento problém je vlastne jednorozmernou verziou nášho pôvodného problému – vyhľadanie najbližšieho prvku v rovine, preto dolná časová zložitosť $\Omega(\log n)$ je aj dolnou časovou zložitosťou rovinnej verzie.

Nech teraz vieme, že A, B sú lineárne separovateľné. Urobme takú transformáciu roviny, aby sa body množín A a B dali oddeliť vodorovnou priamkou a množina B ležala "nad" množinou A. Nájdeme bod VD množiny A s maximálnou $y-$ovou súradnicou a vodorovnú priamku

vedieme cez tento bod. Všetky prvky množiny B ležia nad touto priamkou, preto ak hľadáme pre nejaký bod b_i Voronoiovu oblasť, v ktorej leží, stačí nám prehľadať jeden "pás", v ktorom tento bod určite leží.

V najhoršom prípade sa časová zložitosť algoritmu nezmení aemphni vtedy, keď sú množiny A, Blineárne separovateľné. □

Cvičenie 4.2 (30 bodov). *Nech A a B sú dve množiny bodov v rovine. Použite Voronoiov diagram na zistenie hodnoty*

$$\min_{a\in A}\min_{b\in B}\operatorname{dist}(a,b)$$

v čase $\mathcal{O}(n\log n)$, kde $n = |A| + |B|$.

Riešenie (Viktória Bakurová, 22.12.2009): Nech množina A má i prvkov, ozn. $a_1, \dots a_i$ a množina B má j prvkov, ozn. $b_1, \dots b_j$, pričom $i + j = n$.

Zostrojme Voronoiov diagram množiny A. Pre každý bod z množiny B nájdeme oblasť tohoto VD, do ktorej patrí a tým aj najbližší prvok z množiny Ak bodu z B.Takto nájdeme j dvojíc, z ktorých vyberieme minimum.

Našli sme $\min_{a\in A}\min_{b\in B}\operatorname{dist}(a,b)$.

Ako nájsť oblasť VD, do ktorej patrí bod q?

Zvolíme si prístup s časovou zložitosťou $\mathcal{O}(\log n)$, kde n je počet prvkov množiny, ktorej VD máme zostrojený. Hľadáme najbližší bod z tejto množiny k ľubovoľnému bodu q t.j. Voronoiovu oblasť, v ktorej tento bod leží.

Cez každý vrchol VD vedieme vodorovnú priamku. Táto priamka rozdelí jednotlivé regióny na trojuholníky a lichobežníky.

Vodorovnými priamkami sa rovina rozdelý na vodorovné "pásy". Okrem toho každý z týchto "pásov" pretína hrany Voronoiovho diagramu a tie sú vzhľadom na ne usporiadané.

Binárnym vyhľadávaním cez $y-$ovú súradnicu bodu q nájdeme teda "pás", v ktorom bod q leží a následne binárnym vyhľadávaním cez x-ovú súradnicu aj hrany VD, medzi ktorými bod q leží. V čase $\mathcal{O}(\log n)$ teda máme Voronoiovu oblasť, v ktorej bod q leží.

Týmto spôsobom nájdeme pre každý prvok množiny B Voronoiovu oblasť, v ktorej daný prvok leží a tým aj najbližší prvok k tomuto bodu z množiny A.

Algoritmus:

1) zostroj VD množiny A

2) pre každý prvok b_k, k=1, $\dots j$, množiny B nájdi Voronoiovu oblasť, v ktorej leží a spočítaj vzdialenosť b_k od generátora

3) z nájdených vzdialeností vyber minimum

časová zložitosť:

1) zostrojenie VD: $\mathcal{O}(i\log i)$

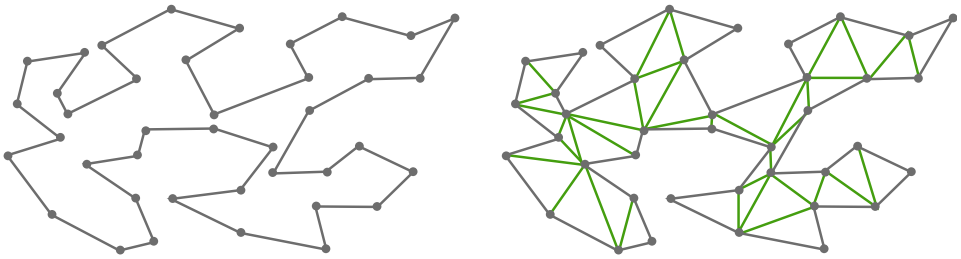
2) nájdenie oblasti pre jeden prvok: $\mathcal{O}(\log i)$

nájdenie oblasti pre j prvkov: $\mathcal{O}(j \log i)$
spočítanie jednej vzdialenosti: $\mathcal{O}(1)$
spočítanie j vzdialeností: $\mathcal{O}(j)$
3) výber minima: $\mathcal{O}(j)$
Celková časová zložitosť: $\mathcal{O}((i+j)\log i) = \mathcal{O}(n \log n)$ □

Cvičenie 4.3 (30 bodov). *Popíšte algoritmus na trianguláciu ľubovoľného n-uholníka, tak aby jeho zložitosť bola* $\mathcal{O}(n \log n)$.

Riešenie (Matej Hudák, 25.11.2010):

Problém triangulácie spočíva v prerozdelení určitej oblasti na jednoduché časti (primitívy). V jednoduchom prípade, v rovine, je na vstupe n-uholník a cieľom je rozdeliť ho na trojuholníky (poz. obr. 4.5).



Obr. 4.5. Vľavo n-uholník, vpravo jeho triangulácia

Hľadáme algoritmus, ktorého zložitosť bude $\mathcal{O}(n \log n)$. Náš algoritmus rozdelíme na dva kroky:

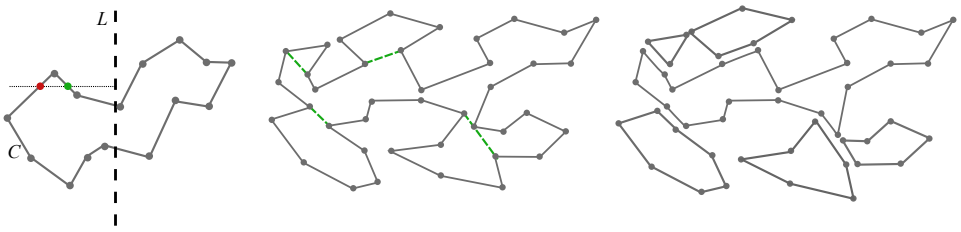
(1) ukážeme ako triangulovať monotónny n-uholník
(2) ukážeme ako konvertovať ľubovoľný n-uholník na monotónne mnohouholníky v x-ovom smere

Zložitosť prvého algoritmu je $\mathcal{O}(n)$ a druhého $\mathcal{O}(n \log n)$, z čoho vyplýva, že celková zložitosť nášho algoritmu je $\mathcal{O}(n \log n)$.

Triangulácia množiny bodov v rovine je hranovo maximálny planárny graf, daný n vrcholmi. Ako štruktúru na prácu s týmto algoritmom si môžeme zvoliť $DCEL$ (double-connected edge list), kde je každá entita zložená z vrcholov, hrán a trojuholníkov, pričom každá entita je spojená so susediacim elementom.

Monotónny n-uholník

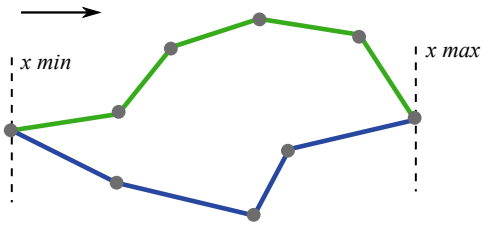
Lomená čiara C je monotónna vzhľadom na priamku L, ak ľubovoľná priamka kolmá na L má prienik s C najviac v jednom bode. N-uholník P sa nazýva monotónny vzhľadom na priamku L ak jeho hranicu ∂P môžeme rozdeliť na dve lomené čiary, ktoré sú monotónne vzhľadom na priamku L.



Obr. 4.6. Zľava doprava, nemonotónny mnohouholník, pridanie diagonál, rozdelenie na monotónne mnohouholníky

Na obr. 4.6 vľavo je n-uholník, ktorý nie je monotónny. Pridaním diagonál rozdelíme n-uholník na monotónne časti. Monotónnosť n-uholníkov budeme určovať vzhľadom na x-ovú os a budeme ich nazývať horizontálne monotónne. Test na monotónnosť (obr. 4.7) mnohouholníka urobíme nasledovne:

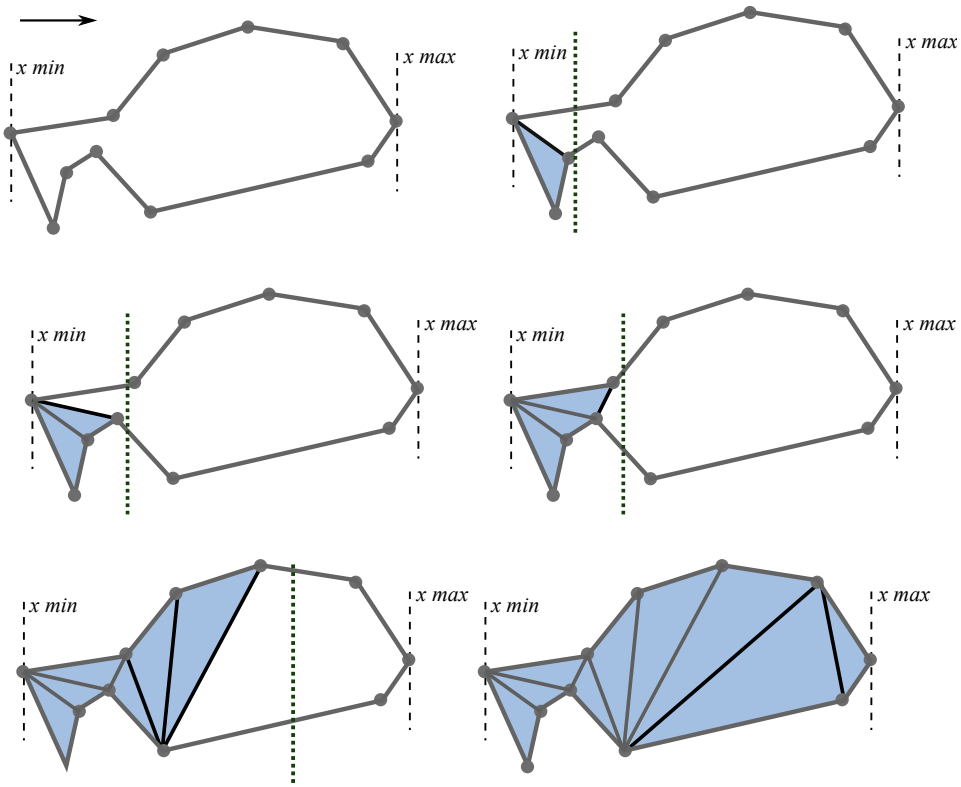
- nájdeme najľavejší a najpravejší vrchol (min a max x-ové súradnice) v čase $\mathcal{O}(n)$
- tieto vrcholy rozdelia n-uholník na dve lomené čiary, hornú a dolnú
- prechodom zľava doprava po každej z lomených čiar testujeme, či sú x-ové súradnice vrcholov neklesajúce, pričom to zaberie čas v najhoršom prípade $\mathcal{O}(n)$



Obr. 4.7. Test na monotónnosť v x-ovom smere

Triangulácia monotónneho n-uholníka

Monotónny n-uholník môžeme triangulovať jednoduchou variáciou rovinnej zametacej metódy. Začneme predpokladom, že vrcholy n-uholníka boli usporiadané podľa x-ových súradníc. Potom nepotrebujeme triediť, ale stačí ak použijeme vrchnú a spodnú lomenú čiaru z predchádzajúceho testu monotónnosti.



Obr. 4.8. Triangulácia monotónneho mnohouholníka

Tieto dve lomené čiary zlúčime v čase $\mathcal{O}(n)$. Postup triangulácie na obr. 4.8 je založený na prechode lomenou čiarou n-uholníka. Prechádzame zľava doprava a postupne pridávame diagonály. Očividne pridanie prvej diagonály nastane až keď prejdeme tri vrcholy. Pridanie ďalších diagonál robíme po prekročení ďalšieho vrchola.

Vrcholy si uchovávame v poli spolu s informáciou, ktorej lomenej čiare daný vrchol patrí. Z testu o monotónnosti si pamätáme maximálnu x-ovú súradnicu. Predpokladáme, že každá z lomených čiar obsahuje aspoň dve časti. Vrchol $v[i]$ je práve spracovávaný.

Postup môžeme opísať nasledujúcim pseudokódom:

```
int i =0;
vrchol u = v[0];
while(v[i].x != max)do
{
   If(prvyPripad)
   {
      pridajDiagonaly(v[i], v[i-1], u);
      u = v[i-1];
   }
   else if(druhyPripad)
   {
```

```
        pridajDiagonaly(v[i], v[i-1], u);
    }
    i++;
}
```

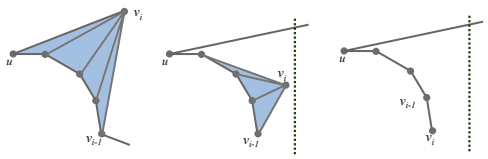
Rozhodnutie pridania alebo nepridania diagonály má dva špeciálne prípady (pozri obrázok 5):

(1) v_i leží v opačnej lomenej čiare ako v_{i-1}:
V tomto prípade pridáme diagonály spojením v_i s každým vrcholom lomenej čiary až spätne k vrcholu u. V tomto prípade vrchol $v[i]$ je viditeľný z vrcholov, s ktorými sa práve spája lomenou čiarou, vzhľadom na tvar vrchnej a spodnej lomenej čiary a monotónnosti n-uholníka.

(2) v_i leží v rovnakej lomenej čiare ako v_{i-1}:
V tomto prípade pridáme diagonály spojením v_i postupne so všetkými vrcholmi v tej istej lomenej čiare, až kým nenastane prípad, že v_i nie je viditeľný z niektorého predchádzajúceho vrchola. Môžu nastať dva prípady, ktoré sú na obrázku 5. Vďaka monotónnosti v x-ovom smere a tvare lomených čiar, do konca prechodu príde vrchol z opačnej lomenej čiary.

Analýza zložitosti:

Usporiadaný zoznam vrcholov vieme konštruovať v čase $\mathcal{O}(n)$ pomocou zlúčenia. Test na pridanie / nepridanie diagonály trvá $\mathcal{O}(1)$ a pridanie ďalšej diagonály vieme spraviť v konštantnom čase. Keďže diagonál v triangulácií je $n-3$, celý tento postup trvá $\mathcal{O}(n)$.



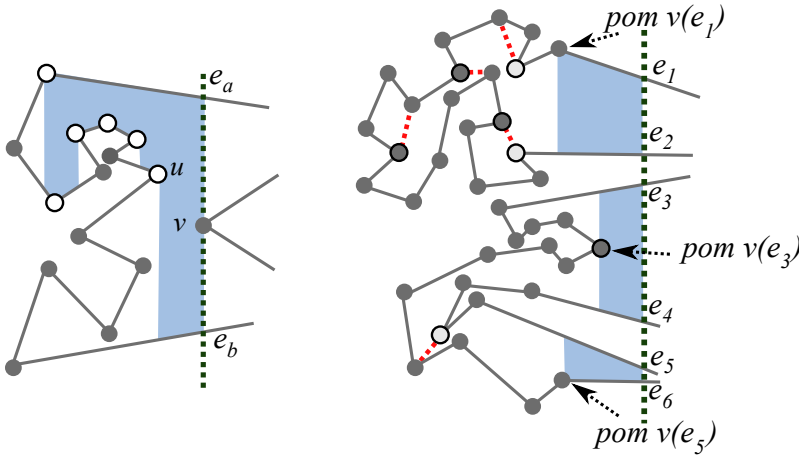
Obr. 4.9. Špeciálne prípady

Rozdelenie ľubovoľného n-uholníka vzhľadom na monotónnosť

Ako teda použijeme predchádzajúci algoritmus na triangluáciu ľubovoľného n-uholníka? Pomocou predspracovania. To čo spravíme je jednoducho predspracovanie n-uhoníka s výsledkom možného prerozdelenia na monotónne mnohouholníky.

Na tento postup použijeme tiež jednoduchú zametaciu metódu. Všimnime si, že porušenie monotónnosti n-uholníka v x-ovom smere nastáva ak v niektorom vrchole je vnútorný uhol väčší ako 180°. Navyše hrany vychádzajúce z tohto vrcholu ležia buď obidve v vľavo, alebo obidve vpravo. Prvý prípad nazveme *zlučovaný vrchol* a druhý bude *rozdeľovaný vrchol*.

Našou metódou zametania zľava doprava pridávame diagonály všetkým rozdeľovaným vrcholom. Opačnou metódou zametania zprava doľava pridáme diagonály všetkým zlučovaným vrcholom.



Obr. 4.10. Zametacia metóda, druhy vrcholov

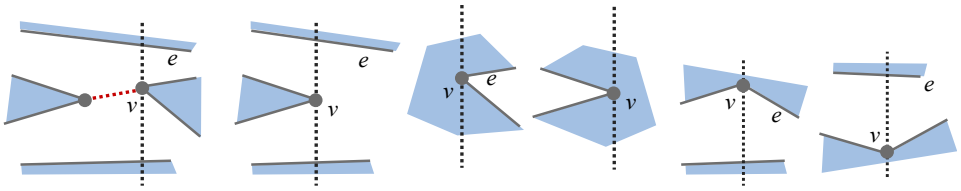
Vezmime si rozdeľovaný vrchol v (obr. 4.10 vľavo). Ak sa pomocou zametania dostaneme k tomuto vrcholu tak existuje hrana e_a nad týmto vrcholom a hrana e_b pod ním. Tu sa musíme starať o vrchol, ktorý je viditeľný z každého rozdeľovacieho vrchola, ktorý môže ležať medzi e_a a e_b. Kvôli tomu budeme osvetľovať vrcholy smerom dole. Na obrázku sú takto osvetlené vrcholy označené bielou farbou. Tieto vrcholy sú vertikálne viditeľné. Potom nech vrchol u je vrchol s najväčšou x-ovou súradnicou spomedzi vertikálne viditeľných vrcholov. Vrchol u je viditeľný zo všetkých vrcholov, ktoré môžu ležať na zametacej priamke medzi e_a a e_b. Pomocný vrchol (na obr. 4.10 vpravo) je vrchol s najväčšou x-ovou súradnicou a vertikálne viditeľný na lomenej čiare medzi e_a a e_b. Tieto pomocné vrcholy sa budú počas prechodu meniť.

Štruktúra zametacia priamka obsahuje zoznam hrán, ktoré pretne zametacia priamka pri prechode. Každým ďalším posunutím aktualizujeme tieto hrany. Ak tu zvolíme reprezentáciu vyváženým binárnym stromom, operácie ako insert, delete, find budú v čase $\mathcal{O}(\log n)$. Pri prechádzaní n-uholníkom budeme spájať každý rozdeľovaný vrchol a $pom\, v(e_a)$. Pracovať budeme s koncovými vrcholmi hrán. Existuje 6 možných prípadov koncových vrcholov v(poz. obr. 4.11):

(1) **rozdeľovací vrchol:**
Nájdeme v zametacej priamke hranu, ktorá leží nad v. Pridáme diagonálu spájajúcu v a $pom\, v(e)$. Aktualizujeme zametaciu čiaru a do $pom\, v$ priradíme v.

(2) **zlučovací vrchol:**
Nájdeme v zametacej priamke dve hrany incidentné k v a zmažeme ich. Vrchol v bude nový *pom* v hrany, ktorá leží priamo nad zmazanými hranami.
(3) **štartovací vrchol:**
Vložíme hrany tohto vrcholu do zametacej priamky. *Pom* v vyššej hrany bude v.
(4) **koncový vrchol:**
Vymažeme obe hrany zo zametacej priamky.
(5) **vrchol ležiaci na vrchnej lomenej čiare:**
V zametacej priamke vymeníme ľavú hranu za pravú hranu. Vrchol v bude nový *pom* v tejto pridanej hrany.
(6) **vrchol ležiaci na spodnej lomenej čiare:**
Podobne ako predchádzajúci prípad, ale vrchol v bude nový *pom* v hrany ktorá je priamo nad vymenenou hranou v zametacej priamke.

Nakoniec v prípadoch $4, 5, 6$ otestujeme, či je *pom* v je zlučovací vrchol a ak áno tak pridáme diagonálu.



Obr. 4.11. Rôzne prípady

Analýza zložitosti:
Tento zametací algoritmus beží v čase $\mathcal{O}(\log n)$ pre každý kontrolovaný vrchol. Tých je tu n, čiže čas zložitosti rozdelenia n-uholníka na monotónne mnohouholníky je $\mathcal{O}(n \log n)$. Dôležité je, že pri prerozdelení n-uholníka na monotónne mnohouholníky nevytvárame viac ako n nových vrcholov. Keďže zložitosť algoritmu na triangulovanie monotónneho mnohouholníka v x-ovom smere je $\mathcal{O}(n)$ a jeho použitie v našom prípade beží v $\mathcal{O}(2n)$, platí, že celková zložitosť je $\mathcal{O}(n \log n)$.
□

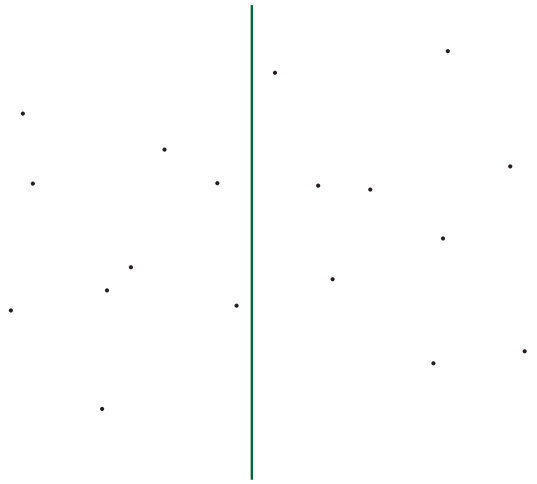
Cvičenie 4.4 (40 bodov). *Popíšte algoritmus vyhľadania najbližšej dvojice bodov v rovine metódou delenia roviny a odhadnite jeho zložitosť.*

Riešenie (Dušan Pácal, 21.12.2008):
Na vstupe máme množinu bodov $P \subset \mathbb{E}^2$. Body sú usporiadané lexikograficky najprv podľa x-ovej, potom podľa y-ovej súradnice.

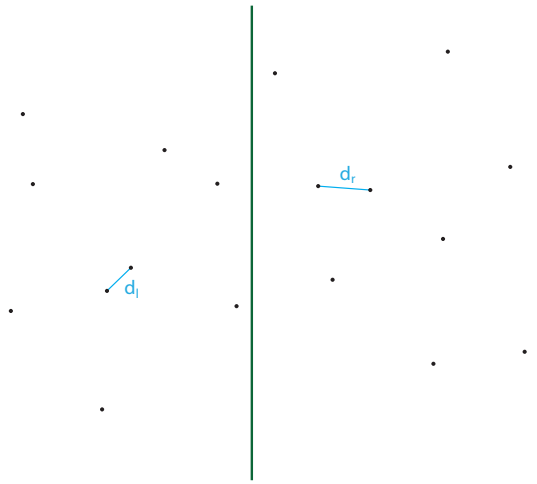
Hľadáme takú dvojicu bodov z P, že ich vzdialenosť je minimálna.

Náš algoritmus bude pracovať nasledovne:

- rozdelíme rovinu na dve časti (pozri obr. 4.12)
- v každej z častí nájdeme riešenie (pozri obr. 4.13)
- spojíme časti a zistíme či je riešenie správne, ak nie nájdeme ho v okolí hranice spojenia



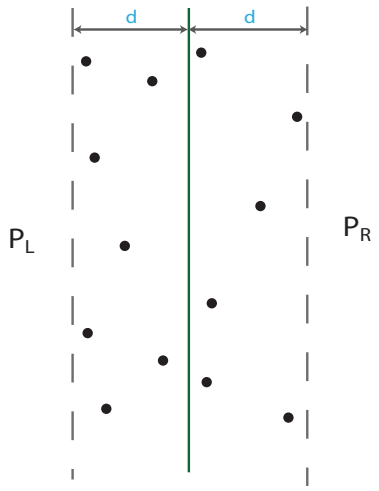
Obr. 4.12. rozdelíme rovinu



Obr. 4.13. nájdeme riešenie v každej časti

V prvom kroku nájdeme priamku, ktorá rozdelí rovinu na dve polroviny, v ktorej bude rovnaký počet prvkov, prípadne v jednej z nich bude o jeden prvok viac. Získame takto dve množiny:

$P_L = \{x_i, i \leq n/2\}$
$P_R = \{x_j, j > n/2\}$



Obr. 4.14. okolie šírky d

V každej z množín nájdeme najbližšiu dvojicu ak je počet prvkov v množine 2 alebo 3. Vzdialenosti najbližších dvojíc označme d_l, d_r. Náš výsledok bude $d = min(d_l, d_r)$.

Ak je v množine viac ako 3 body, budeme ju deliť.

Dôležitý je krok v ktorom budeme jednotlivé časti roviny spájať.

Musíme rozhodnúť, či existuje dvojica p, q, taká, že $p \in P_L, q \in P_R$ a $|pq| < d$.

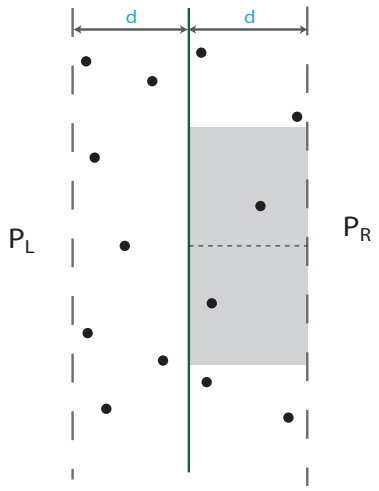
Koľko bodov musíme otestovať na danú podmienku? Ak by sme testovali každý bod z každým, viedlo by to k zložitosti $\mathcal{O}(n^2)$. Tomuto sa chceme vyhnúť.

Nech Q_1, Q_2 sú oblasti šírky d po oboch stranách deliacej priamky (pozri obr. 4.14).

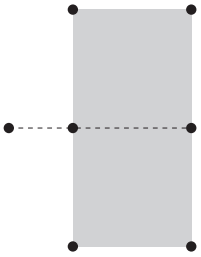
Testovať musíme iba body, ktoré sú vnútri týchto oblastí. Môže však nastať prípad, že všetky body z P_L budú v Q_1 a rovnako aj na druhej strane, čo nás opäť privádza ku kvadratickej zložitosti. Pre jeden bod z Q_1 však nemusíme testovať celú oblasť Q_2, stačí sa nám pozrieť len na obdĺžnik $d \times 2d$ (pozri obr 4.15).

Koľko bodov bude vnútri obdĺžnika? Keďže žiadne dva body nemôžu byť bližšie ako d, v obdĺžniku bude najviac 6 bodov (pozri obr 4.16). Zložitosť tohto kroku je $6 \cdot \mathcal{O}(n/2) \in \mathcal{O}(n)$. Potrebujeme však vedieť ktorých 6 bodov to bude. Budú to tie body, ktoré sú po priemete na deliacu priamku vo vzdialenosti menšej ako d od priemetu bodu z Q_1. Zaujímajú nás teda len y-ové súradnice. Všetky tieto body vieme získať v čase $\mathcal{O}(n)$.

Takýmto algoritmom vieme nájsť najbližšiu dvojicu bodov v rovine v čase $\mathcal{O}(n \log n)$. □



Obr. 4.15. testujeme len body vnútri obdĺžnika

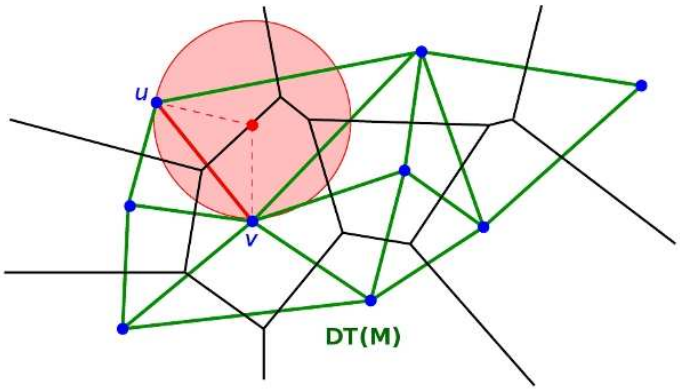


Obr. 4.16. v obdĺžniku je najviac 6 bodov

Cvičenie 4.5 (25 bodov). *Dokážte, alebo vyvráťte tvrdenie: „hrana u,v patrí do Delunayovej triangulácie množiny M práve vtedy, ak existuje kruh, ktorý okrem bodov u,v neobsahuje žiadne iné body množiny M". Uvažujte ako je to v prípade, keď nepripúšťame kocirkulárne štvorice bodov v množine generátorov.*

Riešenie (Marian Maričák, 13.11.2008): Dokážeme dve implikácie (uvažujeme pre $|M| \geq 3$).

$\Leftarrow$: Majme prázdny kruh so stredom x (označme ho $C(x)$), na ktorého hranici ležia u a v. Chceme ukázať, že $(u, v) \in DT(\mathbb{M})$. Všimnime si, že x leží v prieniku Voronoiových oblastí $V(u)$ a $V(v)$, teda



Obr. 4.17. Prazdny kruh so stredom na hrane Voroniovho diagramu duálnej ku hrane Delaunayovej triangulacie.

$x \in V(u) \cap V(v)$. Označme $V(u) \cap V(v)$ ako e. Keďže na hranici $C(x)$ okrem u a v neležia žiadne body, môžeme posunúť x o malý kúsok pozdĺž e, pričom $C(x)$ bude stále obsahovať iba u a v. Z toho vyplýva, že x leží na Voronoiovej hrane medzi oblasťami $V(u)$ a $V(v)$, a teda $(u, v) \in DT(\mathbb{M})$.

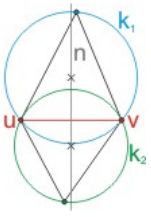
$\Rightarrow$: Ak sú dva vrcholy susedné (je medzi nimi hrana) v Delaunayovej triangulácii, potom sú susedné aj ich oblasti vo Voronoiovom diagrame. Takže ako stred nášho kruhu môžeme zobrať ľubovoľný bod x vo vnútri hrany medzi týmito oblasťami. Takto bude mať x rovnakú vzdialenosť r od u aj od v. Potom kruh s polomerom r a stredom x nebude obsahovať žiadne iné body okrem u a v (pozri obr. 4.17).

□

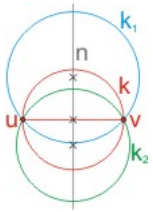
Riešenie (Marta Režnáková, 15.11.2008):

(1) Dokážme tvrdenie: Ak patrí hrana u, v do Delunayovej triangulácie množiny M, tak existuje kružnica, ktorá okrem bodov u, v neobsahuje žiadne iné body množiny M.

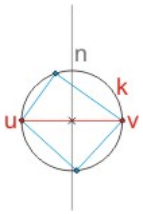
To, že hrana u, v patrí do Delunayovej triangulácie znamená, že existuje kružnica opísaná trojuholníku, ktorého stranou je práve hrana u, v taká, že obsahuje iba vrcholy daného trojuholníka.



Vezmime si teda hranu u, v. Vieme, že existujú kružnice k_1 a k_2 (v prípade, že hrana u, v bude ležať na hranici triangulácie, počítame iba s jednou kružnicou) také, že okrem hrany u, v obsahuje každá už len jeden bod. Pre každú kružnicu opísanú hrane u, v pritom platí, že jej stred leží na normále n hrany u, v prechádzajúcej jej stredom. Ak chceme teda nájsť takú kružnicu k, ktorá bude obsahovať iba hranu u, v, jej stred bude ležať na normále n ohraničenej stredmi kružníc k_1 a k_2. Najmenšia takáto kružnica je taká, ktorej stred leží priamo na hrane u, v.



Ak by teda pre obe kružnice k_1 a k_2 v Delunayovej triangulácii ležal ich stred na hrane u, v, nedokázali by sme už zostrojiť takú kružnicu k, ktorá by okrem hrany u, v už žiadne iné útvary neobsahovala, no zároveň by sme porušili podmienku o nekocirkularite bodov.

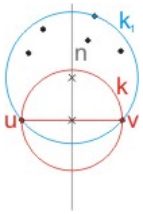


Znamená to teda, že ak hrana u, v patrí do Delunayovej triangulácie, vieme nájsť takú kružnicu k, ktorá okrem bodov u, v neobsahuje žiadne iné body množiny M.

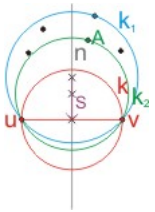
(2) Dokážme tvrdenie: Ak existuje kružnica, ktorá okrem bodov u, v neobsahuje žiadne iné body množiny M, tak hrana u, v patrí do Delunayovej triangulácie množiny M.

Ukázali sme si už, že najmenšia kružnica k taká, že okrem bodov u, v neobsahuje žiadne iné body, je kružnica so stredom v strede hrany u, v. Cheme teraz dokázať, že existuje taká kružnica, ktorá okrem bodov u, v obsahuje práve jeden bod z množiny M.

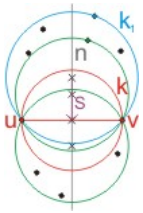
Predpokladajme, že vieme nájť takú kružnicu k_1, ktorá by okrem bodov u, v obsahovala aspoň dva ďalšie body z množiny M. Existujú dve možnosti usporiadania týchto bodov (okrem bodu, ktorý by spolu s bodmi u, v tvoril trojuholník), a síce body, ktoré by ležali vnútri kružnice, alebo body na kružnici. Keďže ide o generátory a je daná podmienka o nekocirkularite generátorov, ostáva nám iba prvý prípad.



Ak bod z množiny M leží vnútri kružnice k_1, vieme nájsť takú kružnicu k_2 s polomerom menším ako k_1, ktorej stred bude ležať na normále n posunutý v smere vektora $s =$ (stred hrany u, v - stred kružnice k_1). Vieme, že kružnica k_2 už body ležiace priamo na kružnici k_1 (okrem bodov u, v) neobsahuje. Znamená to teda, že ak by kružnica k_1 obsahovala bod A, ktorý by nebol priamo na nej, vieme nájsť takú kružnicu k_2, ktorá by prechádzala bodmi u, v a bodom A. V prípade, že by kružnica k_1 obsahovala viac bodov, vieme sa postupne dopracovať k takej kružnici, ktorá bude obsahovať už len jeden bod.



Vieme, že existuje kružnica k, ktorá obsahuje iba body u, v. Táto kružnica je tiež jednou z kružníc k_2, ktoré získame posunutím. Inými slovami, časť kružníc k_1 a k_2 oddelených nadrovinou $\overline{u,v}$, na opačnej strane ako stredy kružníc, sú menšie ako takáto časť kružnice k.



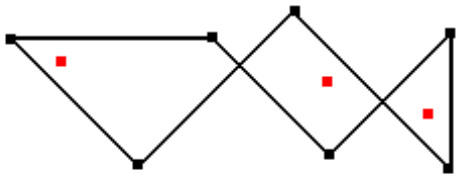
Ak teda budeme posúvať kružnicu k_2, na opačnej stranehrany u,v žiadne body nebudú kým by sme týmto spôsobom nedostali kružnicu k. Pri ďalšom posunutí by sme teda riešili už prípad na opačnej strane hrany u, v.

□

Cvičenie 4.6 (50 bodov). *Dokážte, alebo vyvráťte tvrdenie: "Na stráženie ľubovoľného n-uholníka stačí $\frac{n}{3}$ strážcov."*

Riešenie (Jana Hlinková, 14.11.2008):

Tvrdenie platí iba pre jednoduché mnohouholníky. Obrázok 4.18 znázorňuje príklad "nejednoduchého"n-uholníka, na stráženie ktorého



OBR. 4.18. Ukážka nejednoduchého mnohouholníka, na ktorého stráženie treba viac ako $\lfloor \frac{n}{3} \rfloor$ strážcov.

je potrebných viac ako $\lfloor \frac{n}{3} \rfloor$ strážcov. Ďalej teda pojmom n-uholník budem označovať iba jednoduché mnohouholníky.

Problém stráženia je známy ako problém galérie (Art Gallery Problem). Victor Klee ho formuloval nasledovne: "Nech pôdorys galérie tvorí mnohouholník s n vrcholmi (/hranami). Aký je minimálny počet strážcov (vyjadrený ako funkcia n-ka), ktorý stačí na stráženie galérie?"

Teda hľadáme minimálny počet strážcov, ktorí ustrážia hocijaký mnohouholník s n vrcholmi. Strážcovia sú reprezentovaní bodmi a vidia do 360° okolo seba. Ustráženie mnohouholníka znamená, že každý bod vnútra mnohouholníka je viditeľný aspoň pre jedného strážcu. Bod x je viditeľný pre strážcu (bod) y, ak vnútro úsečky, ktorá ich spája, leží celé vo vnútri daného mnohouholníka.

O probléme galérie (a teda aj dôkaze tvrdenia) som čítala skôr, ako som sa rozhodla tento príklad riešiť, preto sa nebudem snažiť vymyslieť vlastný dôkaz, ale uvediem dôkaz podľa Steva Fiska (z poznámok *Design and Analysis of Computer Algorithms, David M. Mount*).

V dôkaze sa využíva triangulácia a farbenie grafov. Preto najprv napíšem niekoľko pomocných tvrdení.

DEFINÍCIA 4.1. *Nech P je n-uholník. Triangluláciou n-uholníka P je taká planárna subdivízia P, ktorej vrcholy sú vrcholmi P a všetky steny sú trojuholníky. (A všetky hrany sú buď hranami P alebo diagonálami P, hovoríme tomu aj simpliciálny komplex).*

LEMA 4.1. *Každý n-uholník má triangluláciu s $n-3$ diagonálami a $n-2$ trojuholníkmi.*

Toto tvrdenie sa dá jednoducho dokázať matematickou indukciou. Nech $n = 3$, potom tvrdenie triviálne platí (triangluláciou trojuholníka je jeden trojuholník s nula diagonálami). Nech teda $n \geq 4$. Takýto n-uholník má diagonálu. (Toto je netriviálne topologické tvrdenie, ktorého dôkaz presahuje rámec takéhoto príkladu.) Táto diagonála rozdelí n-uholník na dva menšie mnohouholníky, jeden s n_1 a druhý s n_2 vrcholmi, pričom $n_1 + n_2 = n + 2$, lebo vrcholy diagonály obsahujú oba

mnohouholníky. Tieto mnohouholníky majú triangluláciu s $n_1 - 2$ a $n_2 - 2$ trojuholníkmi. Obe triangulácie sú disjunktné, teda triangulácia pôvodného n-uholníka má $(n_1 - 2) + (n_2 - 2) = n_1 + n_2 - 4 = n + 2 - 4 = n - 2$ trojuholníkov. Podobne sa dá dokázať aj tvrdenie o počte diagonál.

LEMA 4.2. *Triangulácia n-uholníka je planárny graf. (Vrcholmi grafu sú vrcholy n-uholníka, hranami grafu sú hrany triangulácie n-uholníka.)*

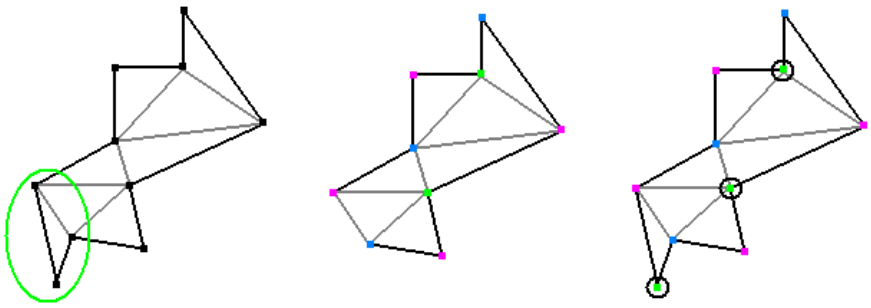
Jednoduchý n-uholník je planárny graf. Triangluláciou pridávame iba nepretínajúce sa diagonály, a teda graf ostáva planárny.

LEMA 4.3. *Planárny graf má vrcholové štvorfarbenie.*

Toto tvrdenie nechám bez dôkazu (dôkaz sa dá nájsť v textoch z teórie grafov). Dôkaz tohoto tvrdenia je náročný. Bol známym motorom rozvoja teórie grafov.

LEMA 4.4. *Graf zodpovedajúci triangulácii n-uholníka vieme zafarbiť 3 farbami.*

Tvrdenie sa dá dokázať pomocou indukcie. Trojuholník ($n = 3$) má očividne trojfarbenie. Nech teda $n \geq 4$. Triangulácia n-uholníka (ktorý je jednoduchý mnohouholník bez dier) obsahuje aspoň jeden trojuholník, ktorého dve hrany sú hranami hranice tohto n-uholníka. Keď takýto trojuholník odoberieme, dostaneme triangluláciu $(n-1)$-uholníka, ktorá podľa indukčného predpokladu má trojfarbenie. Vrátime odobraný trojuholník naspäť. Dva jeho vrcholy sú už ofarbené (lebo boli súčasťou už ofarbeného $(n-1)$-uholníka). Na ofarbenie tretieho vrcholu použijeme zvyšnú tretiu farbu. Takto sme pôvodný n-uholník ofarbili tromi farbami. (Obrázok 4.19.)



OBR. 4.19. Triangulácia n-uholníka s označeným odobrateľným trojuholníkom. Trojfarbenie $(n-1)$-uholníka. Pridanie odobraného trojuholníka a zvolenie strážcov.

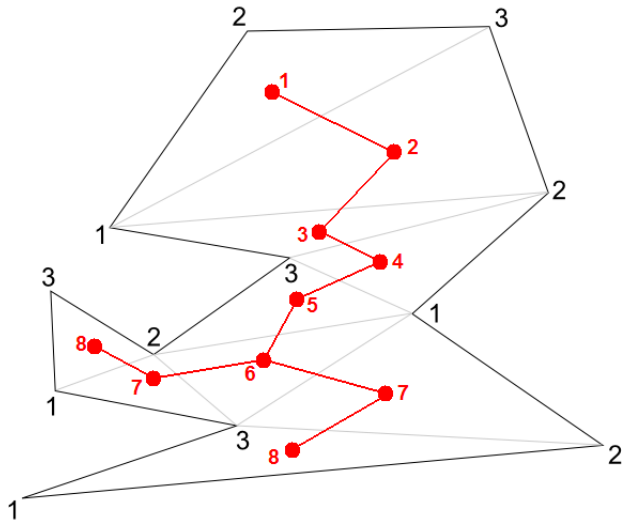
LEMA 4.5. *V trojfarbení grafu triangulácie n-uholníka existuje farba, ktorou je ofarbených najviac $\lfloor \frac{n}{3} \rfloor$ vrcholov.*

Ak by bolo každou farbou ofarbených aspoň $\frac{n}{3}+1$ vrcholov, po sčítaní by sme dostali viac ako n vrcholov, ale graf má n vrcholov, a tak by sme prišli k sporu.

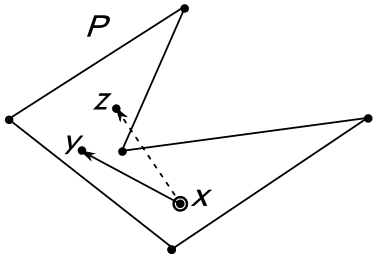
Teraz už môžeme jednoducho ukázať, že na stráženie n-uholníka stači $\lfloor \frac{n}{3} \rfloor$ strážcov. Za strážcov zvolíme vrcholy ofarbené farbou, ktorou je ofarbených najviac $\lfloor \frac{n}{3} \rfloor$ vrcholov (z predchádzajúceho tvrdenia vieme, že takáto farba existuje). V každom trojuholníku triangulácie je vrchol ofarbený touto farbou (lebo na ofarbenie trojuholníka potrebujeme všetky tri farby). Vrchol trojuholníka vidí na všetky body tohto trojuholníka. A teda strážcovia spolu vidia na všetky body v n-uholníku a je ich najviac $\lfloor \frac{n}{3} \rfloor$. □

Riešenie (Pavol Beluško, 18.11.2008): Tvrdenie platí pre jednoduchý mnohouholník. V nasledujúcom texte budem pod pojmom mnohouholník chápať práve takýto mnohouholník. Dôkaz:

(1) Zostrojme trianguláciu mnohouholníka (konštrukčný dôkaz jej existencie je v cvičení 5.7).
(2) Trianguláciu možno považovať za planárny graf, ktorého vrcholy sú vrcholy n-uholníka a hrany sú strany trojuholníkov. Tento graf zafarbime troma farbami. Ukážeme, že takéto ofarbenie je možné uskutočniť. Zostrojme duálny graf triangulácie, pričom vynechajme vonkajšiu oblasť. Tento graf je strom. Neobsahuje cyklus (pretože potom by mnohouholník obsahoval dieru) a je súvislý (všetky trojuholníky sú prepojené). Vezmime jeden trojuholník triangulácie. Ofarbime jeho vrcholy. Tento trojuholník je koreňom stromu. Farbíme prechádzaním stromu do šírky. V našom strome otec indukuje ofarbenie syna, pretože otec je ofarbený skôr ako syn a zdieľajú spoločnú hranu (tj. 2 vrcholy z 3 už sú ofarbené). Postupujeme smerom k ešte neofarbeným trojuholníkom, takže môžme ofarbovať. Keďže strom nemá cyklus, nestane sa ani, že by sa stretli dve vetvy ofarbovania, ktoré by vyústili v konflikt v ofarbovaní (napr. dva rôzne trojuholníky by indukovali rôzne ofarbenie ich spoločného suseda). Ofarbenie triangulácie tromi farbami je teda možné zrealizovať.
(3) Najzriedkavejšia farba sa nevyskytuje viac než $\lfloor n/3 \rfloor$-krát. Ak by tomu tak nebolo, každá farba by sa vyskytovala aspoň $(\lfloor n/3 \rfloor + 1)$-krát a $(3 \times (\lfloor n/3 \rfloor + 1)) > n$, teda nemali by sme dostatok vrcholov.
(4) Do vrcholov označených najzriedkavejšou farbou osaďme strážnikov. Takýto vrchol (strážnik) vidí všetky vrcholy (aspoň jedného) trojuholníka, ktorého je vrcholom. Každý trojuholník má vrcholy ofarbené všetkými 3 farbami, čiže aj najzriedkavejšou, strážnici spolu teda vidia všetky vrcholy mnohouholníka.



Obr. 4.20. Ofarbená triangulácia a duálny graf s očíslovanými vrcholmi podľa ich hĺbky



Obr. 4.21. Bod y je priamo viditeľný z bodu x, bod z už nie je.

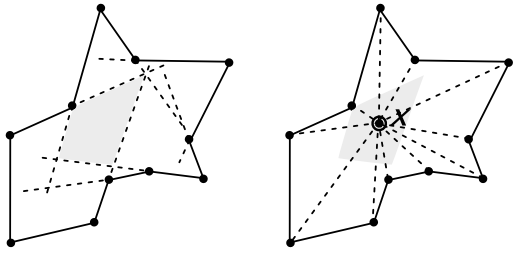
Z toho vyplýva, že na stráženie ľubovoľného n-uholníka naozaj stačí $n/3$ strážcov.

Je vhodné ešte ukázať, či na stráženie vo všeobecnosti nestačí menej než $n/3$ strážnikov. Toto však možno vyvrátiť na triviálnom príklade pre $n = 3$. $n/3 = 1$, ale menej ako 1 strážnik samozrejme nestačí. Ľahko sa dá nájsť aj komplikovanejší kontrapríklad pre väčšie n. □

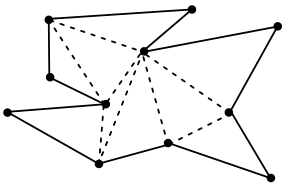
Riešenie (Elena Dušková, 20.11.2008):

Uvažujeme n-uholník bez samopriesekov a dier. Naši strážcovia sú nehybný a majú schopnosť súčasne vidieť 360° okolie, no ich pohľad je ohraničený stenami miestnosti (hranami n-uholníka). Čiže strážca má pod dohľadom všetky body, na ktoré má priamu viditeľnosť (obr. 4.21).

Ľubovoľný konvexný alebo hviezdicový mnohouholník (obr. 4.22) sa dá „ustriehnuť“ jediným strážcom, lebo celé vnútro je viditeľné z jedného bodu (takýchto bodov môže byť viac -– pri konvexných n-uholníkoch sú to všetky body polygónu, pri hviezdicových je to prienik



Obr. 4.22. Hviezdicový polygón s oblasťou, do ktorej treba umiestniť strážcu, aby ustrážil celý polygón.



Obr. 4.23. Jedna z možných triangulácií polygónu P.

vnútorných polrovín). Trojuholník (najmenší konvexný mnohouholník) presne spĺňa hranicu zo zadania ($n = 3$, počet strážcov = 1). Pri konvexných n-uholníkoch ($n > 3$) alebo hviezdicových mnohouholníkoch je pre n hrán takisto postačujúci 1 strážca, čo určite spĺňa podmienku jednej tretiny.

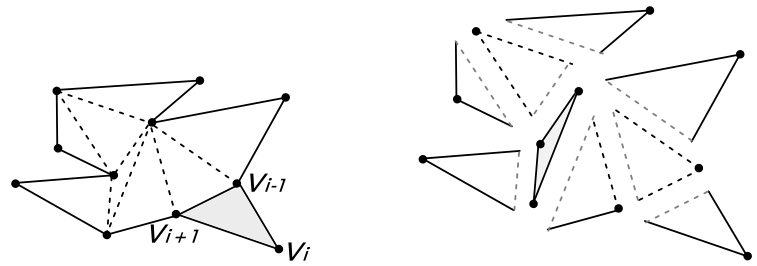
Všeobecné (nekonvexné) mnohouholníky najskôr ľubovoľne (korektne, bez priesekov atď.) triangulujeme. Podľa cvičenia 4.3 sa to dá v zložitosti $\mathcal{O}(n \log n)$.

Keď triangulujeme mnohouholník, tak hrany pôvodného polygónu patria triangulácii. Okrem toho je triangulácia tvorená ešte tzv. vnútornými uhlopriečkami (obr. 4.23). Môžeme triangulovať rekurzívne (čo asi nie je optimálny spôsob, ale dáva korektný výsledok).

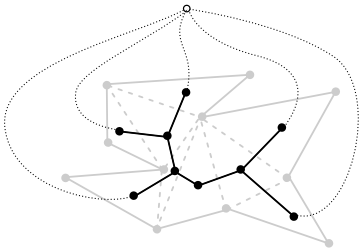
V n-uholníku vyberieme dve susedné hrany ($V_{i-1}V_i$ a V_iV_{i+1}), ktorých vnútorný uhol je menší ako 180 stupňov (taký uhol určite existuje v každom n-uholníku) a zároveň je z bodu V_{i-1} do bodu V_{i+1} priama viditeľnosť. Predpokladáme, že existuje aspoň jeden takýto úsek. Ak by takýto úsek neexistoval, nemali by sme v očakávanej triangulácii ani jeden trojuholník, ktorý má 2 hrany z pôvodného polygónu. Všetky trojuholníky by boli vytvorené 3 alebo 2 vnútornými uhlopriečkami a 0 alebo 1 hranou polygónu. Takýmto spôsobom ale nevieme zostrojiť trianguláciu polygónu bez dier.

Dostali sme, že úsek $V_{i-1}V_iV_{i+1}$ existuje. Ak je ich viac, vyberieme si ľubovoľný z nich.

Medzi vrcholmi V_{i-1} a V_{i+1} vytvoríme hranu triangulácie. Polygón tak rozdelíme na trojuholník a $(n-1)$-uholník (obr. 4.24, vľavo). Ďalej oddeľujeme trojuholníky rekurzívne od $(n-1)$-uholníka, až kým neprídeme k rozdeleniu na samé trojuholníky (obr. 4.24, vpravo).



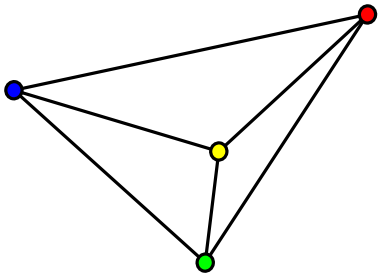
Obr. 4.24. Vľavo: n-uholník rozdelíme na trojuholník a $(n-1)$-uholník, ktorý ďalej triangulujeme. Vpravo: Časti triangulácie polygónu, posledný trojuholník je zvýraznený.



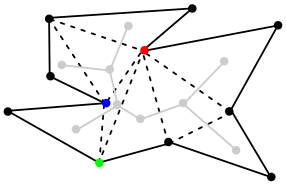
Obr. 4.25. Duálny graf ku triangulácii.

Takýmto spôsobom nám v triangulácii medzi počtom hrán (E), počtom trojuholníkov (F) a počtom vrcholov (V) vznikne určitý vzťah. Pri každom vytvorení $(n-1)$-uholníka nám na spracovanie zostane „o 2 hrany a 1 trojuholník menej“ až nakoniec zostanú 3 hrany a 1 trojuholník. Keď postupujeme opačne a od trojuholníka budujeme n-uholník, tak na začiatku máme 3 hrany (E), 1 trojuholník (F) a 3 vrcholy (V). K tomu v každom kroku pribudnú 2 hrany (E), 1 trojuholník (F) a 1 vrchol (V) (obr. 4.24, vpravo). Po k krokoch máme $E = 3 + 2k$, $F = 1 + k$, $V = 3 + k$ pre $k = 0, \ldots, (n-3)$. Čiže pre trianguláciu n-uholníka platí $E : F : V = (3+2(n-3)) : (1+(n-3)) : (3+(n-3)) = (2n-3) : (n-2) : (n)$. Celkový počet hrán je $2n-3$, z toho n hrán patrí hranici polygónu a $n-3$ je vnútorných uhlopriečok. Tie oddeľujú $n-2$ trojuholníkov. Duálna štruktúra k tomuto grafu (bez vrcholu a hrán z oblasti mimo polygónu) je strom s $n-2$ vrcholmi a $n-3$ hranami (obr. 4.25) -– neexistuje tam cyklus, čo bude dôležité pri ofarbovaní. Koreňom stromu bude posledný trojuholník z triangulovania.

Z teórie grafov je známe, že na ofarbenie ľubovoľného rovinného grafu potrebujeme najviac 4 farby. Príklad jednoduchého planárneho grafu (triangulácie), ktorý má 4 vrcholy, 6 hrán a 3 trojuholníky a na jeho ofarbenie potrebujeme 4 farby je na obr. 4.26. Tento graf však nie je trianguláciou mnohouholníka, neplatí preň vyššie uvedený vzťah. Pri triangulácii mnohouholníka sa nám nebudú vyskytovať prípady, ktoré si vyžadujú 4 farby. Vyplýva to zo spôsobu, akým sme vytvárali



Obr. 4.26. Na ofarbenie tohto planárneho grafu sú potrebné 4 farby.



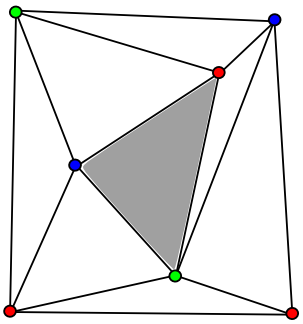
Obr. 4.27. Rozširovanie ofarbenia.

trianguláciu (viď zdôvodnenie). Vrcholy triangulácie nášho polygónu P budeme môcť ofarbiť tromi farbami (napr. červená, modrá, zelená) tak, že žiadny trojuholník nebude mať 2 (alebo 3) vrcholy rovnakej farby.

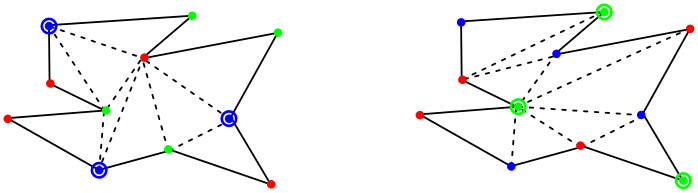
Zdôvodnenie: Pri ofarbovaní postupujeme opačne ako pri konštrukcii triangulácie a začneme od trojuholníka, ktorý sme odrezali ako posledný -– koreň stromu. V ňom zafarbíme každý vrchol jednou z troch farieb (obr. 4.27). Keď už máme trojuholník zafarbený 3 farbami (R, G, B) a chceme zafarbiť susedný trojuholník (hrana a dva vrcholy s určenou farbou sú spoločné), tak máme len jednu možnosť pre farbu tretieho vrcholu druhého trojuholníka. Takto budeme rozširovať ofarbenie na zvyšné vrcholy triangulácie postupovaním po jednotlivých vetvách stromu. Pri rozširovaní ofarbenia na susedné trojuholníky pri všeobecnom grafe by mohlo dôjsť ku kolízii (napr. obr. 4.28 (triangulácia mnohouholníka s dierou), lebo k jednému miestu vieme dôjsť viacerými cestami, v časti duálneho grafu (bez vrcholu a hrán mimo polygónu) sa vyskytol cyklus. Pri stromoch cykly nemáme, a preto ku kolízii prísť nemôže.

Rozširovanie ofarbenia od koreňa stromu nás doviedlo k jednoznačnému ofarbeniu celého mnohouholníka. Vo všeobecnosti nezáleží na poradí, v ktorom trojuholníky ofarbujeme, ktorý vrchol prehlásime za koreň nášho stromu. Takže môžeme začať aj iným trojuholníkom. Pretože pri triangulácii jednoduchého mnohouholníka bez dier nemáme cykly, tak dospejeme vždy k rovnakému riešeniu až na zámenu farieb ($3! = 6$ možností).

Každý trojuholník v P má už vrchol jednej z troch farieb (červenej, modrej, zelenej). Bod v trojuholníku je určite strážený, ak jeden

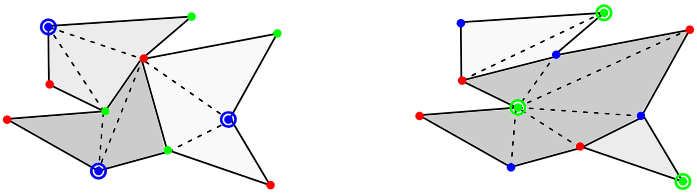


Obr. 4.28. Triangulácia mnohouholníka s dierou sa nedá ofarbiť 3 farbami, na niektorej hrane dôjde ku kolízii farieb.



(a) (b)

Obr. 4.29. Ofarbenie polygónu P v závislosti od triangulácie.

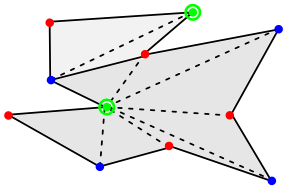


Obr. 4.30. Výsledná pozícia strážcov v závislosti od triangulácie (vľavo spôsob a), vpravo spôsob b)) a oblasti, na ktoré daný strážca určite dohliada (t.j. trojuholníky, do ktorých patrí).

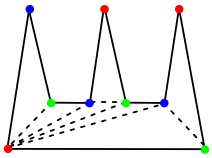
z vrcholov trojuholníka je strážca. Z 3 farieb si zvolíme najmenej-krát použitú (v prípade rovnosti ľubovoľnú) a našich strážcov postavíme na vrcholy tejto farby (obr. 4.29).

V prípade triangulácie a) máme 4 červené, 4 zelené a 3 modré vrcholy. Aby sme zaistili, že každý trojuholník bude pod dohľadom a použijeme čo najmenej vrcholov, tak strážcov postavíme na modré vrcholy (obr. 4.30). Pri triangulácii b) máme najmenej vrcholov zafarbených na zeleno, ale sú inak rozmiestnené ako v prípade a), takže tu budú strážcovia stáť na troch zelených vrcholoch.

Máme n vrcholov a 3 farby, podľa Dirichletovho princípu vieme, že jednou z troch farieb budú vrcholy ofarbené nanajvýš $\lfloor \frac{n}{3} \rfloor$-krát. Túto



Obr. 4.31. Dvaja strážcovia pokryjú celý polygón.



Obr. 4.32. Príklad polygónu dosahujúceho hranicu $\frac{n}{3}$ strážcov.

farbu si zvolíme, a teda budeme mať nanajvýš $\lfloor \frac{n}{3} \rfloor$ strážcov. V príklade sme mali 11 vrcholov a vytvorili sme 3 strážcov, $\lfloor \frac{11}{3} \rfloor = 3$.

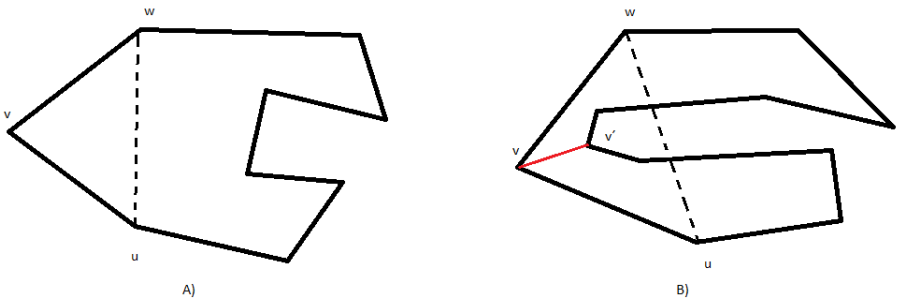
Tento postup nedáva optimálne rozloženie strážcov v polygóne, ale ukazuje aká je horná hranica problému. Aj v predchádzajúcom príklade sa dá nájsť rozmiestnenie, pri ktorom postačuje menší počet strážcov (obr. 4.31).

Existujú polygóny, ktoré si vyžadujú minimálne $\lfloor \frac{n}{3} \rfloor$ strážcov (obr. 4.32), čiže táto hranica sa určite nedá posunúť nižšie, t.j. $\lfloor \frac{n}{3} \rfloor$ je pre niektoré prípady nevyhnutný počet a pre všetky ostatné je dostačujúci.

□

Riešenie (Kiss Gábor, 28.11.2010): Majme jednoduchý mnohouholník P (teda bez dier a pretínajúcich sa strán). Vzhľadom na to, že tento mnohouholník môže byť veľmi komplikovaný, rozdeľme si ho na časti, ktoré je možné jednoducho strážiť- v našom prípade sú to trojuholníky. Mnohouholník si rozdeľme na trojuholníky pridaním nepretínajúcich sa diagonál- teda spojením vrcholov mnohouholníka, ktoré ešte nie sú spojené a ich spojnica je vo vnútri mnohouholníka. Rozdelenie P pomocou takýchto diagonál na trojuholníky budeme nazývať triangulóciou P. Mali by sme si ukázať, že každý mnohouholník je triangulovateľný.

Pre mnohouholník s troma vrcholmi je dôkaz jednoznačný. Predpokladajme teda, že je triangulovateľný mnohouholník s m vrcholmi a teraz sa to pokúsime dokázať pre $n > m$. Mnohouholník s n vrcholmi si označme P. Najprv si teda ukážme , že existuje uhlopriečka v mnohouholníku. Vyberme si najľavejší vrchol z P. Tento vrchol susedí s vrcholmi w a u. Ak celá úsečka wu patrí mnohouholníku P, tak je jeho uhlopriečkou (obr. 4.33, A). Ak ale wu pretína hranicu P, tak určite existuje aspoň jeden bod v trojuholníku wuv. Z bodov, ktoré ležia v



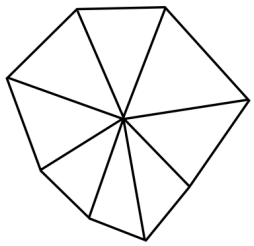
Obr. 4.33. Vnútorná uhlopriečka v mnohouholníku

trojuholníku wuv vyberme najvzdialenejší od úsečky wu. Označme ho $v´$. Priamka $vv´$ určite nepretína hranicu P, a teda je uhlopriečkou (obr. 4.33, B). Touto uhlopriečkou sa mnohouholník rozdelil na dve časti, pre ktoré platí, že obe majú určite menší počet vrcholov, ako mal pôvodný mnohouholník. A pre mnohouholníky s menším počtom vrcholov, ako n sme už dokázali že sú triangulovateľné, teda celý mnohouholník je triangulovateľný. Teda každý jednoduchý mnohouholník je triangulovateľný.

Následne si dokážme aj to, že triangulácia obsahuje $n-2$ trojuholníkov. Vezmime si teda ľubovoľnú diagonálu P v niektorej z triangulácií P. Táto diagonála nám rozdelí P na mnohouholníky P_1 a P_2 s m_1 a m_2 vrcholmi. Každý vrchol P patrí práve jednému z P_1 alebo P_2 okrem vrcholov na danej diagonále. Tie body patria obom. Teda $m_1+m_2 = n+2$. Pomocou indukcie je teda možné dokázať, že akákoľvek triangulácia P_i pozostáva z m_i-2 trojuholníkov. Teda triangulácia celého P pozostáva z $(m_1-2)+(m_2-2)$ trojuholníkov, a to sa rovná $n-2$.

Teda galériu je možné strážiť pomocou $n-2$ kamier, ak by sa každá nachádzala v jednom z trojuholníkov. Ale ak by sme kamery rozložili na správne vybrané diagonály podarilo by sa nám ich počet znížiť na $\frac{n}{2}$, lebo by každá kamera dokázala strážiť oba trojuholníky, ktorých spoločnou hranou je daná diagonála. Pomocou podobnej úvahy jednoducho dospejeme k tomu, že ešte výhodnejšie by bolo kamery umiestniť do vrcholu trojuholníka. Z tejto pozície by bolo možné strážiť všetky trojuholníky, ktorým tento vrchol patrí. Skúsme teda vybrať čo najmenšiu množinu vrcholov z P takých, že každý trojuholník bude mať aspoň jeden vrchol v tejto množine. K nájdeniu takej množiny použijeme ofarbenie vrcholov mnohouholníka pomocou 3 farieb tak, aby vrcholy s rovnakou farbou nemali spoločnú ani hranu, ani diagonálu. Vyberme si vrcholy s farbou , ktorá bola použitá pri farbení najmenej krát. Ak do týchto vrcholov umiestnime kamery, tak budeme vedieť sledovať celú galériu.

Poslednou nezodpovedanou otázkou ostáva, či je možné každú triangulóciu zafarbiť troma farbami. Na dôkaz použijeme takzvaný duálny



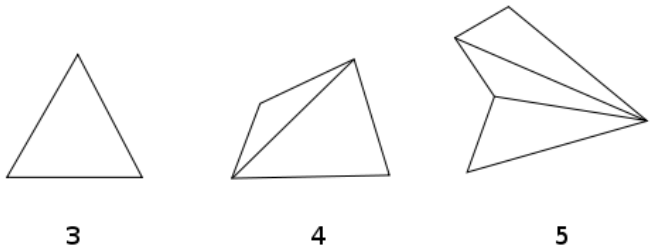
Obr. 4.34. Vejár trojuholníkov

graf k triangulácii. Každý vrchol grafu bude priradený jednému trojuholníku z triangulácie. Vrcholy grafu budú spojené vtedy, ak k nim prislúchajúce trojuholníky majú spoločnú hranu. Keďže platí, že každá diagonála mnohouholníka rozdelí mnohouholník na dve časti (rozrezaním pozdĺž danej diagonály sa rozpadne na dve časti), tak platí aj to, že vynechaním jednej hrany z duálneho grafu sa tento graf rozpadne na dve časti, teda sa jedná o strom (toto neplatí v prípade, že v mnohouholníku je diera). Následne začnime prehľadávať tento strom z ľubovoľného vrcholu. Vrcholom trojuholníka prislúchajúceho k danému vrcholu grafu priradíme farby napríklad modrú, zelenú a červenú farbu – každú farbu použijeme práve raz. Z tohto vrcholu grafu prejdime k jednému z jeho susedov. Trojuholník prislúchajúci k tomuto vrcholu má spoločnú jednu hranu s už zafarbeným trojuholníkom, teda nám ostáva určiť farbu posledného vrchola. Bez ujmy na všeobecnosti môžeme tvrdiť, že vrcholy spoločnej hrany majú napríklad modrú a červenú farbu, teda na nezafarbený vrchol nám ostala len jedna, a to je zelená. Týmto postupom vieme postupne ofarbiť všetky vrcholy P.
□

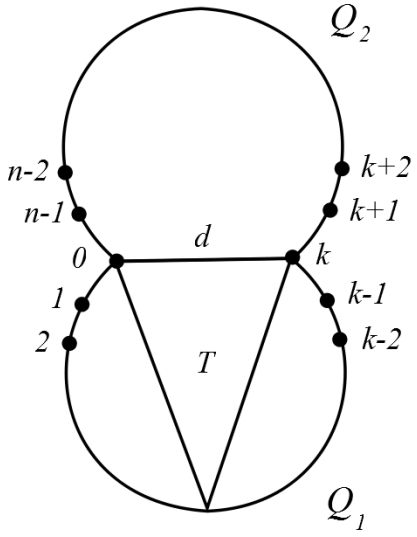
Riešenie (Júlia Kučerová, 28.11.2010): Toto tvrdenie platí iba pre jendoduché polygóny. Na jeho dokázanie použijeme Chvátalov dôkaz. Najprv si definujeme pojem vejár ako trianguláciu s jedným vrcholom (stred vejára), ktorý je spoločný pre všetky trojuholníky (obr. 4.34). Chvátal vyslovil hypotézu: " Každá triangulácia polygónu s n vrcholmi môže byť rozdelená na $g \leq \lfloor \frac{n}{3} \rfloor$ vejárových častí." Dôkaz tejto hypotézy je založený na nasledujúcej leme: *Každá triangulácia polygónu P s n vrcholmi má diagonálu d, ktorá rozdelí P na dva polygóny P_1 a P_2, kde P_1 má 5,6, alebo 7 vrcholov, $n \geq 6$.*

Majme polygón P s $n \geq 3$. Ak $n = 3, 4, 5$, tento polygón je typu vejár. Ako môžme vidieť na obrázku (obr. 4.35) v polygónoch s 3,4, alebo 5 vrcholmi vždy existuje bod typu "stred vejára".

Určime si teda polygón P s $n \geq 6$. Algoritmus je založený na odstránení časti triangulácie, následnom aplikovaní hypotézy a pridaní



Obr. 4.35. Existencia vrcholu vejára v mnohouholníku s 3,4 a 5 vrcholmi



Obr. 4.36. Rozdelenie mnohouholníka vnútornou uhlopriečkou

odstránenej časti naspäť. Aplikujme si navrhnutú hypotézu na polygón Q. Rozdeľme ho diagonálou d na dve časti Q_1 a Q_2. Nech Q_1 je triangulácia oddelená diagonálou d, ktorá má $k+1$ hrán (obr. 3). Nech Q_2 je zyvšok triangulácie zdieľajúci d s Q_1 a nech má $n-k+1$ vrcholov. Podľa hypotézy bude Q_2 rozdelená na $g' = \lfloor \frac{(n-k+1)}{3} \rfloor$ vejárov. Pokiaľ je $k \geq 4$, $g' \leq \lfloor \frac{(n-3)}{3} \rfloor = \lfloor \frac{n}{3} \rfloor - 1$.

Potrebujeme teda ukázať, že pre Q_1 sa pridá iba jedna vejárová časť do celého polygónu. Pri dokazovaní tejto skutočnosti je potrebné analyzovať viacero prípadov.

V nasledujúcich prípadoch budeme mať vrcholy $Q1$ označené $1, \ldots, 6$.

Prípad 1 ($k = 4$)

Q_1 je päťuholník. Už sme zistili, že každý päťuholník je typu vejár (obr. 4.35). Preto Q bol rozdelený na $\lfloor \frac{n}{3} \rfloor - 1 + 1 = \lfloor \frac{n}{3} \rfloor$ vejárových

častí.

Prípad 2 ($k = 5$)
Nech Q_1 je šesťuholník. Uvažujme nad trojuholníkom T z Q_1 ktorý má hranu d, s vrcholom v t. Nemôžme mať t rovné 1, alebo 4, lebo by diagonály (0,t) resp. (5,t) oddelili len 4 vrcholy namiesto predpokladaných 5. Prípady pre $t = 2$ a $t = 3$ sú očividne symetrické, teda v nasledujúcich úvahách predpokladajme , že $t = 2$ bez ujmy na všeobecnosti (obr. 4.37). Švoruholník (2,3,4,5) je možné triangulovať dvoma spôsobmi:

Prípad 2a
Pomocou diagonály (2,4)(obr. 4.37a)). Potom je Q_1 vejár a my sme skončili.
Prípad 2b
Diagonála (3,5) (obr. 4.37b)). Skonštruujme graf Q_0 ako zjednotenie Q_2 a T. Q_0 teda bude mať $n - 5 + 1 + 1 = n - 3$ hrán. Použijme na to indukčný predpoklad, a rozdeľme na $g' = \lfloor \frac{(n-3)}{3} \rfloor = \lfloor \frac{n}{3} \rfloor - 1$ vejárových častí. Nech F je vejár. Teraz musí byť T časťou vejáru v Q_0 a stredom F musí byť jeden z vrcholov T'.

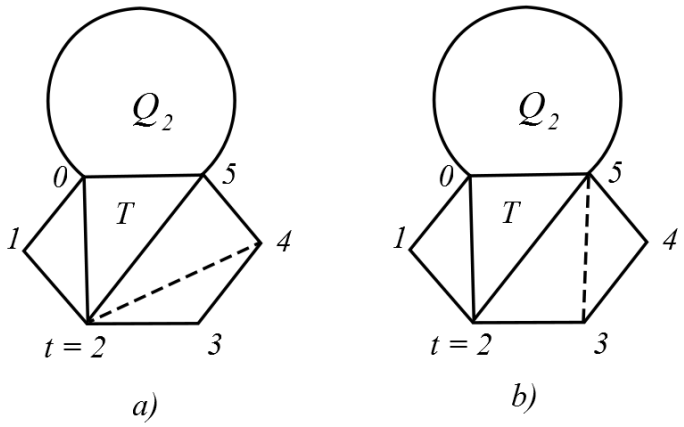
Prípad 2b.1
F je centrované v 0 alebo 2. Potom zlúčme (0,1,2) do F, a spravíme z (2,3,4,5) vejár. Teraz máme všeto v Q pokryté pomocou $\lfloor \frac{n}{3} \rfloor$ vejárových častí.

Prípad 2b.2
F je centrované v 5. Zlúčme teda (2,3,5) a (3,4,5) do F a spravme z (0,1,2) oddelený vejár. Výsledkom je $g' + 1$ vejárových častí.

Prípad 3($k = 6$)
Q_1 je sedemuholník. Vrchol t trojuholníka s hranou d nemôže byť v 1,2,4 alebo 5, kedže potom by existovala diagonála odrezávajúca $4 \leq k < 6$ hrán, čo je v spore s minimalitou k. Nech teda $t = 3$. Každý z (0,1,2,3)a (3,4,5,6) má dve možné triangulácie vedúce k 4 podprípadom. (obr. 5)

Prípad 3a
Diagonály (3,1) a (3,5) (obr 5 a)). Potom Q_1 je vejár centrovaný v 3, a teda sme skončili.

Prípad 3b
Diagonály (0,2) a (3,5) (obr. 5 b)). Spojme štvoruholník (0,2,3,6) s Q_2 do Q_0 s $n - 6 + 1 + 2 = n - 3$ vrcholmi, čo na základe indukčného predpokladu možno rozdeliť na $g' = \lfloor \frac{n}{3} \rfloor - 1$ vejárových častí. Nech



Obr. 4.37. Prípady $k = 5$

F je vejár tohto rozdelenia kam patrí trojuholník (0,2,3). Centrom F musí byť jeden z jeho vrcholov.

Prípad 3b.1
F má stred v 0 alebo 2. Spojme (0,1,2) do F a vytvorme (3,4,5,6) ako oddelenú vejárovú časť.

Prípad 3b.2
F má stred v 3. Spojme (3,4,5,6) do F a vytvorme (0,1,2) ako oddelenú vejárovú časť.

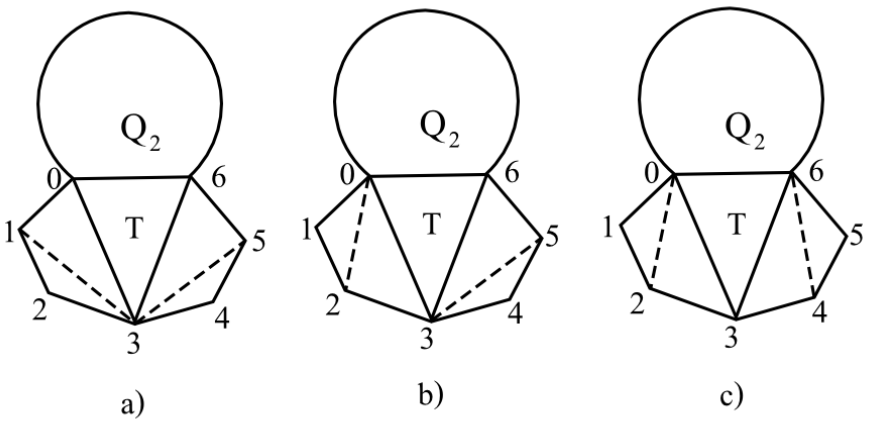
Vo všetkých prípadoch je Q rozdelený do $g' + 1 = \lfloor \frac{n}{3} \rfloor$ častí.

Prípad 3c
Diagonály (1,3) a (4,6) sú prítomné. Toto je zrkadlový obraz prípadu 3b.

Prípad 3d
Diagonály (0,2) a (4,6) (obr. 5 c)). Spojme T s Q_2 a vytvorme polygón Q_0 s $n-6+1+1 = n-4$ vrcholmi. Aplikovaním hypotézy rozdelíme Q_0 na $g' = \lfloor \frac{(n-4)}{3} \rfloor \leq \lfloor \frac{n}{3} \rfloor - 1$ vejárových častí. Nech F je vejárová oblasť obsahujúca T.

Prípad 3d.1
F má stred v 0. Spojme (0,1,2,3) do F a vytvorme (3,4,5,6) ako oddelenú vejárovú časť.

Prípad 3d.2
F má stred v 3. Kým celá časť Q_2 je za diagonálou $d = (0, 6)$ je jasné,



Obr. 4.38. Prípad $k = 6$

že môžme uvažovať F centrované v 0, čo spadá do prípadu 3d.1.

Prípad 3d.3
F má stred v 6. Toto je zrkadlový obraz prípadu 3d.1.

Vo všetkých prípadoch je Q rozdelený do $g' + 1 = \lfloor \frac{n}{3} \rfloor$ častí.

Podľa hypotézy bude vejárových častí maximálne $\lfloor \frac{n}{3} \rfloor$. Keďže každá časť má práve jeden stred vejára bude týchto stredov práve toľko, koľko bude týchto častí. Do každého stredu vejára umiestnime strážcu. Teda počet strážcov je rovnaký ako vejárových častí.

(Zdroj : ART GALLERY THEOREMS AND ALGORITHMS, JOSEPH O'ROURKE, New York, Oxford, OXFORD UNIVERSITY PRESS, 1987) □

Cvičenie 4.7 (35 bodov). *Popíšte transformáciu problému nájdenia Delunayovej triangulácie na problém nájdenia konvexného obalu v 3D, a zdôvodnite, prečo funguje.*

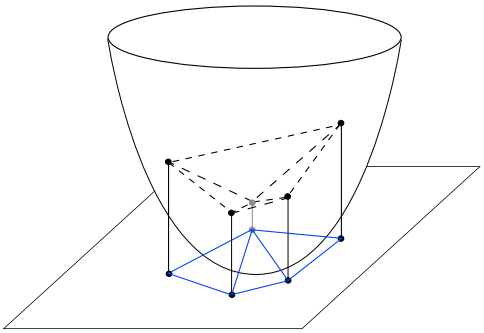
Riešenie (Miroslava Fekiačová, 1.11.2008): Body z množiny P umiestnené na rovine xy premietneme v smere osi z na rotačný paraboloid R s rovnicou $x^2 + y^2 = z$. Dostaneme body P_i^* so súradnicami $[x, y, x^2 + y^2]$.

Označme A_i dotykovú rovinu k paraboloidu v bode P_i^*. Táto rovina má rovnicu $z = 2x_i x + 2y_i y - (x_i^2 + y_i^2)$. Nech $L_{ij}^* = A_i \cap A_j$ a L_{ij} nech je jej kolmý priemet do roviny xy. Rovnicu L_{ij} $2(x_i - x_j)x + 2(y_i - y_j)y - (x_i^2 + y_i^2 - x_j^2 - y_j^2)$ dostaneme elimináciou rovnice z-ovej súradnice z rovníc rovín A_i a A_j. Z rovnice vyplýva, že L_{ij} je osou bodov P_i, P_j.

Vezmime si polpriestor H_j určený rovinou A_j, v ktorom sa nachádza paraboloid R. Priemet časti roviny A_i, ktorá sa nachádza v polpriestore H_j do roviny xy nám vytvorí množinu bodov, ktoré sú bližšie k bodu P_i ako k bodu P_j. Vezmime si teda index i pevne. Voronoiov mnohouholník bodu P_i dostaneme ako ortogonálny priemet množiny $A_i \cap \bigcap_{k=1}^{n} H_k$, kde H_k je polpriestor vytvorený rovinou A_k, ktorý obsahuje paraboloid R.

□

Riešenie (Martin Manduch, 22.11.2009):Máme danú množinu bodov P. Zobrazíme ju na paraboloid zobrazením s predpisom $(x, y) \to (x, y, x^2 + y^2)$. Obraz bodu $\mathbf{x}$ označme $\mathbf{x}^*$ a obraz P označme P^*. Urobíme konvexný obal množiny P^*. Spodnú časť konvexného obalu tvoria tie steny, ktoré pod sebou nemajú iné body konvexného obalu. Spodnú časť konvexného obalu P^* zobrazíme naspäť do roviny xy zobrazením $(x, y, z) \to (x, y)$. Obrazy stien spodnej časti konvexného obalu tvoria Delaunayovu trianguláciu množiny P.



Obr. 4.39. Modrou farbou je znázornená Delaunayova triangulácia. Čiarkovanou čiarou je zobrazený konvexný obal v $\mathbb{E}^3$.

Dotyková rovina k paraboloidu $z = x^2 + y^2$ v bode $(a, b, a^2 + b^2)$ má tvar $z = 2ax + 2by - (a^2 + b^2)$. Posuňme túto rovinu o kladné číslo r^2, dostaneme rovinu s predpisom: $z = 2ax + 2by - (a^2 + b^2) + r^2$. Nájdime prienik takto posunutej roviny s paraboloidom. Porovnaním z súradníc dostaneme rovnosť $x^2 + y^2 = 2ax + 2by - (a^2 + b^2) + r^2$. Z toho jednoduchou úpravou dostaneme $(x - a)^2 + (y - b)^2 = r^2$. Vidíme, že prienik paraboloidu s rovinou zobrazený do xy roviny je kružnica so stredom (a, b) a polomerom r. A tiež je jasné, že body paraboloidu, ktoré sa premietnu do vnútra kružnice, musia ležať na spodnej strane sečnej roviny. Majme tri body $\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*$ z P^* ktoré patria spodnej časti konvexného obalu. Tvoria rovinu, ktorej priesečník s paraboloidom sa zobrazí na opísanú kružnicu trojuholníka $\mathbf{xyz}$. Keďže $\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*$ sú zo spodnej časti konvexného obalu, neleží žiadny vrchol pod nimi a preto sa nezobrazí do vnútra kružnice.

□

Cvičenie 4.8 (20 bodov). *Opíšte algoritmus na vyhľadanie Voronoiovho diagramu pre n bodov v rovine, uveďte jeho pamäťovú a časovú náročnosť.*

Riešenie (Peter Danko, 30.11.2008):

Na zostrojenie Voronoiovho diagramu v rovine použijeme metódu, ktorú zaviedli Shamos a Hoey. Podstatou metódy je paradigma rozdeľuj a panuj (divide and conquer). Pomocou tejto paradigmy vieme zostrojiť Voronoiov diagram v čase $\mathcal{O}(N \log N)$.

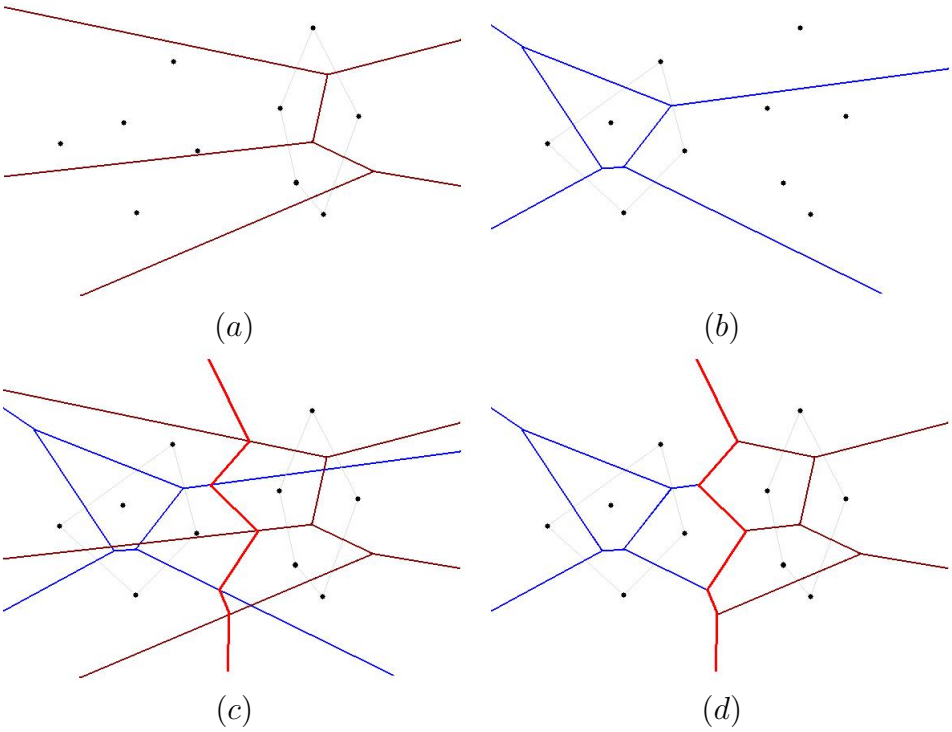
Algoritmus VoronoiDivideAndConquer:
Vstup: množina bodov S (pričom žiadne 4 body nie sú kocirkulárne)
Výstup: Voronoiov diagram Vor(S) množiny S

(1) krok: Ak množina S obsahuje práve 1 bod, potom Voronoiov diagram pre tento bod je celá rovina.
(2) krok: Nájdi vertikálnu priamku m pomocou ktorej rozdelíme S na dve približne rovnako veľké množiny S_L a S_R (S_L leží vľavo do S_R).
(3) krok: Skonštruuj Voronoiove diagramy Vor(S_L) a Vor(S_R) (napríklad Obr.4.40(a)-(b)).
(4) krok: Spoj Vor(S_L) a Vor(S_R) do výsledného Voronoiovho diagramu Vor(S), zohľadni pritom monotónnu (vzhľadom na os y) cestu σ, ktorá oddeľuje S_L a S_R a je rovnako vzdialená od týchto dvoch množín. Ilustrácia cesty σ a spojenie dvoch Voronoiových diagramov je na Obr.4.40(c)-(d).

Ak vieme kroky 1 a 3 vykonať v lineárnom čase, potom dosiahneme chcenú časovú zložitosť $\mathcal{O}(N \log N)$, lebo z rekurentného vzťahu dostávame $\mathcal{T}(N) = 2\mathcal{T}(N/2) + \mathcal{O}(N)$. Čiže najdôležitejšou časťou algoritmu je nájdenie rozdeľujúcej priamky a spojenie Vor(S_L) a Vor(S_R) do Voronoiovho diagramu množiny S. Priamku m vieme vytvoriť pomocou optimálneho triediaceho algoritmu v čase $\mathcal{O}(N \log N)$, ale postačí ak zoberieme ľubovoľný algoritmus, ktorý zvládne vytvorenie priamky m v čase $\mathcal{O}(N)$. Krok 3 rozoberieme podrobne v nasledujúcom algoritme MergeVoronoi.

Algoritmus MergeVoronoi:

(1) krok: Skonštruujeme konvexné obaly množín S_L a S_R, označme ich conv(S_L) a conv(S_R).
(2) krok: Pre conv(S_L) a conv(S_R) zostrojíme ich oporné úsečky P_aP_b a P_cP_d tak, aby tieto úsečky spájali conv(S_L) a conv(S_R) do konvexného obalu conv(S). Uvedomme si, že $P_a, P_c \in S_L$ a $P_b, P_d \in S_R$. Nech úsečka P_aP_b sa nachádza nad úsečkou P_cP_d. Na začiatku algoritmu nech je cesta σ prázdna množina.



OBR. 4.40. Ilustrácia krokov algoritmu VoronoiDivide-AndConquer na zadanej množine bodov. Voronoiov diagram $\mathrm{Vor}(S_L)$ (a). Voronoiov diagram $\mathrm{Vor}(S_R)$ (b). Cesta σ je znázornená červenou lomenou čiarou. Výsledok po odstránení nadbytočných strán diagramu (d).

(3) krok: Pridáme prvý vrhol W_0 cesty σ. Nech os úsečky P_aP_b je priamka k, potom bod $W_0 \in k$ je bod v nekonečne smerom nahor. Označme si pomocnú úsečku $P_xP_y = P_aP_b$.

(4) krok: Pokiaľ je $P_xP_y \neq P_aP_b$ opakujeme tieto kroky:

(a) Nájdeme prienik K_L priamky k s hranicou $\mathrm{Vor}(S_L)$ taký, ktorý nepatrí ceste σ (je rôzny od vrchol cesty), nachádza sa "pod" úsečkou P_xP_y a je k nej najbližšie.

(b) Nájdeme prienik K_R priamky k s hranicou $\mathrm{Vor}(S_R)$ taký, ktorý nepatrí ceste σ a je najbližšie k úsečke P_xP_y.

(c) Ak y-ová súradnica prieniku K_L je menšia,

- tak pridáme K_R do zoznamu vrcholov cesty σ a do P_y pridáme bod z hranice $\mathrm{conv}(S_R)$, ktorý nasleduje za bodom P_y v kladnom smere orientácie.
- inak pridáme K_L do zoznamu vrcholov cesty σ a do P_x pridáme bod z hranice $\mathrm{conv}(S_L)$, ktorý nasleduje za bodom P_x v zápornom smere orientácie.

(d) Nájdi os k úsečky P_xP_y.

(e) Pridaj posledný bod cesty σ. Je to bod v nekonečne smerom nadol na priamke k.

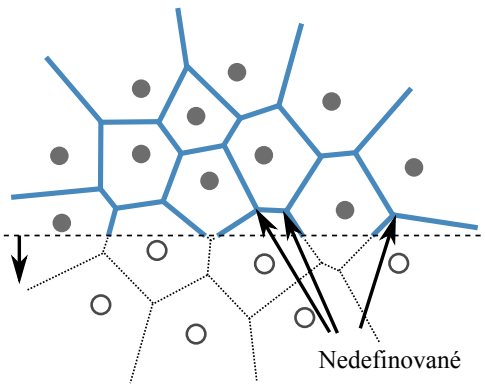
(5) Z $\mathrm{Vor}(S_L)$ odstráň hrany ležiace vpravo od σ a z $\mathrm{Vor}(S_R)$ odstráň hrany ležiace vľavo od σ.

Teraz uvedieme časovú zložitosť algoritmu MergeVoronoi. Na vytvorenie konvexných obalov $\mathrm{conv}(S_L)$ a $\mathrm{conv}(S_R)$ vieme zostrojiť algoritmus s lineárnou zložitosťou. Nájdenie oporných úsečiek týchto konvexných obalov trvá $\mathcal{O}(N)$. Pridanie prvého vrchola cesty vykonáme v konštantnom čase. Všetky podkroky kroku 4 algoritmu MergeVoronoi vykonáme v konštantnom čase a celý krok 4 zopakujeme najviac N-krát, teda jeho zložitosť je $\mathcal{O}(N)$. Odstránenie nadbytočných hrán z $\mathrm{Vor}(S_L)$ a $\mathrm{Vor}(S_R)$ vieme taktiež vykonať v lineárnom čase. Sčítaním zložitostí jednotlivých krokov dostávame lineárnu zložitosť $\mathcal{O}(N)$ algoritmu MergeVoronoi. Tým sme ukázali, že algoritmus VoronoiDivideAndConquer vieme vykonať v čase $\mathcal{O}(N \log N)$.

Pamäťová zložitosť algoritmu je $\mathcal{O}(N \log N)$, pretože rekurzívne prerozdeľovanie si predstavme ako binárny strom, ktorého pamäťová zložitosť je $\mathcal{O}(\log N)$, a navyše musíme v celej hierarchii pamätať Voironov diagram čo je $\mathcal{O}(N)$. □

Riešenie (Matej Hudák, 25.11.2010):

Nech je v rovine daná množina $P = \{p_1, \ldots, p_n\}$, $n \in N$. Problém použitia klasického zametacieho algoritmu v prípade nájdenia Voronoiovho diagramu je zobrazený na obrázku 1. Nato aby sme definovali určitý segment VD potrebujeme zametacou priamkou prejsť generátor tohto segmentu. Tu však vznikajú nepredpokladané segmenty VD, o ktorých v čase tesne pred prejdením generátora nového segmentu VD nevieme dostatok informácií.



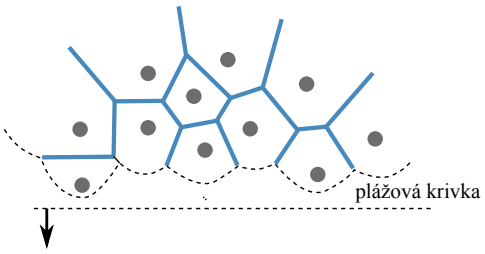
Obr. 4.41. Problém klasického zametacieho algoritmu

Na nájdenie Voronoiovho diagramu pre n bodov v rovine môžeme použiť zametací algoritmus, ktorý pôvodne popísal S. Fortune.

Fortune popísal hranu Voronoiovho diagramu pomocou dvoch kužeľov v priestore E^3 vztýčených v dvoch bodoch množiny generátorov $VD(P)$. Tretí rozmer v priestore reprezentuje čas. V určitom čase sa dva kužele pretnú, pričom ich prienikom bude krivka (hyperbola), ktorá po kolmom premietnutí do roviny XY vráti priamku, ktorá je osou úsečky bodov p_i a p_j. Nápad pána S. Fortuna spočíva v zametaní týchto kužeľov s naklonenou rovinou so sklonom 45 stupňov a prieniky s kužeľmi premietať do roviny XY. Výsledkom takého prechodu sú kúsky parabol spojené do „parabolického frontu“, pričom body, v ktorých sa jednotlivé kúsky parabol stretnú sú bodmi hrán $VD(P)$. Pomocou naklonenej roviny takto zamedzujeme nedefinovaným segmentom, až kým neprejdeme ďalším bodom z P.

Prispôsobenie klasického zametacieho algoritmu je teda vo vyriešení problému, že body, ktoré ležia pred zametacou priamkou môžu ovplyvniť VD, ktorý leží za zametacou priamkou. Nasledujúci algoritmus je založený na pôvodnom algoritme pána Fortuna, ale trochu pozmenený. Namiesto skreslenia diagramu a použitia naklonenej roviny, skreslíme zametaciu priamku. Skreslená zametacia priamka*tzv. plážová krivka* je monotónna krivka zložená z častí parabol. Samozrejme využijeme aj klasickú zametaciu priamku, no tesne za ňou bude nasledovať plážová krivka kopírujúc len už definované časti VD.

Plážová krivka je definovaná ako hraničná krivka spracovanej oblasti. V jednom sú body, ktoré sú bližšie k nejakému z bodov množiny P ako k zametacej priamke. V druhom sú body, ktoré sú bližšie k zametacej priamke ako k niektorému z bodov P. Hranicu týchto regiónov tvorí krivka zložená z parabol, pričom body, v ktorých sa krivka láme (spájajú sa tu jednotlivé paraboly) ležia na hranách VD.



Obr. 4.42. Príklad plážovej krivky

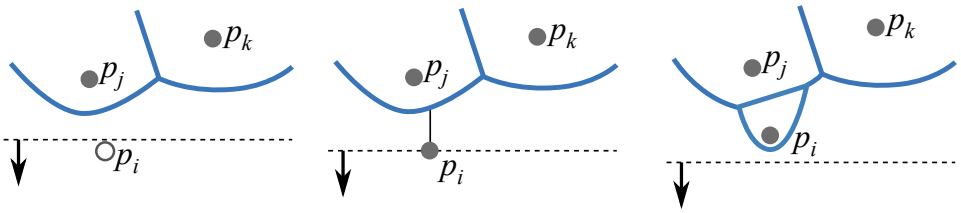
Za touto krivkou sa nachádza časťVD, ktorá je už dobre definovaná. Plážová krivka sa bude meniť v závislosti od pohybu zametacej priamky zhora nadol (pozri obrázok 2).

Priebeh algoritmu

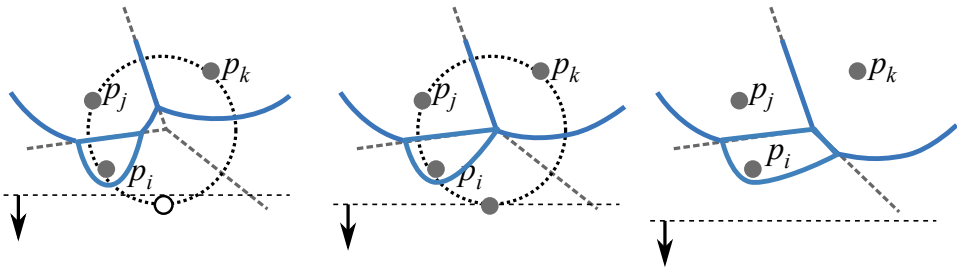
Algoritmus spočíva v zmenách plážovej krivky počas prechodu zametacou priamkou zhora nadol a určení spoločných bodov parabol, z

ktorých je zložená plážová krivka. Tieto body vytvárajú hrany VD. Počas prechodu budeme stále aktualizovať stav zametacej priamky (plážová krivka sa správa automaticky po každom aktualizovaní). Ďalej nás budú zaujímať len udalosti, ktoré môžu zmeniť topológiu VD.

- **zametacia priamka:**
  Uchovávame si y-ovú súradnicu zametacej priamky. Potrebujeme si pamätať aj tie a len tie body, ktorými je definovaná plážová krivka, utriedené podľa x-ových súradníc.
- **prejdenie bodu z P:**
  Keď zametacia priamka prejde cez nejaký z bodov p_i, pridáme do plážovej krivky novú parabolu. Pritom jeden z oblúkov v plážovej krivke rozdelíme. Body si udržiavame v rastúcom poradí, aby sme vedeli kam pridáme novú parabolu (vertikálny lúč z bodu p_i). Všimnime si, že zakrivenie jednotlivých parabol sa zmenšuje so vzdialenosťou od zametacej priamky (pozri obrázok 3).
- **prejdenie bodu z hrán VD:**
  Body hrán VD sa generujú dynamicky pomocou troch bodov, ktoré sú v plážovej krivke za sebou. Ak určitá časť kružnice opísanej týmito troma bodmi leží pred zametacou priamkou, bod VD ešte nie je definovaný. Táto vlastnosť je z základných vlastností VD. Potom zametacia priamka prejde cez celú kružnicu, definujeme bod VD (pozri obrázok 4).



Obr. 4.43. Spracovanie bodu množiny P



Obr. 4.44. Výpočet bodu hrany VD

Technické detaily

Voronoiov diagram:
Voronoiov diagram uložíme napríklad v $DCEL$. Keďže počas prechodu vznikajú neohraničené oblasti, ktoré sa dynamicky dotvárajú, využijeme ohraničujúci box, v ktorom bude uložený celý diagram.

Plážová krivka:
Plážová krivka pozostáva z utriedených bodov P, ktoré generujú jednotlivé paraboly. Na jej reprezentáciu môžeme využiť napríklad vyvážený binárny strom. Celú parabolu nepotrebujeme nikde ukladať alebo pamätať si. Stačí nám bod, ktorý vytára danú parabolu. Medzi každými dvoma bodmi p_i a p_j sa nachádza bod zlomu. Ako vidíme na obrázku 4, tento bod je stred kružnice, ktorá prechádza p_i a p_j. Tieto body zlomu stačí vypočítať práve v čase, keď sa kružnica dotýka zametacej priamky. Vtedy definujeme ďalší bod voronoiovho diagramu. Dôležité operácie, ktoré musíme implementovať pre prácu s plážovou krivkou sú *search*, *insert*, *split* a *delete*.

- operácia *search* má ako vstup y-ovú súradnicu zametacej priamky a nový bod p_i z P. Definujeme časť paraboly, ležiacej nad bodom p_i. Ak p_j (ako na obrázku 4 vľavo) je za bodom p_i a časti parabol sa pretínajú, pomocou tohto vyhľadania vrátime ako výstup aj bod p_j.
- operáciou *insert* vložíme nový bod p_i do plážovej krivky, spolu s jej parabolou. To znamená na miesto jednej časti paraboly $(\dots p_j \dots)$ tu teraz bude $(\dots p_j, p_i, p_j \dots)$. Na rozdelenie použijeme operáciu *split*. Vidíme to opäť na obrázku 4 vľavo, kde v x-ovom smere je najprv definovaná časť paraboly bodu p_j, potom je vložená časť paraboly bodu pi a potom opäť časť paraboly bodu p_j.
- keď sa zametacou priamkou dostaneme ďalej (na obrázku 4 vpravo), môžeme smerník na p_j vymazať (operácia *delete*) v plážovej krivke. Postupnosť $(\dots p_j, p_i, p_j, p_k \dots)$ sa nahradí postupnosťou $(\dots p_j, p_i, p_k \dots)$.

Všetky spomenuté operácie vieme spraviť v čase $\mathcal{O}(log\, n)$. Algoritmus sa začína ako klasicky zametací algoritmus. Do zametacej priamky vložíme body P s najmenšou y-ovou súradnicou. Spracujeme tieto body a posunieme sa ďalej.

Pri každom prejdení bodu x z P pomocou zametacej priamky, vyhľadáme, kde tento bod bude uložený v plážovej krivke. Vyrátame časť paraboly, definovanej týmto bodom, uložíme tento bod do plážovej krivky. Určíme dočasné hrany, generované až kým sa kružnica pretínajúca dva susedné body z množiny P nedotýka zametacej priamky a potom definujeme bod VD. Takýmto spôsobom prejdeme všetkými bodmi s výsledkom Voronoiovho diagramu n bodov v rovine.

Analýza zložitosti

Rovnako ako pre klasický zametací algoritmus. Spracovanie každej novej udalosti je v čase $\mathcal{O}(1)$. Veľkosť uložených dát je $\mathcal{O}(n)$, pričom každá operácia trvá $\mathcal{O}(log\, n)$. Celková časová zložitosť je preto $\mathcal{O}(n\, log\, n)$ a celková pamäťová zložitosť je $\mathcal{O}(n)$. □

Cvičenie 4.9 (35 bodov). *Opíšte transformáciu problému nájdenia Voronoiovho diagramu v rovine na problém vyhľadania konvexného obalu v 3D.*

Riešenie (Júlia Kučerová, 28.11.2010): Majme množinu $P = \{p_i = [x_i, y_i, 0]; i = 1, \ldots, n\}$ bodov v rovine xy. Premietneme P do priestoru na paraboloid $Q : z = x^2 + y^2$ s vrcholom v začiatku súradnicovej sústavy priestoru $\mathbb{E}^3$.

Získame novú množinu bodov

$$P^* = \{p_i^* = [x_i, y_i, x_i^2 + y_i^2]; i = 1, \ldots, n\},$$

ktorá je zdvihom množiny P.

Nech A_i je dotyková rovina k paraboloidu Q v bode p_i^* s rovnicou

$$z = 2x_i x + 2y_i y - (x_i^2 + y_i^2).$$

Potom $L_{ij}^* = A_i \bigcap A_j$ je hraničná priamka medzi A_i a A_j. Túto priamku kolmo premietneme do roviny xy. Týmto priemetom nám vznikne os bodov $p_i p_j$, označme ju L_{ij}. Rovnicu osi dostaneme elimináciou z-ovej súradnice vo vyjadrení priamky L_{ij}^*. Táto rovnica bude mať tvar:

$$L_{ij} : 2x(x_i - x_j) + 2y(y_i - y_j) - (x_i^2 + y_i^2 - x_j^2 - y_j^2) = 0.$$

Hľadajme bod priamky L_{ij}^*, ktorý patrí L_{ij}. Výpočtom zistíme, že je to bod $\frac{1}{2}p_i + \frac{1}{2}p_j$, čiže stred úsečky $p_i p_j$.

Zoberme bod $p_i \in P$. Nech $\overline{A_j^+}$ je polpriestor určený rovinou A_j obsahujúci paraboloid Q. Potom

$$A_i \cap \bigcap_{j=1, j\neq i}^{n} \overline{A_j^+}$$

je po premietnutí do xy práve $V(p_i)$.

$$\bigcap_{j=1, j\neq i}^{n} (A_i \cap \overline{A_j^+}) = A_i \cap \bigcap_{j=1, j\neq i}^{n} \overline{A_j^+}$$

Ak $\pi\colon \mathbb{E}^3 \to \mathbb{E}^2$ je projekcia v smere osi z do xy, potom

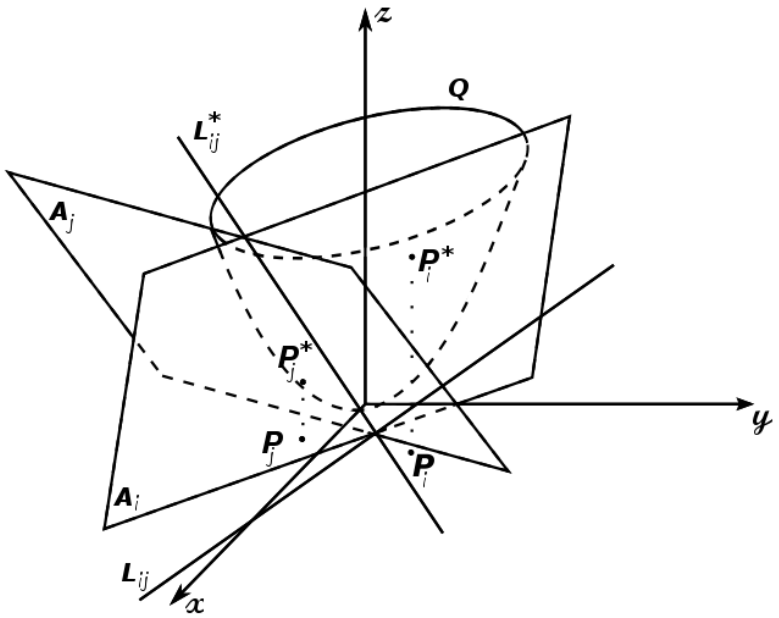
$$\pi(\bigcap_{j=1}^{n} (A_i \cap \overline{A_j^+})) = V(p_i)$$

Z toho vyplýva, že časť prienikov dotykových rovín v priestore dáva po kolmom premietnutí do roviny xy priamo Voronoiov diagram. □

Cvičenie 4.10 (10 bodov). *Ukážte, že triangulácia n bodov v rovine vyžaduje asymptoticky čas najmenej $\mathcal{O}(n \log n)$.*

Riešenie (Ján Karahuta, 21.11.2008):

Existuje súvis medzi triedením a trianguláciou, a preto sa dá jej zložitosť odvodiť zo zložitosti triedenia. Pre n daných reálnych čísel v

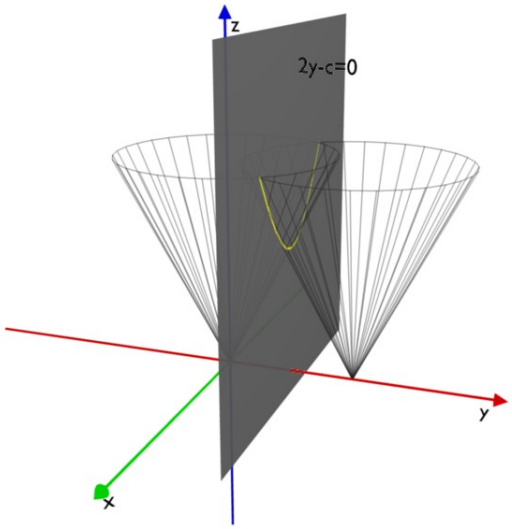


Obr. 4.45. Výpočet osi bodov pomocou dotykových rovín Q

rovine ich y-ovú súradnicu položíme rovnú nule. Získame tak n bodov ležiacich na jednej priamke. Ak k týmto bodom pridáme ľubovoľný bod s nenulovou y-ovou súradnicou získame tak množinu $n+1$ bodov, ktorá ma jedinú trianguláciu. Triangulácia nám v čase $T(n)$ vytvorila zoznam hrán a ten môžme použiť na utriedenie pôvodných prvkov v čase $\mathcal{O}(n)$. Triedime teda v čase $\mathcal{O}(T(n)+n)$. Zo zložitosti triedenia $T : \Omega(n \log n)$ nám teda aj pre trianguláciu vyplýva zložitosť asymptoticky najmenej $T : \mathcal{O}(n \log n)$. □

Cvičenie 4.11 (10 bodov). *Ukážte, že prienik dvoch rôznych rotačných kužeľov s rovnobežnými osami, vrcholmi ležiacimi v jednej rovine kolmej na osi rotácie a rovnakými vrcholovými uhlami je hyperbola.*

Riešenie (Miroslava Fekiačová, 1.11.2008): Majme dve rovnice rotačných kužeľov: $\frac{x^2}{a^2}+\frac{y^2}{a^2}-\frac{z^2}{b^2}=0$ a $\frac{x^2}{a^2}+\frac{(y-c)^2}{a^2}-\frac{z^2}{b^2}=0$. Z druhej rovnice si vyjadríme z^2: $z^2 = b^2(\frac{x^2+(y-c)^2}{a^2})$ Po dosadení do prvej rovnice dostaneme výraz: $\frac{x^2}{a^2}+\frac{y^2}{a^2}-\frac{x^2+(y-c)^2}{a^2}=0$. Upravením tejto rovnice dostaneme $2y-c=0$, čo je rovnica roviny (pozri obr. 4.46). To znamená, že priesečník dvoch rotačných kužeľov sa nachádza v rovine $2y-c=0$. Stačí teda brať do úvahy priesečník jedného z rotačných kužeľov s rovinou



Obr. 4.46. Prienik dvoch rotačných kužeľov s rovnakým vrcholovým uhlom.

danou rovnicou $2y - c = 0$. Vieme, že priesečník rotačného kužeľa a roviny je kužeľosečka. Stačí určiť, o ktorý typ ide. Rovina daná rovnicou $2y - c = 0$ nepretína os kužeľa, teda dostávame hyperbolu. □

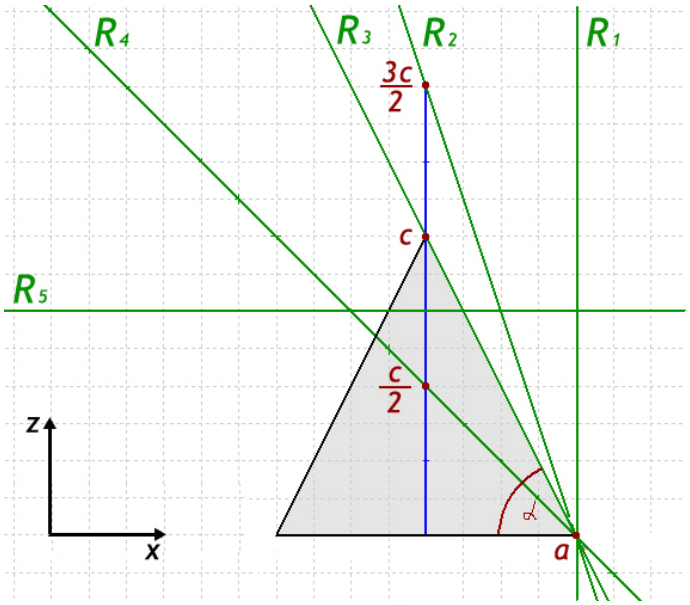
Riešenie (Kristína Lidayová, 19.11.2011): Majme dva rôzne rotačné kužele. Jeden má stred v $S_1[0,0,0]$ a teda má rovnicu $\frac{x^2}{a^2} + \frac{y^2}{a^2} - \frac{z^2}{c^2} = 0$ a druhý rotačný kužeľ má stred v $S_2[i,j,k]$. Jeho rovnica je $\frac{(x-i)^2}{a^2} + \frac{(y-j)^2}{a^2} - \frac{(z-k)^2}{c^2} = 0$. Premenná a vyjadruje polomer podstavy kužeľa, c je výška kužeľa. Postavme tieto dve rovnice do rovnosti.

$$\frac{x^2}{a^2} + \frac{y^2}{a^2} - \frac{z^2}{c^2} = \frac{(x-i)^2}{a^2} + \frac{(y-j)^2}{a^2} - \frac{(z-k)^2}{c^2}$$

$$\frac{x^2}{a^2} + \frac{y^2}{a^2} - \frac{z^2}{c^2} = \frac{x^2-2ix+i^2}{a^2} + \frac{y^2-2jy+j^2}{a^2} - \frac{z^2-2kz+k^2}{c^2}$$

$$-\frac{2i}{a^2}x - \frac{2j}{a^2}y + \frac{2k}{c^2}z + \frac{i^2}{a^2} + \frac{j^2}{a^2} - \frac{k^2}{c^2} = 0$$

Úpravami sme dostali rovnicu roviny. To znamená, že priesečník dvoch rotačných kužeľov sa nachádza v rovine. Na získanie riešenia teda stačí brať do úvahy priesečník jedného rotačného kužeľa a tejto roviny. Priesečník rotačného kužeľa a roviny je kužeľosečka. Typ kužeľosečky bude závisieť od vzájomnej polohy rotačných kužeľov, a teda aj od roviny, v ktorej sa ich priesečník bude nachádzať.



Príklad rôznych polôch rovín pretínajúcich kužeľ.

Pokiaľ vrcholy oboch rotačných kužeľov budú ležať v jednej rovine kolmej na os rotácie, ich spoločný prienik bude ležať v rovine kolmej na podstavu kužeľov a bude to **časť hyperboly**. Rovnica roviny je $x = d$. Na obrázku je označená ako R_1. Prienik vyrátame nasledujúcim výpočtom, keď si do rovnice kužeľa za x dosadíme d.

$$\frac{d^2}{a^2} + \frac{y^2}{a^2} - \frac{z^2}{c^2} = 0$$
$$\frac{y^2}{a^2} - \frac{z^2}{c^2} = -\frac{d^2}{a^2}$$

Dostávame rovnicu hyperboly.
Rovnicu hyperboly by sme dostali, aj v prípade, že by rovina zvierala s podstavou kužeľa uhol väčší ako α (znázornený na obrázku), ale menší ako 90°. Príkladom takejto polohy je rovina R_2. Táto rovina má rovnicu $z = \frac{3c}{2a}x + q$. Výpočet prieniku s takouto rovinou by vyzeral takto:

$$\frac{x^2}{a^2} + \frac{y^2}{a^2} - \frac{(\frac{3c}{2a}x+q)^2}{c^2} = 0$$
$$\frac{x^2}{a^2} + \frac{y^2}{a^2} - \frac{(\frac{9c^2}{4a^2}x^2+2\frac{3c}{2a}qx+q^2)}{c^2} = 0$$
$$\frac{x^2}{a^2} + \frac{y^2}{a^2} - \frac{9c^2x^2}{4a^2c^2} - \frac{6cqx}{2ac^2} - \frac{q^2}{c^2} = 0$$
$$c^2x^2 + c^2y^2 - \frac{9c^2x^2}{4} - acxq - a^2q^2 = 0$$
$$-\frac{5c^2x^2}{4} + c^2y^2 = acqx + a^2q^2$$

V prípade, keď by rovina zvierala s podstavou kužeľa uhol α, kde $tan\ \alpha = \frac{c}{a}$, bola by prienikom dvoch rotačných kužeľov **časť paraboly**. Na obrázku je znázornený príklad takéhoto prípadu rovinou R_3. Táto rovina má rovnicu $z = \frac{c}{a}x + q$. Výpočet prieniku s rovinou R_3 by vyzeral takto:

$$\frac{x^2}{a^2} + \frac{y^2}{a^2} - \frac{(\frac{c}{a}x+q)^2}{c^2} = 0$$

$$\frac{x^2}{a^2} + \frac{y^2}{a^2} - \frac{(\frac{c^2}{a^2}x^2 + 2\frac{c}{a}qx + q^2)}{c^2} = 0$$
$$\frac{x^2}{a^2} + \frac{y^2}{a^2} - \frac{c^2x^2}{a^2c^2} - \frac{2cqx}{ac^2} - \frac{q^2}{c^2} = 0$$
$$c^2x^2 + c^2y^2 - c^2x^2 - 2acxq - a^2q^2 = 0$$
$$c^2y^2 = 2acqx + a^2q^2$$

Vidíme, že riešením by bola parabola. Tento prípad však pri prieniku dvoch rôznych rotačných kuželov s rovnobežnými osami rotácie a rovnakými vrcholovými uhlami nenastáva.

Ďalším možným prípadom je rovina R_4, ktorej všeobecná rovnica je $z = \frac{c}{2a}x + q$. Do tejto skupiny patria prípady, keď rovina zviera s podstavou kužeľa uhol menší ako α, kde $tan\ \alpha = \frac{c}{a}$. Výsledkom v týchto prípadoch je **časť elipsy**. Výpočet prieniku je nasledujúci:

$$\frac{x^2}{a^2} + \frac{y^2}{a^2} - \frac{(\frac{c}{2a}x + q)^2}{c^2} = 0$$
$$\frac{x^2}{a^2} + \frac{y^2}{a^2} - \frac{(\frac{c^2}{4a^2}x^2 + 2\frac{c}{2a}qx + q^2)}{c^2} = 0$$
$$\frac{x^2}{a^2} + \frac{y^2}{a^2} - \frac{c^2x^2}{4a^2c^2} - \frac{2cqx}{2ac^2} - \frac{q^2}{c^2} = 0$$
$$c^2x^2 + c^2y^2 - \frac{c^2x^2}{4} - acxq - a^2q^2 = 0$$
$$\frac{3c^2x^2}{4} + c^2y^2 = acqx + a^2q^2$$

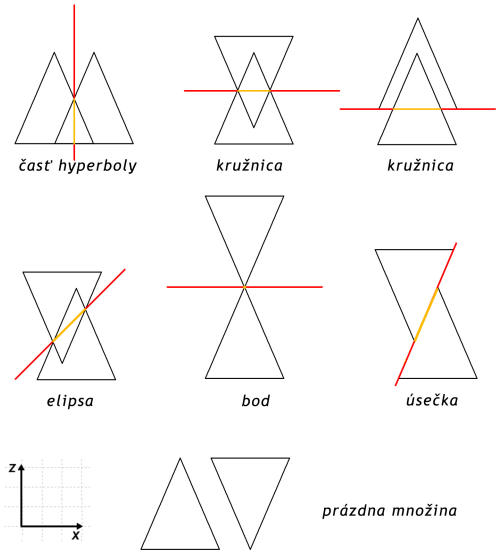
Overili sme, že riešením je elipsa.

Špeciálny prípad je pokiaľ oba rotačné kužele majú rovnakú x-ovú a y-ovú súradnicu a líšia sa len v súradnici z. Vtedy bude ich spoločný prienik ležať v rovine rovnobežnej s podstavou kužeľov. Prienikom bude špeciálny prípad elipsy, čiže **kružnica**. Rovina v tomto prípade bude mať rovnicu $z = d$. Na obrázku je označená ako R_5. Výpočet prieniku spočíva v dosadení do rovnice kužeľa za z premennú d.

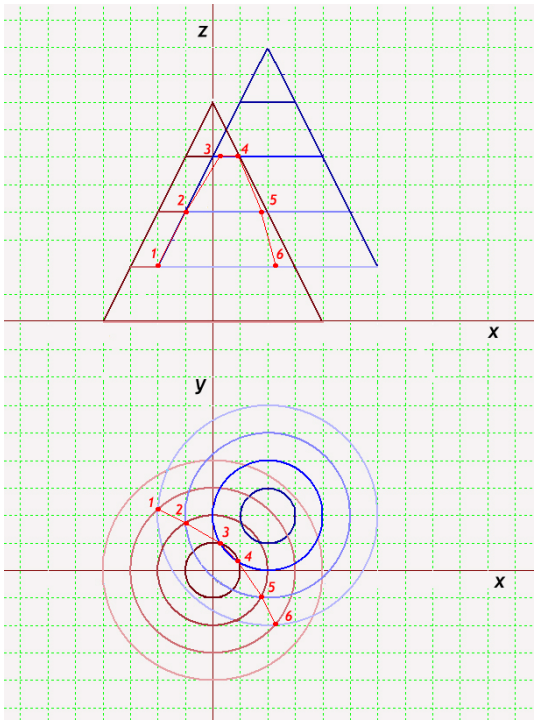
$$\frac{x^2}{a^2} + \frac{y^2}{a^2} - \frac{d^2}{c^2} = 0$$
$$\frac{x^2}{a^2} + \frac{y^2}{a^2} = -\frac{d^2}{c^2}$$

Dostali sme rovnicu kružnice.

Pokiaľ hľadáme prieniky 2 rôznych rotačných kužeľov s rovnobežnými osami rotácie a rovnakými vrcholovými uhlami môžu nastať nasledujúce prípady: **časť hyperboly, časť elipsy, kružnica** a špeciálne prípady: **úsečka, bod a prázdna množina**. Parabola za týchto podmienok nenastáva.



Príklad rôznych typov prienikov 2 rôznych rotačných kuželov s rovnobežnými osami rotácie a rovnakými vrcholovými uhlami. y-ová súradnica kužeľov je v týchto príkladoch rovnaká.



Príklad prieniku dvoch rôznych rotačných kužeľov, keď y-ová súradnica kužeľov je rôzna -> časť hyperboly. Premietnutie do rovín xy a xz.

□

CVIČENIE 4.12 (25 bodov). *Nech* $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \subseteq \mathbb{E}^2$ *má* n *prvkov. Dokážte, že pri transformácii roviny transformáciou danou predpisom* $\mathbf{x} = (x_1 + d(\mathbf{x}, \mathbf{p}_i), x_2)$*, kde* $\mathbf{x} \in V(\mathbf{p}_i)$*,*

(1) *je bod s minimálnou súradnicou* x_1 *v transformovanom* $V(\mathbf{p}_i)$ *práve generátor* $\mathbf{p}_i$*,*
(2) *je obrazom osi dvoch susedných bodov z* $\mathcal{V}(P)$ *buď priamka, polpriamka alebo časť hyperboly.*

CVIČENIE 4.13 (25 bodov). *V rovine je daných n bodov a ich Voronoiov diagram v dátovej štruktúre DCEL. Popíšte algoritmus na vyhľadanie minimálnej euklidovskej kostry a odhadnite jeho zložitosť*

Riešenie (Martin Havala, 18.11.2009):

Predpokladajme, že steny sú označené podľa generátorov, ktoré obsahujú. Počet generátorov je n. Z vlastností Voronoiovho diagramu vieme, že rozdeľuje priestor podľa príslušnosti ku generátorom v závislosti od ich vzdialenosti. Teda množina najkratších hrán, ktoré spájajú nejakú množinu bodov, bude podmnožinou Voronoiovho diagramu ktorý tieto body generujú.

(1) `pre všetky hrany` e `{`
(2) `    zisti susedné steny (generátory) ku` e // toto vieme podľa predpokladu, keďže DCEL uchováva informáciu o susedných stenách
(3) `    vlož trojicu (vzdialenosť,` e`) do zoznamu` α
(4) `    }`

(5) `utrieď zoznam` α `podľa vzdialenosti`
(6) `vytvor zoznam` β
(7) `vytvor množinu` E
(8) `i = 0`

(9) `pokiaľ (dĺžka zoznamu` Γ `< n ){`
(10) `    ak` α_i`.stena1` $\notin E$ `||` α_i`.stena2` $\notin E${
(11) `        vlož {` α_i`.stena1,` α_i`.stena2 } do` E
(12) `        vlož hranu` α_i `do` β
(13) `        }`
(14) `    i++`
(15) `}`
(16) `vráť` β

Zoznam β obsahuje hrany, ktoré spájajú všetky vrcholy grafu a určite neobsahuje žiaden cyklus, keďže algoritmus ošetruje pridávanie takejto hrany (oba jej krajné vrcholy by už boli v množine E a teda túto hranu preskočíme). Tieto hrany taktiež tvoria minimálnu možnú euklidovskú kostru, pretože pridávame vždy najkratšiu možnú hranu,

až kým nespojíme všetky vrcholy. Taktiež je algoritmus konečný, pretože po prejdení celého zoznamu α určite prejdeme všetky vrcholy.

Zložitosť vytvorenia zoznamu α je $\mathcal{O}(n)$, pretože ide o jednoduché prejdenie pôvodného zoznamu hrán a zistenie vzdialenosti dvoch bodov. Potom zoznam α vytriedime, čo dokážeme vyriešiť v čase $\mathcal{O}(n \log n)$ a napokon ním opäť jedenkrát prejdeme. Teda časová zložitosť tohoto algoritmu je $\mathcal{O}(n \log n)$. □

Riešenie (Matej Hudák, 25.11.2010):

Nech je daných n bodov v rovine a Voronoiov diagram. Tieto body si môžeme definovať pomocou euklidovského grafu, kde hrana definovaná jej koncovými bodmi p_i a p_j má určitú váhu rovnú euklidovskej vzdialenosti z p_i do p_j. Minimálna euklidovská kostra je potom množina $n-1$ hrán, ktoré spájajú body tak, aby váha týchto hrán bola čo najmenšia. Nato môžeme použiť Kruskalov algoritmus. Ten funguje tak, že najprv zoradí hrany podľa ich váh a potom ich pridáva po jednej. Najprv však potrebujeme euklidovský graf. Zložitosť takého algoritmu je však $\mathcal{O}(n^2 \, log \, n)$.

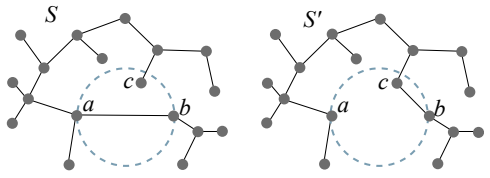
Práve preto použijeme Voronoiov diagram, presnejšie Delaunayovu trianguláciu a to nasledovne.

(1) výpočet Delaunayovej triangulácie z VD - $\mathcal{O}(n \, log \, n)$
(2) použiť Kruskalov algoritmus - $\mathcal{O}(n \, log \, n)$

Celkový čas $\mathcal{O}(n \, log \, n)$. Prečo to funguje? Predpoklad je taký, že množina hrán minimálnej euklidovskej kostry je podmnožinou Delaunayovej triangulácie. Výpočtom Delaunayovej triangulácie teda vytvoríme výrazne jednoduchší graf, ktorý potom použijeme na výpočet MEK.

Môžeme to dokázať nasledovne. Nech S je minimálna euklidovská kostra. Nech $w(S)$ je celková váha S. Vezmime si dva ľubovoľné body a a b, ktoré tvoria hranu v MEK. Zároveň predpokladajme, že hrana $\overline{ab}$ nepatrí do Delaunayovej triangulácie. Ukážeme, že ak je to tak, potom táto hrana nepatrí ani MEK.

Z predchádzajúceho ($\overline{ab}$ nepatrí Delaunayovej triangulacií) vyplýva, že neexistuje prázdna kružnica prechádzajúca a, b. Takáto kružnica(s priemerom $\overline{ab}$) teda obsahuje ďalší bod c (pozri obrázok 1).



Obr. 4.47. Minimálna euklidovská kostra

Povedzme, že teraz vymažeme hranu $\overline{ab}$ z S a štruktúra sa rozdelí na dva subštruktúry. Bez ujmy na všeobecnosti, predpokladajme, že

bod c leží v rovnakej subštruktúre ako bod a. Tak vymažme hranu $\overline{ab}$ a pridajme hranu $\overline{bc}$. Keďže priemer kružnice bol $\overline{ab}$ a c patril tejto kružnici tak platí $||\overline{bc}|| < ||\overline{ab}||$. Potom ale platí

$$w(S') = w(S) + ||bc|| - ||ab|| < w(S),$$

z čoho dostávame spor pre pôvodné tvrdenie, lebo S nie je MEK. Teda ak hrana $\overline{ab}$ nepatrí Delaunayovej triangulácií, tak nepatrí ani MEK. Ak sa v kružnici nachádza aj iný bod x, tento dôkaz stále platí, keďže vzdialenosť akéhokoľvek bodu (v kružnici) k bodom a a b bude vždy menšia ako vzdialenosť $\overline{ab}$, teda priemer kružnice.

Otázkou je, či vytvorená hrana $\overline{bc}$ nepretne inú hranu $\overline{de}$, ktorá prechádza kružnicou a zároveň sa jej krajné body v nej nenachádzajú. V tomto prípade sa treba zamyslieť či tu takáto hrana existuje. Táto hrana by musela prechádzať priestorom medzi bodmi b a c. Potom vzdialenosť $\overline{cd}$ alebo $\overline{ce}$ je menšia ako vzdialenosť $\overline{de}$. Z vlastností Delaunayovej triangulácie potom platí, že hrana $\overline{de}$ do nej nepatrí a teda nepatrí ani MEK. □

Cvičenie 4.14 (10 bodov). *Dokážte, že pre ľubovoľné $n > 3$ existuje množina n bodov v rovine taká, že aspoň jedna oblasť Voronoiovho diagramu má $n-1$ vrcholov.*

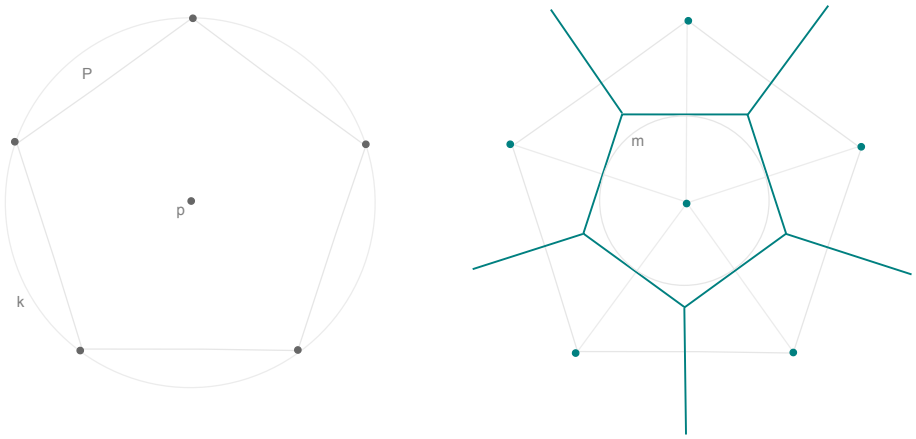
Riešenie (Martina Bátorová, 1.11.2008): Požiadavka existencie nám umožňuje vybrať si ľubovoľnú množinu, ktorá vyhovuje zadaným kritériám. Skonštruujme teda n–bodovú množinu, ktorej práve jeden vrchol bude mať Voronoiovu oblasť s $(n-1)$ vrcholmi.

Takáto množina vyzerá tak, že $(n-1)$ jej vrcholov tvorí pravidelný $(n-1)$–uholník a jeden vrchol je stred opísanej kružnice tohto $(n-1)$-uholníka (pozri obr. 4.48). Mnohouholník označme P, opísanú kružnicu k a jej stred p. Potom Voroniova oblasť p má práve $(n-1)$ vrcholov.

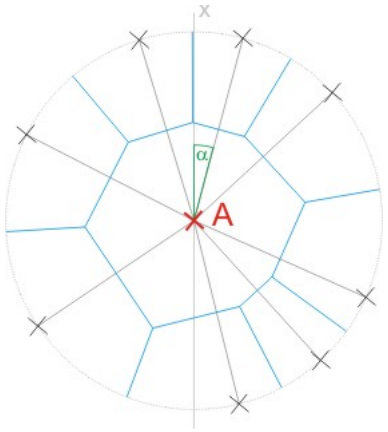
Dôvod, prečo je to tak, vyplýva priamo z konštrukcie Voronoiovho diagramu takejto množiny: keďže p je stredom k, je od každého iného generátora vzdialený rovnako. Kružnica m s polomerom rovným $\frac{1}{2}$ polomeru k obsahuje $(n-1)$ stredov úsečiek pv, $v \in P\backslash\{v\}$. Osi týchto úsečiek tvoria dotyčnice tejto kružnice. S osami úsečiek bodov, ktoré ležia na k, sa pretnú v $(n-1)$ bodoch (všetky tieto priesečníky sú od p vzdialené viac ako polomer m). Takto nám vzniknú vrcholy celého diagramu.

Bod p je stred m, preto leží v kruhu určenom m, a teda oblasť p má $(n-1)$ vrcholov. Je zjavné, že m je kružnica vpísaná Voronoiovej oblasti vrcholu p. □

Riešenie (Marta Režnáková, 8.11.2008): Dané body v rovine nám reprezentujú generátory Voronoiovho diagramu. Vezmime si teda bod A taký, že práve oblasť ním generovaná bude mať $n-1$ vrcholov (resp. stien). Znamená to, že ak by sme priradili ostatným generátorom



Obr. 4.48. Vľavo je množina vrcholov pre $n = 6$, vpravo výsledný Voronoiov diagram. Zobrazené sú i pomocné objekty: 5–uholník, kružnice, čiary použité pri konštrukcii diagramu.



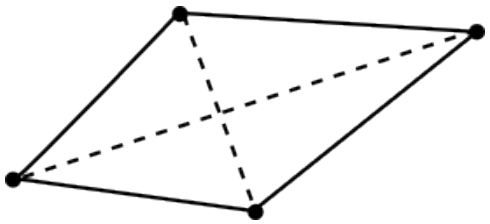
Obr. 4.49. Oblasť Voronoiovho diagramu s $n - 1$ hranami.

polárne súradnice, kde bod A by bol stredom danej kružnice, žiadne dva z generátorov by nemali priradený rovnaký uhol α. Inými slovami, ak chceme nájsť ľubovoľnú množinu n bodov takú, že aspoň jedna oblasť bude mať $n - 1$ vrcholov, môžeme si vziať napríklad body na kružnici a jej stred. Takáto množina môže mať nekonečne veľa bodov, čo inými slovami znamená, že sme schopní nájsť množinu vyhovujúcu podmienke a s $n > 3$ bodmi (pozri obr. 4.49). □

Cvičenie 4.15 (30 bodov). *Dokážte, že n bodov v rovine sa dá triangulovať exponenciálne veľa spôsobmi.*

Riešenie (Jakub Mišún, 30.11.2011):

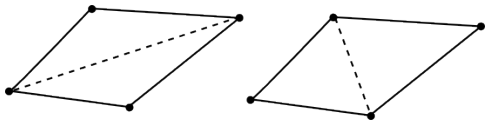
V učebnici sa uvádza, že každá n-tica bodov v rovina sa dá triangulovať $2^{8n+\theta(nlogn)}$ spôsobmi. Nám, netreba ukazovať presný počet triangulácií, stačí ukázať, že ich je exponenciálne veľa. Znalosť presného počtu nám pomôže len ako vodítko. To nám hovorí, že základ tejto exponencialnej funkcie je dva. Najprv sa pozrime na tie najjednoduchšie príklady. Zjavne, pre 3 body vieme nájsť len jediný trojuholník a teda triangulácia je jediná $1 = 2^0$. Pre 4 body už nadobúdame triangulácie $2 = 2^1$ a pre 5 bodov sú $4 = 2^2$.



Obr. 4.50. Dve triangulácie štvoruholníka

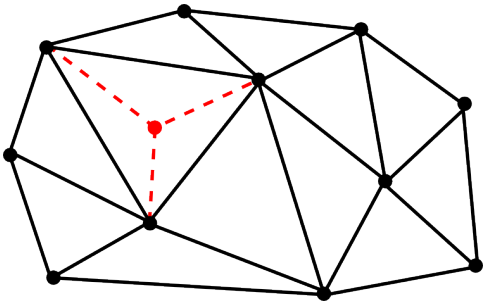
Exponenciálnosť pri základe 2 teda vyzerá sľubne. Problém nastáva pri vyšších počtoch bodov. Tu použijeme matematickú indukciu. Predpokladajme že pre n bodov máme najviac 2^{an+b} triangulácií. Pozrime sa čo sa stane ak pridáme ďalší bod P. Tu môžu nastať 2 prípady.

Bod sa môže nachádzať v konvexnom obale. V tomto prípade pre každú triangluláciu platí, že leží v jednom z trojuholníkov (počet trojuholníkov sa zvýši o dva) alebo na hrane dvoch trojuholníkov (počet trojuholníkov sa taktiež zvýši o 2). Nech je tak alebo onak, pridá nám do triangulácie vždy dva trojuholníky. Každá triangulácia takto nadobudne ďalšie dva trojuholníky, čo ale počet triangulácií neovplyvňuje. Dostávame rovnaký počet triangulácií ako pre n bodov.

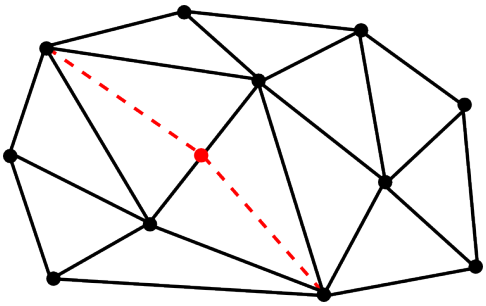


Obr. 4.51. Flipovanie trojuholníkov

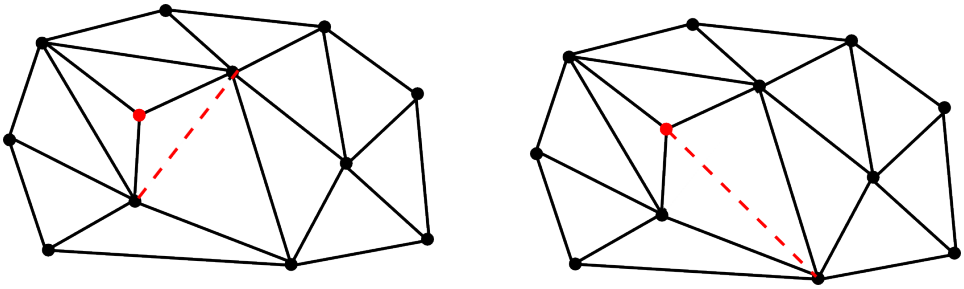
Teraz v triangulácií môžeme flipnúť dva trojuholníky, a to jeden nový a jeden starý, ktorý s ním má spoločnú hranu. Flipnutím nadobudneme novú triangluláciu. Potenciálne teda môžeme dostať 3 nové triangulácie ak je bod v trojuholníku a 4 nové triangulácie ak je na hrane susedných trojuholníkov. Keďže týmto spôsobom môžeme pre každú triangluláciu nadobudnúť minimálne dve nové triangulácie, tak minimálne zdvojnásobíme počet triangulácií. Špeciálnym prípadom je situácia, keď sa nový bod nachádza na hrane konvexného obalu. V tomto prípade sa nám trojuholník, ktorému patrí nový bod predelí na dva nové.



Obr. 4.52. Bod v trojuholníku pridá ďalšie dva



Obr. 4.53. Podobne aj keď sa nachádza na hrane



Obr. 4.54. Flipovanie nám zdvojnásobí počet triangulácií

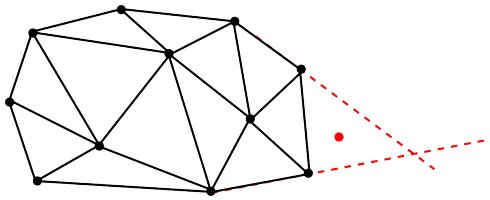
Bod sa môže nachádzať mimo konvexný obal a teda ho mení. Tento môže ležať v trojuholníku tvorenom predĺžením hrán prechádzajúcim jeho dvoma najbližšími susedmi P_i, P_j.

V tomto prípade pridá každej triangulácií jeden trojuholník P_i, P_j, P. Opäť vďaka flipovaniu zdvojnásobíme počet triangulácií. Tentoraz nemôže nastať prípad, keď nieje flipovanie možné.

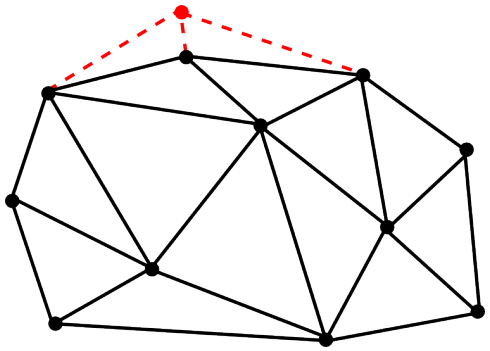
Alebo môže ležať mimo trojuholníku tvorenom predĺžením hrán prechádzajúcim jeho dvoma najbližšími susedmi P_i, P_j.

V tomto prípade pridáme dva trojuholníky každej triangulácií. Opäť flipovaním minimálne zdvojnásobíme počet triangulácií.

Samozrejme, môže nastať aj prípad, keď susedné trojuholníky nieje možné flipnúť pretože netvoria konvexný štvoruholník. Toto však neovplyvňuje našu možnosť zdvojnásobovať počet triangulácií, pretože



Obr. 4.55. Bod pridáva jeden trojuholník



Obr. 4.56. Bod pridá dvatrojuholníky

ak nový bod umiestnime tak, že dostávame dvojicu, ktorú nieje možné flipnúť, minimálne jedna dvojica zo zvyšných dvojíc flipovatelná bude.

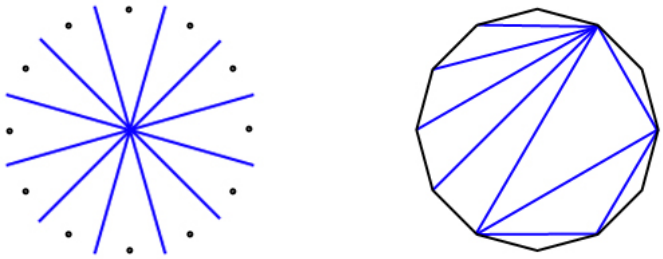
Vidíme teda, že nech bod P pridáme na ľubovolnú pozíciu počet triangulácií sa minimálne zdvojnásobí. Počet triangulácií teda narastie minimálne o základ predpokladanej exponenciálnej funkcie. Počet triangulácií narastá exponenciálne. □

Cvičenie 4.16. *Nech* $P = \{p_1, \ldots, p_n\} \subset \mathbb{E}^2$ *má* n *prvkov. Dokážte, že pri transformácii roviny transformáciou danou predpisom* $x = (x_1 + d(x, p_i), x_2)$, *kde* $x \in V(p_i)$,

- *(1) je bod s minimálnou súradnicou* x_1 *v transformovanom* $V(p_i)$ *práve generátor* p_i,
- *(2) je obrazom osi dvoch susedných bodov z* $V(P)$ *bud priamka, polpriamka alebo časť hyperboly.*

Cvičenie 4.17 (10 bodov). *Opíšte Voronoiov diagram a Delaunayovu trianguláciu pre pravidelný* n*-uholník.*

Riešenie (Lukáš Tencer, 21.10.2008): Voronoiov diagram pre pravidelný n-uholník bude vyzerať tak, že bude mať n otvorených stien, ktoré budú mať 1 spoločný bod T v ťažisku n-uholníka a hranami s_i steny diagramu budú polpriamky začínajúce v bode T zo smerom $\overrightarrow{TL}$ $L = v_i + \frac{1}{2}(v_i - v_{i+1})$ alebo $L = v_i + \frac{1}{2}(v_i - v_{i-1})$. Diagram bude mať taktiež n hrán. Pozri obr. 4.57.



(a) (b)

OBR. 4.57. (a) Voronoiov diagram pre vrcholy pravidelného 12-uholníka. (b) Delaunayova triangulácia pre pravidelný 12-uholník.

Na Delaunayovu trianguláciu pravidelného n-uholníka potrebujeme vložiť diagonály, vzájomne sa nepretínajúce, pokiaľ všetky regióny nebudú triangulované. Triangulácia nie je jednoznačná a pre pravidelný n-uholník ľubovoľná triangulácia bude mať $(n-3)$ diagonál a $(n-2)$ trojuholníkov. Počet všetkých možných triangulácií pravidelného n-uholníka je $C_{n-2} = \frac{1}{n-1}\binom{2n-4}{n-2}$. □

CVIČENIE 4.18 (30 bodov). *Opíšte Voronoiov diagram a Delaunayovu trianguláciu pre konvexný n-uholník.*

Riešenie (Attila Csidey, 21.11.2008):

Voronoiov diagram, ako aj Delaunayova triangulácia konvexného n-uholníka vyzerá podobne, ako pre pravidelný n-uholník (Cvičenie 4.16.). Pravidelný n-uholník je najšpeciálnejším prípadom konvexného n-uholníka.

Voronoiov diagram: Má n otvorených stien, a nemá žiadny ohraničený Voronoiov mnohouholník. Počet hrán v diagrame

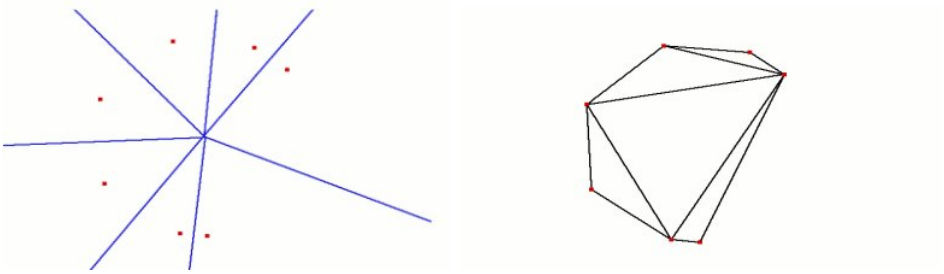
- minimálny počet hrán je n – špeciálny prípad obr. 4.58.
- maximálny počet hrán je $n + (n-3)$ - z toho n priamok a maximálne $(n-3)$ úsečiek, obr. 4.59.

Delaunayova triangulácia: Má $(n-3)$ diagonál a $(n-2)$ trojuholníkov. Počet rôznych triangulácií závisí od počtu hrán vo Voronoiovom diagrame. Pohybuje sa to medzi 1 a $(\frac{1}{n-1}\binom{2n-4}{n-2})$.

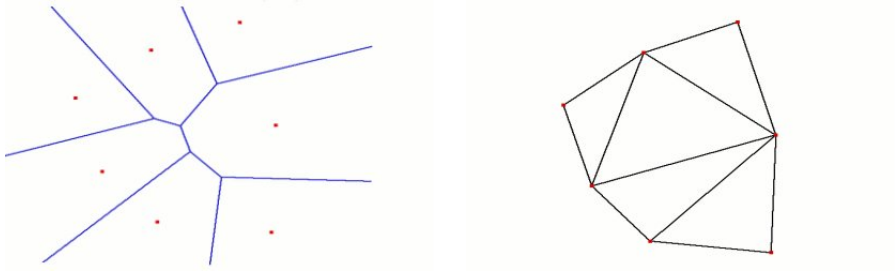
V najšpeciálnejšom prípade je počet hrán minimálny a počet triangulácií $(\frac{1}{n-1}\binom{2n-4}{n-2})$. (viď. Cvičenie 4.16.)

Pre maximálny počet hrán má jedinú Delaunayovu trianguláciu. □

CVIČENIE 4.19 (50 bodov). *Opíšte Voronoiov diagram a Delaunayovu trianguláciu vnútri n-uholníka, ktorý má len zvislé a vodorovné hrany, pričom prienik ľubovolnej zvislej resp. vodorovnej priamky s týmto mnohouholníkom má najviac jeden komponent súvislosti (tzv.* ortogonálne konvexný mnohouholník*).*



Obr. 4.58. (a) Voronoiov diagram vrcholov konvexného sedemuholníka. Počete hrán je minimálny n = 7, (b) Delaunayova triangulácia konvexného sedemuholníka, počet možných triangulácií je minimalny, len jedna.



Obr. 4.59. (a) Voronoiov diagram vrcholov konvexného sedemuholníka. Počete hrán je maximálny n+(n-3) = 11, (b) Delaunayova triangulácia konvexného sedemuholníka, počet možných triangulácií je minimalny, len jedna.

Riešenie (Júlia Kučerová, 28.11.2010): Majme ortogonálne konvexný mnohouholník v ktorom na každej hrane ležia iba 2 body, ktoré sú jej koncové. Voronoiov diagram sa skladá z ohraničených Voronoiových mnohouholníkov a neohraničených stien.

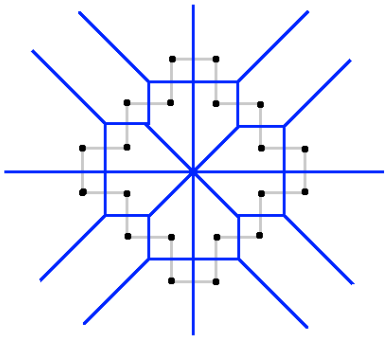
V ortogonálne konvexnom mnohouholníku zvierajú hrany pri každom vrchole tohto mnohouholníka uhol s veľkosťou 90°, alebo 270°. Zadefinujme si 2 typy uhlov:

$\alpha = 90°$

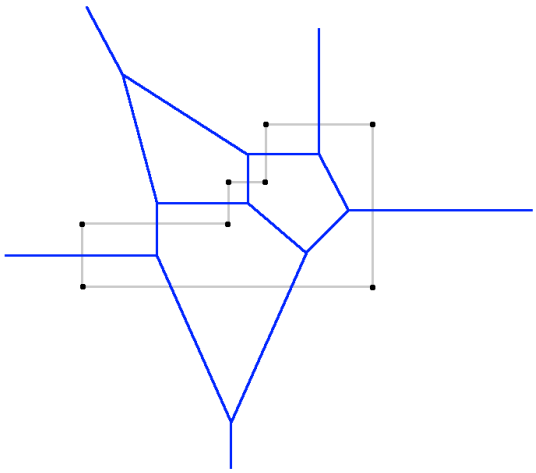
$\beta = 270°$.

Ak každý vrchol mnohouholníka ležiaci pri uhle typu α patrí konvexnému obalu daného mnohouholníka môžme určiť Voronoiov diagram mnohouholníka nasledovne:

Počet neohraničených stien Voronoiovho diagramu, bude zodpovedať počtu vrcholov pri uhloch typu α. Počet ohraničených Voronoiových mnohouholníkov bude rovný počtu vrcholov mnohouholníka pri uhloch typu β. Ortogonálny mnohouholník má vždy párny počet vrcholov. Môžeme preto určiť počet ohraničených Voronoiových mnohouholníkov ako $(\frac{n}{2}) - 2$ a počet neohraničených stien ako $(\frac{n}{2}) + 2$. Ako



Obr. 4.60. Ohraničené mnohouholníky pri uhloch typu β



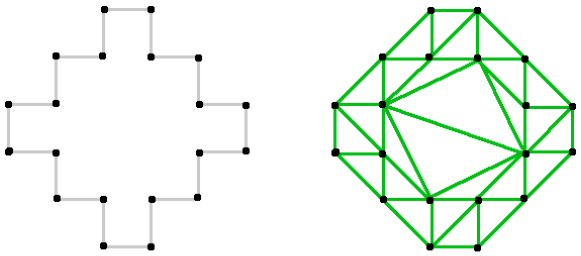
Obr. 4.61. Vrchol typu α mimo konvexný obal

možno vidieť na obrázku (obr. 4.60), počet bodov pri uhloch typu β je 8, rovnako ako ohraničených Voronoiových mnohouholníkov.

V prípade, kde nie všetky vrcholy pri konvexných uhloch mnohouholníka patria jeho konvexnému obalu si Voronoiov diagram zadefinujeme nasledovne:

Počet ohraničených Voronoiových mnohouholníkov bude zodpovedať počtu vrcholov mnohouholníka, ktoré nepatria jeho konvexnému obalu.

Určime si premennú a, ktorá zodpovedá počtu vrcholov pri uhloch typu α, ktoré nepatria konvexnému obalu mnohouholníka. Následne počet neohraničených stien bude zodpovedať počtu vrcholov patriacich jeho konvexnému obalu, čo si určíme pomocou predchádzajúcich vyjadrení, kde počet neohraničených stien bude zodpovedať predpisu $(\frac{n}{2}) + 2 - a$. Počet ohraničených Voronoiových mnohouholníkov bude určený vzťahom $(\frac{n}{2}) - 2 + a$.



Obr. 4.62. Ortogonálny mnohouholník so všetkými vrcholmi typu α na konvexnom obale

Ako možno vidieť na obrázku (obr. 4.61) počet vrcholov pri uhloch typu α je 6, avšak jeden z nich nepatrí konvexnému obalu mnohouholníka. Teda počet ohraničených Voronoiových mnohouholníkov sa zvýši o 1 a počet neohraničených stien sa zníži o 1.

Vytvoríme si Delaunayovu trianguláciu.

Keďže triangulácia n bodov je súvislý rovinný graf s n vrcholmi, má najviac $3n-6$ hrán. Každá z týchto hrán patrí najviac dvom trojuholníkom danej triangulácie.

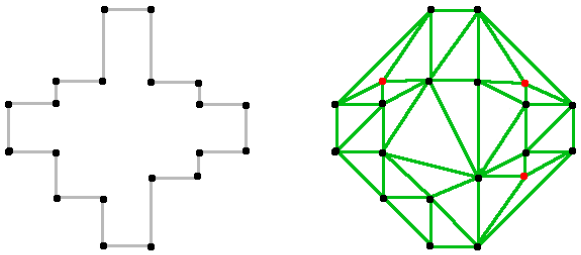
Ak každý z bodov pri uhloch typu α patrí konvexnému obalu množiny vrcholov tohto mnohouholníka, môžeme určiť počet trojuholníkov v danej triangulácii ako $(\frac{(3n-6)}{2}) - 1$. Na obrázku (obr. 4.62) máme ortogonálne konvexný mnohouholník, ktorého všetky vrcholy pri uhloch typu α patria konvexnému obalu množiny jeho vrcholov. Počet vrcholov je $n = 20$. Podľa predchádzajúceho tvrdenia môžeme určiť počet trojuholníkov triangulácie ako $(\frac{3*20-6}{2}) - 1 = 26$.

V prípade, že nie všetky vrcholy mnohouholníka, ktoré ležia pri uhloch typu α patria do konvexného obalu množiny vrcholov si zadefinujeme a ako počet takýchto vrcholov. Predpis pre počet trojuholníkov triangulácie bude mať tvar $(\frac{(3n-6)}{2}) - 1 + a$.

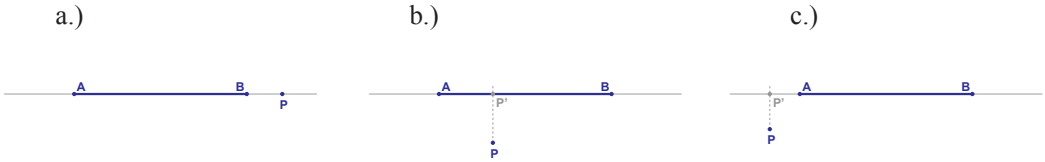
Ako možno vidieť na obrázku (obr. 4.63), červenou farbou sú vyznačené vrcholy, ktoré sú pri uhloch typu α, ale nepatria konvexnému obalu množiny vrcholov. Teda $a = 3$. Počet vrcholov mnohouholníka je $n = 20$. Podľa predchádzajúceho tvrdenia je počet trojuholníkov triangulácie určený ako $(\frac{3*20-6}{2}) - 1 + 3 = 29$. □

Cvičenie 4.20 (40 bodov). *Analyzujte, čo je množina bodov rovnako vzdialených od pevne daného bodu a úsečky v* $\mathbb{E}^2$.

Riešenie (Dušan Pácal, 26.10.2008): Majme v rovine body A, B a P. Pričom $A \neq B \neq P$. Nech $A = (a_x, a_y), B = (b_x, b_y), P = (p_x, p_y)$.



Obr. 4.63. Červené vrcholy sú mimo konvexného obalu



Obr. 4.64. Tri vzájomné polohy úsečky AB a bodu P.

Body A, B určujú úsečku a bod P na nej neleží. Majme tri prípady ako môžu byť bod a úsečka rozložené (pozri obr. 4.64):

- *a.)* body A,B,P ležia na jednej priamke,
- *b.)* kolmý priemet bodu P na priamku, na ktorej leží úsečka AB, leží na úsečke,
- *c.)* kolmý priemet bodu P na priamku, na ktorej leží úsečka AB, má s úsečkou prázdny prienik.

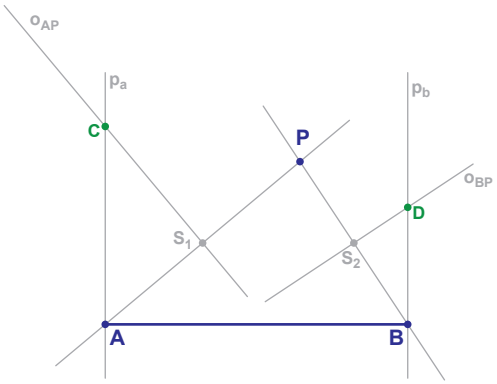
V celom riešení predpokladajme, že úsečka AB leží na x-ovej osi, bod A má menšiu x-ovú súradnicu ako bod B a bod P, ak neleží s bodmi A,B na jednej priamke, nech má kladú y-ovú súradnicu a teda nech sa nachádza "nad" úsečkou AB. Ak by body ležali v rovine inak ako predpokladáme, môžeme súradnicovú sústavu transformovať posunutím, otočením, prípadne zrkadlením aby platil náš predpoklad.
V prípade *a)* bude hľadanou množinou bodov priamka. ak $d(A,P) < d(B,P)$ bude ňou os bodov A,P, čiže priamka: $x = \frac{a_x+p_x}{2}$
ak $d(A,P) > d(B,P)$ bude ňou os bodov A,B, čiže priamka: $x = \frac{b_x+p_x}{2}$

$$d(A,P) > d(B,P) \qquad\qquad d(A,P) < d(B,P)$$

V prípade *b.)* sa bude naša hľadaná množina skladať z troch častí. Z dvoch polpriamok a jednej časti paraboly.

Nájdeme stredy úsečiek AP a BP, označme ich S_a a S_b. Týmito bodmi prechádzajú osi bodov A,P resp. B,P. Označme ich o_{AP} resp. o_{BP}.

Obr. 4.65. Prípad bodu ležiaceho na priamke AB.



Obr. 4.66. Význačné body a priamky v prípade bodu P nad úsečkou AB.

Ďalej majme priamky p_a, p_b. Budú to priamky rovnobežné s y-ovou osou, prechádzajúce koncovými bodmi A, B.

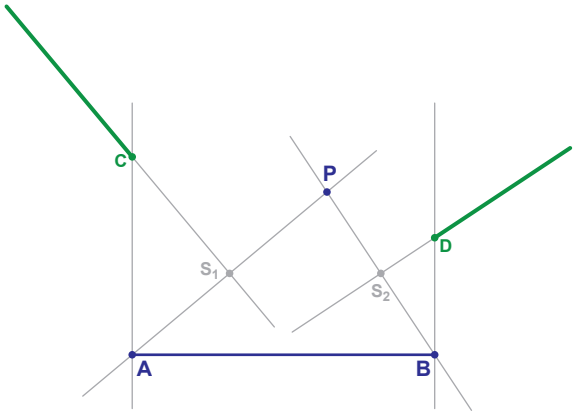
Nech bod C je prienikom priamok o_{AP} a p_a, a nech bod D je prienikom priamok o_{BP} a p_b.

Polpriamky $\overrightarrow{CS_1}^*$ (opačná ku $\overrightarrow{CS_1}$) a $\overrightarrow{DS_2}^*$ (opačná ku $\overrightarrow{CS_2}$) budú časťami hľadanej množiny. Hľadáme teda priamku, ktorá je kolmá na vektor $(P - A)$ a prechádza bodom S_1, čiže bodom $\frac{P+A}{2}$. A rovnako hľadáme aj priamku kolmú na vektor $(P - B)$, ktorá prechádza bodom S_2, čiže bodom $\frac{P+B}{2}$. Rovnice hľadaných polpriamok sú:

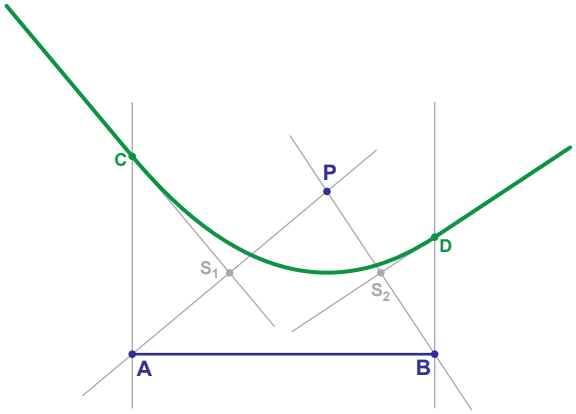
$(p_x - a_x)x + (p_y - a_y)y - [(p_x - a_x)(\frac{p_x+a_x}{2}) + (p_y - a_y)(\frac{p_y+a_y}{2})]$ pre $x \leq a_x$

$(p_x - b_x)x + (p_y - b_y)y - [(p_x - b_x)(\frac{p_x+b_x}{2}) + (p_y - b_y)(\frac{p_y+b_y}{2})]$ pre $x \geq b_x$

Teraz už len potrebujeme nájsť časť paraboly, ktorá spája nájdené polpriamky. Vieme, že vzdialenosť ľubovoľného bodu $Q = (q_x, q_y)$, ktorý je "nad" úsečkou od bodu P je $\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$. Jeho



Obr. 4.67. Polpriamky v riešení pre P nad AB.



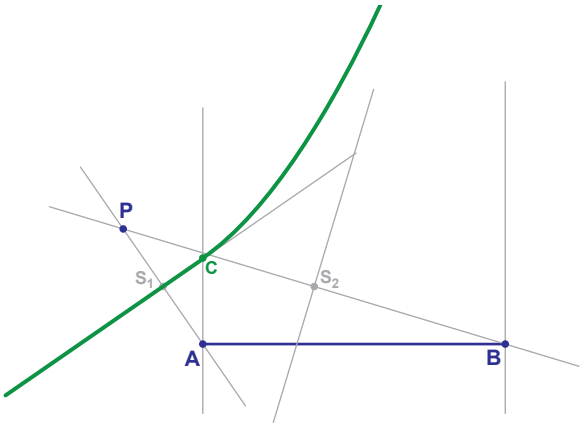
Obr. 4.68. Množina bodov rovnako vzdialených od P a AB.

vzdialenosť od úsečky AB je $|q_y - a_y|$. Z rovnosti týchto dvoch vzdialeností dostávame rovnicu paraboly: $(y - a_y) = \sqrt{(p_x - x)^2 + (p_y - y)^2}$ čiže rovnica hľadanej časti paraboly je:

$x(2p_x - x) + y(2p_y - 2a_y) + (a_y^2 - p_y^2 - p_x^2) = 0$, pre $x \in (a_x, b_x)$

Prípad *c.)* je identický s prípadom *b.)* s jedným malým rozdielom: V prípade ak $d(A,P) < d(B,P)$, tak hľadaná polpriamka nebude $\overrightarrow{CS_1}^*$, ale polpriamka $\overrightarrow{CS_1}$. A takisto ak $d(A,P) > d(B,P)$, tak hľadaná polpriamka nebude $\overrightarrow{DS_2}^*$, ale polpriamka $\overrightarrow{DS_2}$. Jej rovnica, ani interval sa však nemení. □

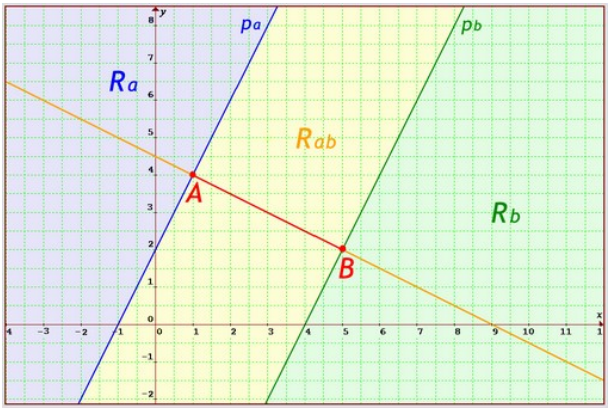
Riešenie (Kristína Lidayová, 18.11.2011): Majme v rovine 3 body P, A, B, pričom $P \neq A \neq B$. P označuje pevne daný bod a body A a B sú krajné body pevne danej úsečky. Predpokladajme, že bod P neleží na úsečke. V prvotnej úvahe predpokladajme, že riešime úlohu, čo je množina bodov rovnako vzdialených od pevne daného bodu P



Obr. 4.69. Prípad bodu P nad priamkou AB ale nie nad úsečkou AB.

a priamky, prechádzajúcej bodmi A, B. V tomto prípade je riešením parabola.

Teraz sa pozrime ako sa riešenie zmení, keď v zadaní namiesto celej priamky budeme uvažovať len jej časť, úsečku AB. Zostrojme si 2 pomocné priamky p_a, p_b, ktoré obe budú kolmé na úsečku AB, no p_a bude prechádzať bodom A a p_b bodom B. Tieto 2 priamky nám rozdelia rovinu na 3 časti. Označme si tieto 3 časti R_a, R_{ab} a R_b, tak ako je znázornené na obrázku.



Čiže oblasť R_a bude polrovina určená priamkou p_a (vrátane priamky p_a) neobsahujúca bod B, oblasť R_b bude polrovina určené priamkou p_b (vrátane priamky p_b) neobsahujúca bod A a posledná oblasť R_{ab} bude oblasť nachádzajúca sa medzi nimi.

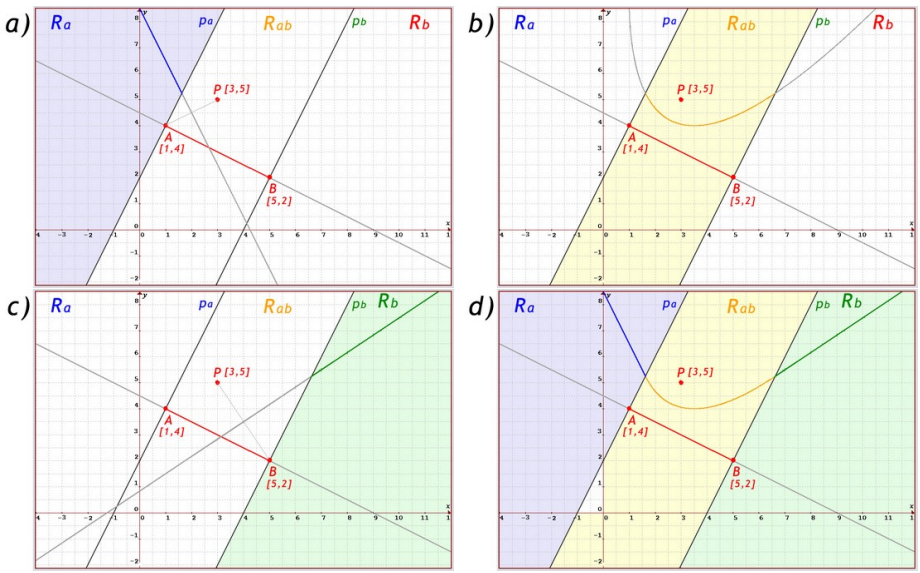
Naša hľadaná množina bodov bude v každej z 3 oblasti vyzerať inak.

Oblasť R_a:: V oblasti R_a sa z úsečky AB nachádza len bod A, preto riešením v tejto oblasti bude množina bodov rovnako vzdialených od 2 bodov P a A -> **os úsečky** PA .

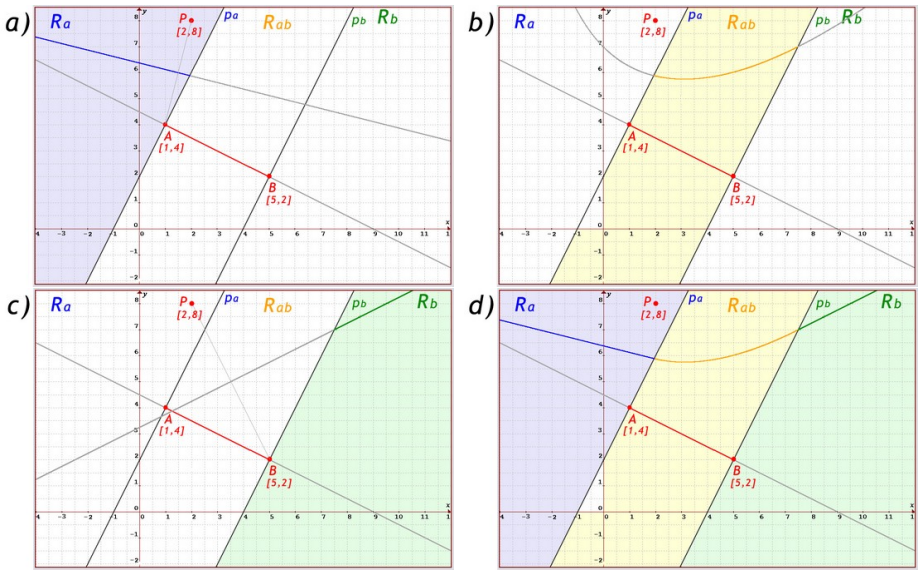
Oblasť R_b:: Podobne v oblasti R_b sa z úsečky AB nachádza len bod B, preto riešením v tejto oblasti bude množina bodov rovnako vzdialená od bodov P a B -> **os úsečky** PB.

Oblasť R_{ab}:: V oblasti R_{ab} leží celá úsečka AB bez krajných bodov, takže lokálne sa v oblasti R_{ab} úsečka AB tvári ako priamka prechádajúca bodmi A, B. Riešením v tejto oblasti teda bude množina bodov rovnako vzdialených od bodu P a priamky prechádzajúcej bodmi A a B-> **časť paraboly**.

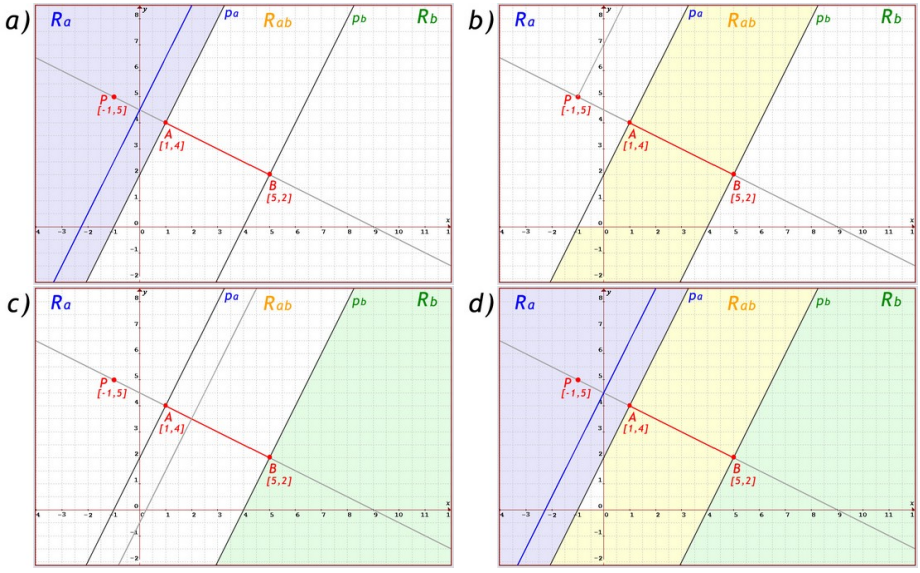
Mohlo by sa zdať, že existuje viacero prípadov riešenia podľa toho ako sú bod P a úsečka AB navzájom rozložené. No z nasledujúcich obrázkov je vidieť, že aj keď sa bod P nachádza na priamke prechádzajúcej bodmi A,B, či kdekoľvek v oblastiach R_a, R_{ab} a R_b, stále bude v každej oblasti riešením množina bodov, tak ako sme to vyššie popísali.



Príklad, keď P sa nachádza v oblasti R_{ab}. a)Hľadaná množina bodov pre oblasť R_a. b)Hľadaná množina bodov pre oblasť R_{ab}. c)Hľadaná množina bodov pre oblasť R_b. d)Výsledná hľadaná množina bodov.



Príklad, keď P sa nachádza v oblasti R_a. a)Hľadaná množina bodov pre oblasť R_a. b)Hľadaná množina bodov pre oblasť R_{ab}. c)Hľadaná množina bodov pre oblasť R_b. d)Výsledná hľadaná množina bodov.



Príklad, keď P sa nachádza na priamke prechádujúcej bodmi A, B. a)Hľadaná množina bodov pre oblasť R_a. b)Hľadaná množina bodov pre oblasť R_{ab}. c)Hľadaná množina bodov pre oblasť R_b. d)Výsledná hľadaná množina bodov.

□

Cvičenie 4.21 (50 bodov). *Analyzujte, čo je množina bodov rovnako vzdialených od dvoch úsečiek v $\mathbb{E}^2$.*

Riešenie (Martin Škorupa, 18.11.2008):

Najprv si ujasníme čo je množina všetkých bodov rovnako vzdialená (MVBRV) od:

a) bodov A a B: :
Ak $\mathbf{A} = \mathbf{B}$, tak celý priestor.
Ak $\mathbf{A} \neq \mathbf{B}$, tak os úsečky $\mathbf{AB}$.

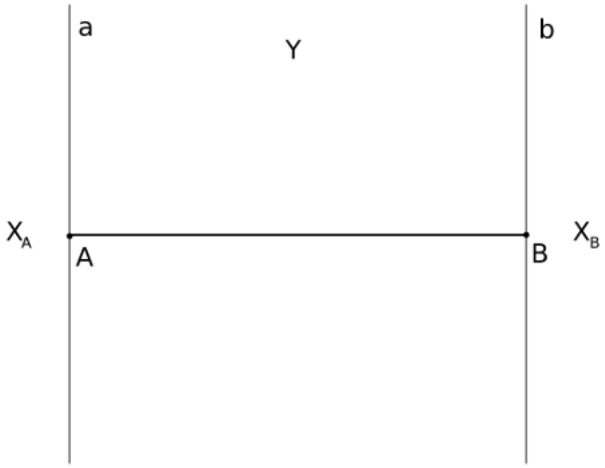
b) bodu A a priamky $\overline{\mathbf{B_1B_2}}$: :
Ak $\mathbf{A} \in \overline{\mathbf{B_1B_2}}$, tak priamka kolmá na $\overline{\mathbf{B_1B_2}}$ a prechádzajúca bodom $\mathbf{A}$.
Ak $\mathbf{A} \notin \overline{\mathbf{B_1B_2}}$, tak parabola definovaná týmto bodom a priamkou.

c) priamok $\overline{\mathbf{A_1A_2}}$ a $\overline{\mathbf{B_1B_2}}$: :
Ak $\overline{\mathbf{A_1A_2}} \nparallel \overline{\mathbf{B_1B_2}}$, tak os ich uhla.
Ak $\overline{\mathbf{A_1A_2}} \parallel \overline{\mathbf{B_1B_2}}$ a $\overline{\mathbf{A_1A_2}} = \overline{\mathbf{B_1B_2}}$, tak celý priestor.
Ak $\overline{\mathbf{A_1A_2}} \parallel \overline{\mathbf{B_1B_2}}$ a $\overline{\mathbf{A_1A_2}} \neq \overline{\mathbf{B_1B_2}}$, tak priamka $\overline{\mathbf{S_1S_2}}$, pričom $\mathbf{S_1}$ je stred $\mathbf{A_1B_1}$ a $\mathbf{S_2}$ je stred $\mathbf{A_2B_2}$.

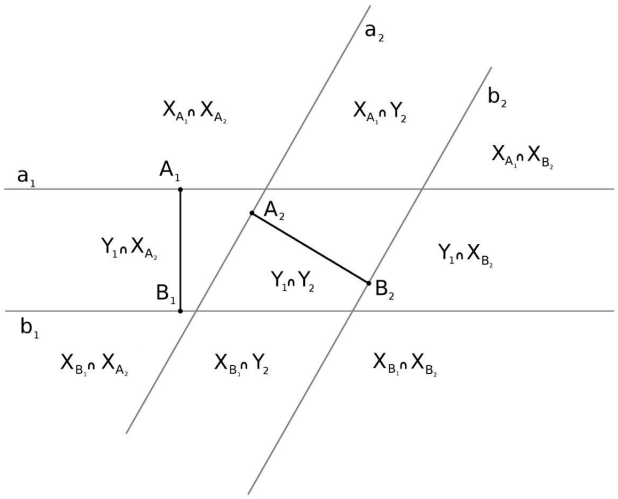
Majme úsečky $\mathbf{A_1B_1}$, $\mathbf{A_2B_2}$. A priamky:
$\mathbf{a_1}$: $\mathbf{a_1} \perp \mathbf{A_1B_1} \wedge \mathbf{A_1} \in \mathbf{a_1}$;
$\mathbf{b_1}$: $\mathbf{b_1} \perp \mathbf{A_1B_1} \wedge \mathbf{B_1} \in \mathbf{b_1}$;
$\mathbf{a_2}$: $\mathbf{a_2} \perp \mathbf{A_2B_2} \wedge \mathbf{A_2} \in \mathbf{a_2}$;
$\mathbf{b_2}$: $\mathbf{b_2} \perp \mathbf{A_2B_2} \wedge \mathbf{B_2} \in \mathbf{b_2}$.

Jedna úsečka rozdelí rovinu na 3 časti c oddelené priamkami $\mathbf{a}$ a $\mathbf{b}$(pozri obr. 4.70).
$\forall\, \mathbf{P} \in \mathbf{X_A}$, $|\mathbf{P\ AB}| = |\mathbf{P\ A}|$, teda pre časť $\mathbf{X_A}$ sa pri meraní vzdialenosti javí úsečka ako jeden bod.
$\forall\, \mathbf{P} \in \mathbf{X_B}$, $|\mathbf{P\ AB}| = |\mathbf{P\ B}|$, teda pre časť $\mathbf{X_B}$ sa pri meraní vzdialenosti javí úsečka ako jeden bod.
$\forall\, \mathbf{P} \in \mathbf{Y}$, $|\mathbf{P\ AB}| = |\mathbf{P}\ \overline{\mathbf{AB}}|$, teda pre časť $\mathbf{Y}$ sa pri meraní vzdialenosti javí úsečka ako priamka.

Pri dvoch úsečkách priamky $\mathbf{a_1}$, $\mathbf{b_1}$, $\mathbf{a_2}$, $\mathbf{b_2}$ rozdelia rovinu na 3, 4, 5, alebo 9 častí v závislosti od vzájomnej polohy úsečiek. Každá táto časť je prienikom jednej z $\mathbf{X_{A_1}}$, $\mathbf{Y_1}$, $\mathbf{X_{B_1}}$ a jednej z $\mathbf{X_{A_2}}$, $\mathbf{Y_2}$, $\mathbf{X_{B_2}}$ (pozri obr. 4.71). Teda môžu vniknúť časti:
$\mathbf{X_{A_1}} \cap \mathbf{X_{A_2}}$ - tu bude MVBRV od dvoch úsečiek totožná s MVBRV od dvoch bodov, prípad $\mathbf{a}$).
$\mathbf{X_{A_1}} \cap \mathbf{Y_2}$ - tu bude MVBRV od dvoch úsečiek totožná s MVBRV od bodu a priamky, prípad $\mathbf{b}$).
$\mathbf{X_{A_1}} \cap \mathbf{X_{B_2}}$ - tu bude MVBRV od dvoch úsečiek totožná s MVBRV od dvoch bodov prípad $\mathbf{a}$).
$\mathbf{Y_1} \cap \mathbf{X_{A_2}}$ - tu bude MVBRV od dvoch úsečiek totožná s MVBRV od bodu a priamky, prípad $\mathbf{b}$).
$\mathbf{Y_1} \cap \mathbf{Y_2}$ - tu bude MVBRV od dvoch úsečiek totožná s MVBRV od dvoch priamok, prípad $\mathbf{c}$).
$\mathbf{Y_1} \cap \mathbf{X_{B_2}}$ - tu bude MVBRV od dvoch úsečiek totožná s MVBRV od bodu a priamky, prípad $\mathbf{b}$).



Obr. 4.70. Rozdelenie roviny jednou úsečkou



Obr. 4.71. Rozdelenie roviny dvoma úsečkami

$\mathbf{X_{B_1} \cap X_{A_2}}$ - tu bude MVBRV od dvoch úsečiek totožná s MVBRV od dvoch bodov, prípad **a**).
$\mathbf{X_{B_1} \cap Y_2}$ - tu bude MVBRV od dvoch úsečiek totožná s MVBRV od bodu a priamky, prípad **b**).
$\mathbf{X_{B_1} \cap X_{B_2}}$ - tu bude MVBRV od dvoch úsečiek totožná s MVBRV od dvoch bodov, prípad **a**).

□

Cvičenie 4.22 (50 bodov). *Analyzujte, čo je množina bodov v* $\mathbb{E}^2$*, ktoré spĺňajú rovnicu*

$$\|\mathbf{x}-\mathbf{p}\|^2 - a_{\mathbf{p}}^2 = \|\mathbf{x}-\mathbf{q}\|^2 - a_{\mathbf{q}}^2,$$

kde $\mathbf{p}, \mathbf{q} \in \mathbb{E}^2$ *sú ľubovoľné, pevne zvolené,* $a_{\mathbf{p}}, a_{\mathbf{q}} \in \mathbb{R}$*. Dajte výsledku geometrickú interpretáciu.*

Riešenie (Kiss Gábor, 15.11.2010): Pozrime sa najprv na ľavú polovicu rovnosti. Ide vlastne o druhú mocninu vzdialenosti dvoch bodov mínus konštanta. Bod $\mathbf{x}$ nech má súradnice x a y. Bod $\mathbf{p}$ má súradnice p_x a p_y a $\mathbf{q}$ má q_x a q_y.Upravme si teda tento vzťah nasledovne
$\|\mathbf{x}-\mathbf{p}\|^2 - a_p^2 = \|\mathbf{x}-\mathbf{q}\|^2 - a_q^2$

$$\sqrt{(x-p_x)^2+(y-p_y)^2}^2 - a_p^2 = \sqrt{(x-q_x)^2+(y-q_y)^2}^2 - a_q^2$$

$$(x-p_x)^2+(y-p_y)^2 - a_p^2 = (x-q_x)^2+(y-q_y)^2 - a_q^2$$

Ak si všimneme táto rovnica predstavuje rovnice kružníc s polomerom a_p resp. a_q a stredom v bode $\mathbf{p}$ resp. $\mathbf{q}$.
Upravujme danú rovnosť ďalej nasledovne :

$$x^2-2xp_x+p_x^2+y^2-2yp_y+p_y^2-a_p^2 = x^2-2xq_x+q_x^2+y^2-2yq_y+q_y^2-a_q^2$$

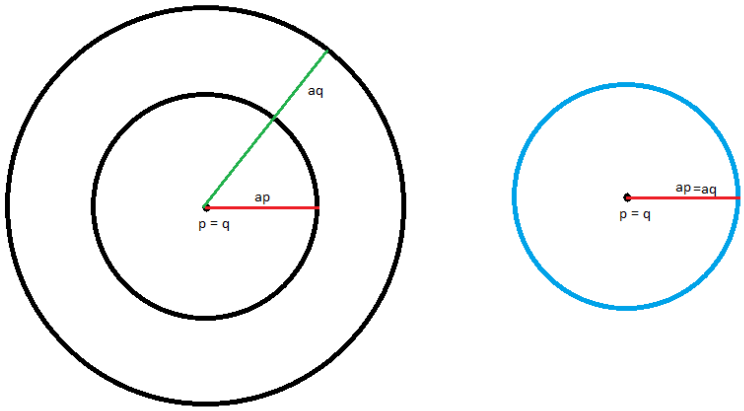
$$-2xp_x+p_x^2-2yp_y+p_y^2-a_p^2 = -2xq_x+q_x^2-2yq_y+q_y^2-a_q^2$$

$$-2xp_x+2xq_x-2yp_y+2yq_y+p_y^2+p_x^2-q_x^2-q_y^2+a_q^2-a_p^2 = 0$$

$$-2x(p_x-q_x)-2y(p_y-q_y)+p_y^2+p_x^2-q_x^2-q_y^2+a_q^2-a_p^2 = 0$$

Vzhľadom na to, že $p_x, q_x, p_y, q_y, a_q^2, a_p^2$ sú nám známe konštanty, tak nie je problém nahliadnuť , že riešením bude priamka.

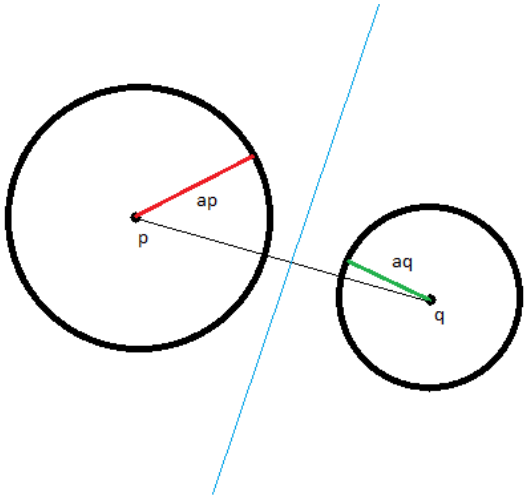
Najprv si vezmime body $\mathbf{p}$ a $\mathbf{q}$ totožné. V tomto prípade sú možné dva prípady- ak sú a_p a a_q taktiež rovnaké- v tomto prípade je riešením, celá rovina, alebo sú a_p a a_q rôzne- v tomto prípade rovnica nemá riešenie.



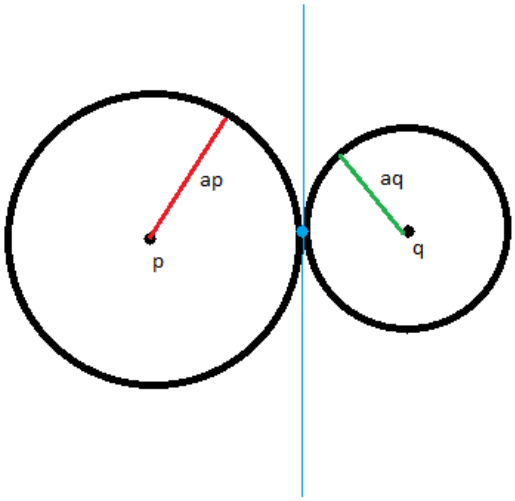
Teraz analyzujme prípad ak **p** a **q** nie sú totožné. V tomto prípade môže nastať 5 prípadov:
1) $a_p + a_q$ je menšie ako vzdialenosť (**p** , **q**)(označme si ju d). V tomto prípade je riešením priamka l , ktorej normálovým vektorom je vektor $\mathbf{q} - \mathbf{p}$. Priamka je kolmá na spojnicu bodov **p** a **q**. Ďalej nech d_1 a d_2 je možné určiť nasledovne :
$\frac{d_1 = d^2 + a_p^2 - a_q^2}{2d}$

a

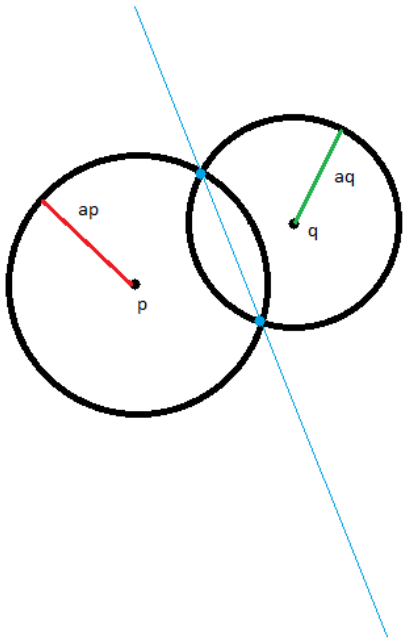
$d_2 = -\frac{d^2 + a_q^2 - a_p^2}{2d}$.
Potom platí, že $d = d_1 - d_2$
Pomocou d_1 a d_2 si teda môžeme určiť presnú polohu l tak, že si na spojnici bodov **p** a **q** nájdeme ten bod, ktorý je vo vzdialenosti d_1 od **p**. Takto získame bod l a normálový vektor l máme už určený vyššie.



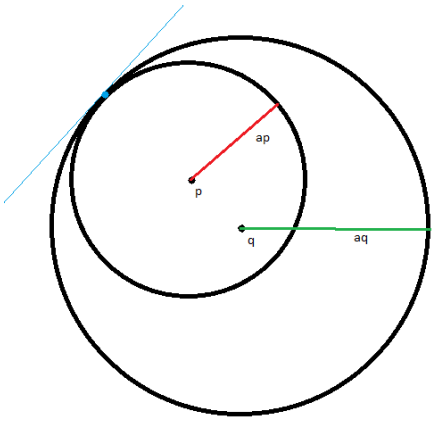
2) $a_p + a_q$ sa rovná vzdialenosti (**p** , **q**). Vtedy je riešením priamka, ktorá je dotyčnicou oboch kružníc a prechádza ich spoločným bodom.



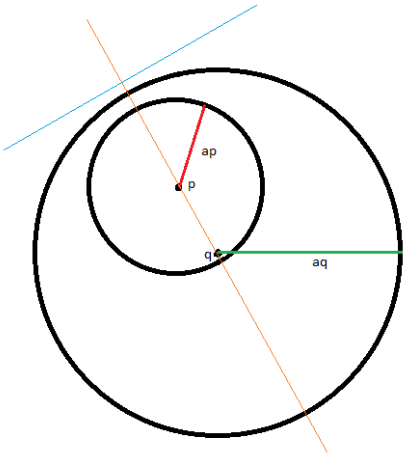
3) $a_p + a_q$ je väčšie ako vzdialenosť (**p** , **q**) a zároveň a_q menšie ako a_p + vzdialenosť (**p** , **q**).Riešením je priamka prechádzajúca cez spoločné body daných kružníc.



4)$a_p + a_q$ je väčšie ako vzdialenosť (**p** , **q**) a zároveň a_q sa rovná a_p + vzdialenosť (**p** , **q**). Riešením je dotyčnica ku kružniciam v ich spoločnom bode.

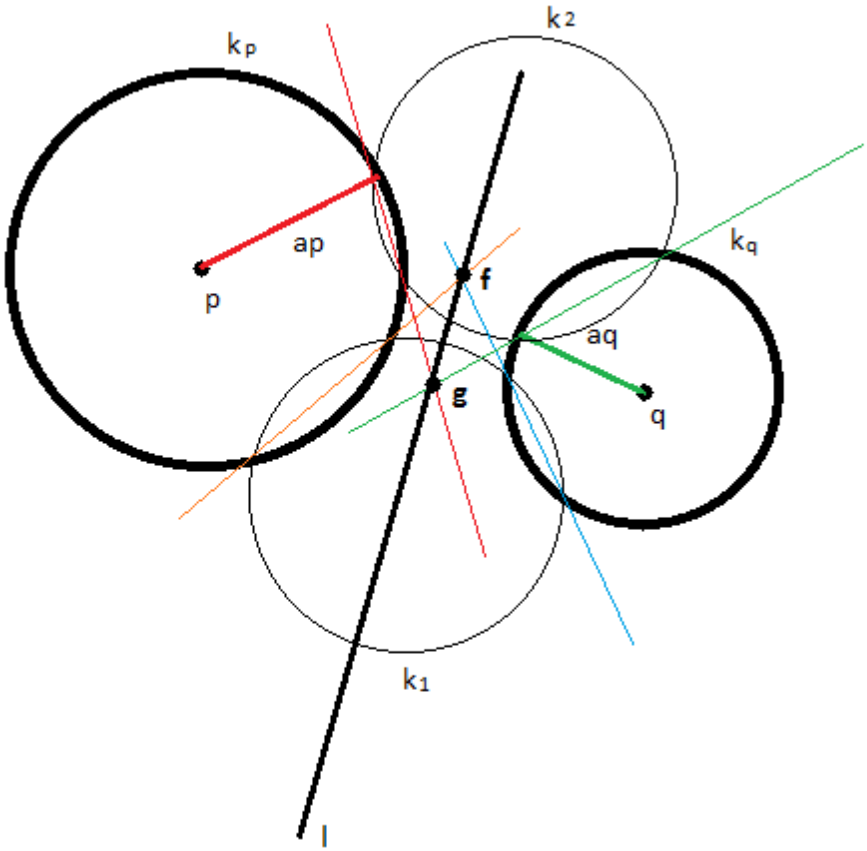


5) poslednou možnosťou je ak $a_p + a_q$ je väčšie ako vzdialenosť (**p** , **q**) a zároveň a_q je väčšie ako a_p + vzdialenosť (**p** , **q**). Výsledná priamka bude kolmá na priamku prechádzajúcu cez **p** a **q**- označme si ju l, ako v prvom prípade. Rovnako ako v prvom prípade si môžeme vypočítať d_1 a d_2 a pomocou nich určiť presnú polohu priamky l.



Poznámka k prípadom 1 a 5: Označme si zadané kružnice ako k_p a k_q V týchto prípadoch sa dá l zostrojiť nasledovným postupom:

(1) Ku k_p a k_q nakreslíme ďalšiu kružicu k_1 tak, aby platilo, že $k_p \cap k_1$ má dva body. Označme si ich a_1 a a_2. Ďalej nech $k_q \cap k_1$ má dva body - b_1 a b_2
(2) Vytvoríme k_2 tak aby $k_p \cap k_2$ malo dva body - c_1 a c_2 a $k_q \cap k_2$ malo dva body - d_1 a d_2
(3) Vytvorme priamky prechádzajúce cez a_1, a_2; b_1, b_2; c_1, c_2 a d_1, d_2
(4) Určime si prienik priamok a_1, a_2 b_1, b_2 - označme si ho f
(5) Určime si prienik priamok c_1, c_2 d_1, d_2 - označme si ho g
(6) Priamka prechádzajúca bodmi f a g je našou hľadanou priamkou.



□

Cvičenie 4.23 (50 bodov). *Ukážte, že platia inklúzie*

$$CP \subset EMST \subset RNG \subset GG \subset DT.$$

Riešenie (Jana Hlinková, 18.11.2008):

Najprv uvediem definície jednotlivých grafov a potom sa pustím do dokazovania platnosti inklúzií. Nech $S = \{p_1, p_2, \ldots, p_n\}$ je množina n bodov v E^d. Táto množina tvorí množinu vrcholov každého z uvedených grafov. Grafy sa teda líšia iba množinou hrán. Pri definovaní množiny hrán sa využíva euklidovská vzdialenosť (ďalej už iba vzdialenosť) bodov (vrcholov). Množinu hrán grafu G budem označovať $E(G)$, hranu medzi vrcholmi p_i, p_j budem označovať $h = (p_i, p_j)$ a euklidovskú vzdialenosť $d(p_i, p_j)$.

Definícia 4.2. ***CP (Closest Pair):*** *Hranou sú spojené také vrcholy p_i, p_j, $i \neq j$, ktoré majú globálne minimálnu vzdialenosť a graf*

má len jednu hranu. Teda platí, že:

$$(p_i, p_j) \in E(CP) \Longrightarrow d(p_i, p_j) = \min\{d(p_k, p_l) |\ k \neq l, k, l = 1, \ldots, n\}.$$

DEFINÍCIA 4.3. ***EMST (Euclidean Minimum Spannig Tree):*** *EMST je minimálna kostra vzhľadom na euklidovskú vzdialenosť. Je to súvislý graf bez cyklov.*

DEFINÍCIA 4.4. ***RNG (Relative neighborhood graph):*** *Hranou sú spojené všetky dvojice vrcholov p_i, p_j, ktoré sú si relatívne blízko. Dva vrcholy sú si relatívne blízko, ak*

$$d(p_i, p_j) \leq \max\{d(p_i, p_k), d(p_j, p_k) |\ \forall k \neq i, j\}.$$

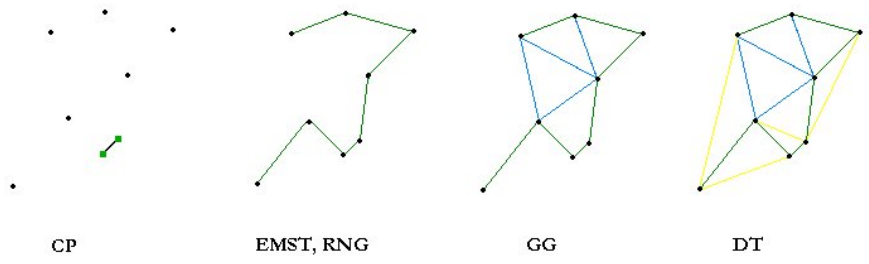
Prakticky túto podmienku môžeme interpretovať nasledovne: dva vrcholy sú si relatívne blízko, ak v priesečníku kruhov (d-rozmerných gulí) so stredmi v p_i, p_j a polomerom $d(p_i, p_j)$ neleží žiaden iný vrchol z S.

DEFINÍCIA 4.5. ***GG (Gabriel Graph):*** *Hranou sú spojené všetky dvojice vrcholov p_i, p_j, pre ktoré platí:*

$$d(p_i, p_j) \leq \sqrt{d^2(p_i, p_k) + d^2(p_j, p_k)} \qquad \forall k \neq i, j.$$

Teda neexistuje vrchol $p_k \in S$, ktorý by ležal v kruhu (d-rozmernej guli) s priemerom $d(p_i, p_j)$, prechádzajúcej vrcholmi p_i, p_j.

DEFINÍCIA 4.6. ***DT (Delaunay Triangulation):*** *Taká triangulácia, že žiaden bod $p_k \in S$ neleží v opísanej kružnici (d-rozmernej guli) trojuholníka (simplexu) triangulácie, ktorého nie je vrcholom.*



OBR. 4.72. Ukážka CP, EMST, RNG, GG a DT.

K danej množine bodov $S = \{p_1, p_2, \ldots, p_n\}$ existuje viacero rôznych grafov CP, EMST, RNG, GG a DT. Uvedené dôkazy majú byť preto chápané nasledovne: pre dokazovanú inklúziu

$$\text{typGrafu1} \subset \text{typGrafu2}$$

ukážeme, že pre daný graf typu typGrafu1 existuje graf typu typGrafu2, ktorý ho obsahuje ako podgraf (poz. obrázok 4.72).

VETA 4.1. $CP \subset EMST$

Dôkaz: Nech hrana $h = (p_i, p_j)$ patri $E(CP)$, teda p_i, p_j je jedna naj-bližšia dvojica bodov množiny S. Ukážeme, že existuje taká minimálna kostra množiny S, ktorá túto hranu obsahuje (teda obsahuje daný CP ako podgraf).
Nech T je ľubovoľná minimálna kostra množiny S. Ak T obsahuje hranu $h = (p_i, p_j)$, tak sme spokojní - existuje EMST ktorá obsahuje CP ako podgraf. Nech teda T hranu $h = (p_i, p_j)$ neobsahuje. V tejto kostre existuje cesta z p_i do p_j, pričom všetky hrany ležiace na tejto ceste majú rovnakú a zároveň minimálnu dĺžku rovnú vzdialenosti bodov p_i, p_j. (Totiž ak by niektorá hrana mala väčšiu dĺžku, mohli by sme ju nahradiť hranou $h = (p_i, p_j)$, pričom by sme dostali "minimálnejšiu" kostru a spor s predpokladom, že T je EMST.) Nahradením ľubovoľnej hrany z tejto cesty hranou $h = (p_i, p_j)$ dostaneme EMST obsahujúcu daný CP ako podgraf. □

Veta 4.2. $EMST \subset RNG$

Dôkaz: Nech $h = (p_i, p_j)$ je hrana grafu EMST. Ukážeme, že táto hrana patrí aj grafu RNG, a to tak, že ukážeme platnosť podmienky relatívnej blízkosti

$$d(p_i, p_j) \leq \max\{d(p_i, p_k), d(p_j, p_k) |\ \forall k \neq i, j\}.$$

Platnosť podmienky dokážeme sporom. Nech hrana $h = (p_i, p_j) \in E(EMST)$ a nech existuje vrchol $p_k \in S$, že platí pre dĺžky hrán $d(p_i, p_j) > \max\{d(p_i, p_k), d(p_j, p_k)\}$ (teda pre vrcholy p_i, p_j neplatí podmienka relatívnej blízkosti). Z uvedenej nerovnosti vyplýva, že platia nerovnosti $d(p_i, p_j) > d(p_i, p_k)$ a $d(p_i, p_j) > d(p_j, p_k)$. V takomto EMST nemôže byť ani hrana (p_i, p_k) a ani hrana (p_j, p_k). Ak by totiž EMST obsahoval napr. hranu (p_i, p_k), mohli by sme namiesto hrany (p_i, p_j) dať hranu (p_j, p_k) a dostali by sme novú minimálnejšiu kostru (a teda spor s tvrdením, že náš graf naozaj je EMST).

Z toho vyplýva, že vrchol p_k je v našom EMST spojený hranou s nejakým iným vrcholom p_l. Teraz ukážeme, že na základe predpokladu $d(p_i, p_j) > \max\{d(p_i, p_k), d(p_j, p_k)\}$ vieme zostrojiť minimálnejšiu kostru, a to nasledovne: vynecháme hranu (p_i, p_j). Ak bol jeden z vrcholov p_i, p_j v pôvodnej kostre listom (samozrejme, listom mohol byť iba jeden z týchto vrcholov), spojíme ho s vrcholom p_k. Ak nie, s vrcholom p_k spojíme ten vrchol, ktorý je k nemu bližšie.

Takto sme prišli k sporu s predpokladom, že náš pôvodný graf bol EMST. Preto pre každú hranu z EMST musí platiť podmienka relatívnej blízkosti $d(p_i, p_j) \leq \max\{d(p_i, p_k), d(p_j, p_k) |\ \forall k \neq i, j\}$, a teda každá hrana EMST musí byť aj hranou RNG. □

Veta 4.3. $RNG \subset GG$

Dôkaz: Pre každú hranu $h = (p_i, p_j)$ grafu RNG platí:

$$d(p_i, p_j) \leq \max\{d(p_i, p_k), d(p_j, p_k) | \ \forall k \neq i, j\}.$$

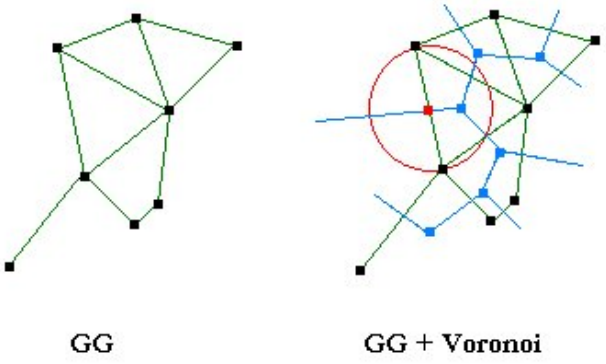
Jednoducho ukážeme, že potom platí aj vzťah

$$d(p_i, p_j) \leq \sqrt{d^2(p_i, p_k) + d^2(p_j, p_k)} \qquad \forall k \neq i, j,$$

a teda hrana $h = (p_i, p_j)$ patrí aj grafu GG. Nech platí nerovnosť $d(p_i, p_j) \leq \max\{d(p_i, p_k), d(p_j, p_k)\}$. Bez ujmy na všeobecnosti, nech $d(p_i, p_k) = \max\{d(p_i, p_k), d(p_j, p_k)\}$. Potom

$$d(p_i, p_j) \leq d(p_i, p_k) = \sqrt{d^2(p_i, p_k)} \leq \sqrt{d^2(p_i, p_k) + d^2(p_j, p_k)}.$$

□



Obr. 4.73. Ukážka k dôkazu GG ⊂ DT.

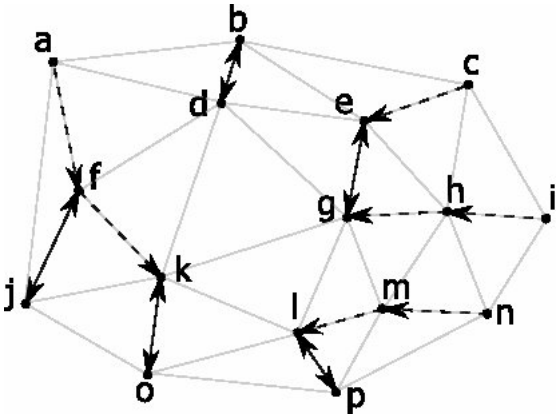
Veta 4.4. *GG ⊂ DT*

Dôkaz: Pri dôkaze tejto inklúzie využijeme dualitu DT a Voronoiovho diagramu. Hrana $h = (p_i, p_j)$ patrí do $E(DT)$ práve vtedy, keď bunky Voronoiovho diagramu bodov (generátorov) p_i, p_j majú spoločnú hranicu (teda ide o susedné bunky). Hranicu týchto buniek tvoria body, ktoré sú rovnako vzdialené od p_i a p_j a zároveň neexistuje generátor p_k, ku ktorému by boli bližšie.

Teraz ukážeme, že každá hrana z $E(GG)$ je aj v $E(DT)$. Nech teda hrana $h = (p_i, p_j)$ patrí GG. Potom platí

$$d(p_i, p_j) \leq \sqrt{d^2(p_i, p_k) + d^2(p_j, p_k)} \qquad \forall k \neq i, j,$$

a teda v kruhu s priemerom $d(p_i, p_j)$ prechádzajúcim bodmi p_i, p_j neleží žiaden iný vrchol z S. Ľahko sa môžeme presvedčiť, že stred tohto kruhu leží na hranici buniek Voronoiovho diagramu prislúchajúcich generátorom p_i, p_j: keďže v spomínanom kruhu neleží žiaden iný generátor okrem p_i, p_j, znamená to, že stred tohto kruhu leží spomedzi generátorov



Obr. 4.74. All Nearest Neighbors

S najbližšie práve k týmto dvom generátorom, pričom od p_i a od p_j má rovnakú vzdialenosť $d(p_i, p_j)/2$, a teda leží na hranici buniek p_i a p_j Voronoiovho diagramu (obrázok 4.73). Teda existuje bod, ktorý patrí hranici týchto dvoch buniek, čo teda znamená, že tieto dve bunky sú susedné. Z duality medzi Voronoiovým diagramom a DT potom vyplýva, že body p_i, p_j sú v DT spojené hranou. □

Veta 4.5. $CP \subset EMST \subset RNG \subset GG \subset DT$

□

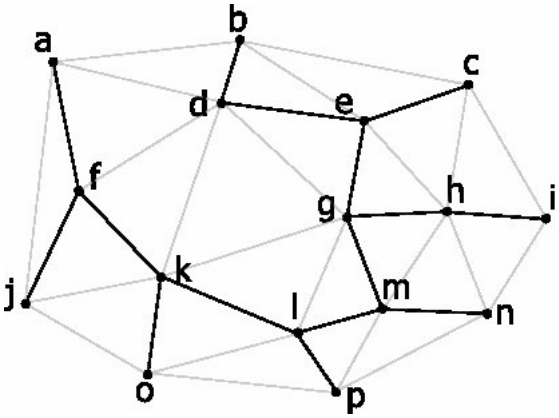
Cvičenie 4.24 (30 bodov). *Dokážte, že obojsmerné hrany grafu ANN sa nachádzajú v EMST.*

Riešenie (Elena Dušková, 12.11.2008):

Graf ANN (All Nearest Neighbors - všetci najbližší susedia):

Najbližší sused je relácia na množine P obsahujúcej n bodov roviny. Graf ANN zostrojíme tak, že pre každý z daných n bodov nájdeme najbližší z ostatných $n-1$. Bod B je najbližší sused bodu A (ozn. $A \to B$), ak vzdialenosť $d(A,B) = \min_{C \in P} d(A,C)$. Relácia najbližšieho suseda nie je symetrická. Pokiaľ je B najbližším susedom A a zároveň A je najbližším susedom B, tak dostaneme obojsmernú hranu v grafe ANN (pozri obr. 4.74). Nájdenie všetkých najbližších susedov množiny n bodov si vyžaduje čas $T : \mathcal{O}(n \log n)$.

Graf EMST (Euclidean Minimum Spanning Tree - minimálna kostra grafu v $\mathbb{E}^n$): EMST je minimálnou kostrou grafu n bodov v euklidovskej rovine. Z kompletného grafu (n vrcholov a $\frac{n(n-1)}{2}$ hrán) v ňom budú použité všetky vrcholy a len $(n-1)$ hrán tak, aby sme v novom grafe mali 1 komponent súvislosti, ktorý neobsahuje cykly (je to strom) a súčet dĺžok všetkých hrán je minimálny možný (pozri obr. 4.75). Takýchto riešení môže existovať viac.



Obr. 4.75. EMST vytvorený z DT

Na nájdenie minimálnej kostry v grafe existuje viacero algoritmov. Na to aby sme našli EMST nemusíme vychádzať z kompletného grafu, korektný výsledok dostaneme aj pri tvorbe kostry z Delaunayovej triangulácie. Dôkaz, že EMST $\subset$ DT je v cvičení 4.23. Jedným z riešení na tvorbu EMST je Primov algoritmus.

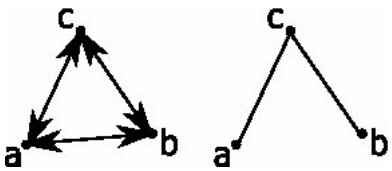
EMST množiny P obsahujúcej n bodov v rovine možno vypočítať z DT(P) v optimálnom čase $T : \mathcal{O}(n)$. Keďže DT(P) vieme dostať v čase $T : (n \log n)$, tak EMST množiny P (n bodov v rovine) môžeme vypočítať v čase $T : \mathcal{O}(n \log n)$.

Primov algoritmus: Majme DT(P). Označme si vrcholy (v našom príklade $a, b, \ldots, p$) a zaznačme si hodnoty hrán (zistené vzdialenosti) medzi vrcholmi, ktoré spája DT. Pri tvorbe EMST budeme narábať s lesom stromov S. Na začiatku je les tvorený množinou všetkých n bodov, bez hrán. Vyberieme si jeden vrchol z množiny, napr. p. Od tohto bodu bude strom narastať pridávaním vrcholov, až na kostru celého grafu.

Najkratšia z hrán $p-o$, $p-l$, $p-m$, $p-n$, je hrana $p-l$, čiže strom T rozšírime o vrchol l a hranu $p-l$. Od T k ostatným vrcholom máme 7 možností, najkratšia z hrán je $l-m$. V ďalšom kroku pridáme hranu $m-g$, aj keď $m-p$ je kratšia hrana, ale bod p už stromu T patrí, vznikol by cyklus. Potom postupne pridávame hrany $g-e$, $g-h$, $h-i$, $m-n$, $e-c$, $e-d$, $d-b$, $l-k$, $k-o$, $k-f$, $f-j$, $f-a$. Skončíme, keď sú všetky vrcholy súčasťou T. Máme minimálnu kostru zo 16 vrcholov a 15 hrán.

Všeobecne:

```
Opakuj kým je počet stromov väčší ako 1
   vyber strom T v lese S
   nájdi minimálnu hranu UV zo všetkých nepoužitých hrán,
      kde U je z T a V je z S\T.
   hranou UV pripoj strom T obsahujúci vrchol V k stromu T
```



Obr. 4.76. EMST, ktorý neobsahuje všetky obojsmerné hrany z ANN.

Zhrnutie: Podľa tohto postupu sa obojsmerné hrany grafu ANN použijú, buď pri pripájaní bodu A k bodu B alebo bodu B k bodu A, pretože Primov algoritmus hľadá vždy najkratšiu hranu z nepoužitých.

Problém nastáva v prípade, keď pri vytváraní ANN sú k nejakému bodu A rovnako najbližšie 2 body (B, C), obdobne pre body B a C máme rovnostranný trojuholník ABC (pozri obr. 4.76). Ak si do grafu ANN zaznačujeme všetky minimálne hrany z bodu, dostaneme cyklus, ktorý v kostre EMST nemôže byť, čiže jedna obojsmerná hrana sa nepoužije. Ak by sa do ANN pridávala z nejakého vrcholu vždy len jedna (ľubovoľná) hrana zo všetkých minimálnych, tento problém by nenastal. □

Cvičenie 4.25 (50 bodov). *Nech* $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \subset \mathbb{E}^2$, $n \in \mathbb{N}$. *Nech* $k \in \mathbb{N}$, $1 \leq k \leq n-1$. *Pre* $Q \subset P$, $Q = \{\mathbf{p}_{i_1}, \ldots, \mathbf{p}_{i_k}\}$ *označme* $\bar{Q} = P - Q$ *a definujme* $V(Q) = \{\mathbf{x} \in \mathbb{E}^2 : \|\mathbf{x} - \mathbf{p}_i\| \leq \|\mathbf{x} - \mathbf{p}_j\| \quad \textit{pre všetky} \quad \mathbf{p}_i \in Q, \mathbf{p}_j \in \bar{Q}\}$. *Opíšte množinu* $V(Q)$. *Nájdite príklad, keď je pre nejaké* Q *množina* $V(Q)$ *prázdna. Ak to nie je možné, zdôvodnite prečo. Množiny* $\{V(Q), Q \subset P\}$ *tvoria tzv.* Voronoiov diagram rádu k. *Nakreslite ho pre niekoľko bodov.*

Riešenie (Martina Bátorová, 19.11.2008):

V rovine máme zadanú množinu P obsahujúcu n bodov. Vyberieme spomedzi nich vlastnú neprázdnu podmnožinu Q s k bodmi, zvyšných $(n-k)$ bodov označíme $\bar{Q}$. Skonštruujeme množinu $V(Q)$, kam vložíme body roviny pre ktoré platí, že sú k bodom Q bližšie než k ľubovoľnému bodu z $\bar{Q}$. Postupne takto z P vyberáme všetky možné k-tice a konštruujeme množiny "blízkych bodov". Vznikne delenie roviny, ktoré sme nazvali Voronoiov diagram rádu k.

$k = 1$

Pre prípad $k = 1$ vyberáme z n bodov zakaždým práve jeden bod. Vzniknutý Voronoiov diagram rádu 1 je nám dobre známy "obyčajný" Voronoiov diagram. Jeho konštrukciu si uvádzať nebudeme.

Odpoveď na otázku, či môže byť Voronoiov región rádu 1 prázdna množina, je nie. Navyše, nielen že jednotlivé regióny sú neprázdne, ale majú i neprázdne vnútro. Dôkaz vyplýva priamo z konštrukcie: keďže uvažujeme vstupné vrcholy ako rôzne body, pre ľubovoľnú dvojicu vieme skonštruovať priamku kolmú na ich spojnicu, na ktorej neleží

ani jeden z bodov (takáto priamka uvažované body oddeľuje). Teda vieme nájsť také (otvorené) okolie bodu $\mathbf{p}_i \in P$, že všetky body v ňom sú bližšie k nemu než k ľubovoľnému inému bodu vstupnej množiny (stačí skonštruovať osi bodov $\mathbf{p}_i, \mathbf{p}_j \in P, i \neq j$, pre každú os vybrať polrovinu, v ktorej leží $\mathbf{p}_i$ a urobiť prienik všetkých takýchto polrovín. Vznikne konvexná množina s neprázdnym vnútrom).

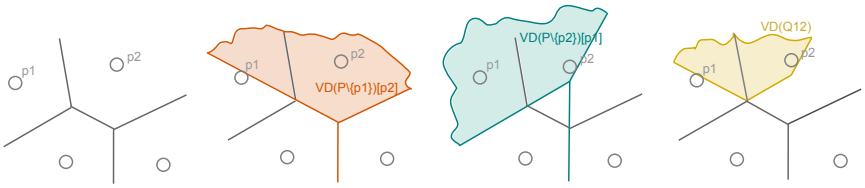
$k = 2$

Pre prípad $k = 2$ vyberáme dvojice bodov. Od Voronoiovho regiónu rádu 2 očakávame, že všetky body v ňom sú bližšie k jeho generátorom, než k iným bodom množiny P. Matematicky by sa dá toto (okrem definície zo zadania) pre $Q = \{\mathbf{p}_1, \mathbf{p}_2\}$ zapísať aj ako

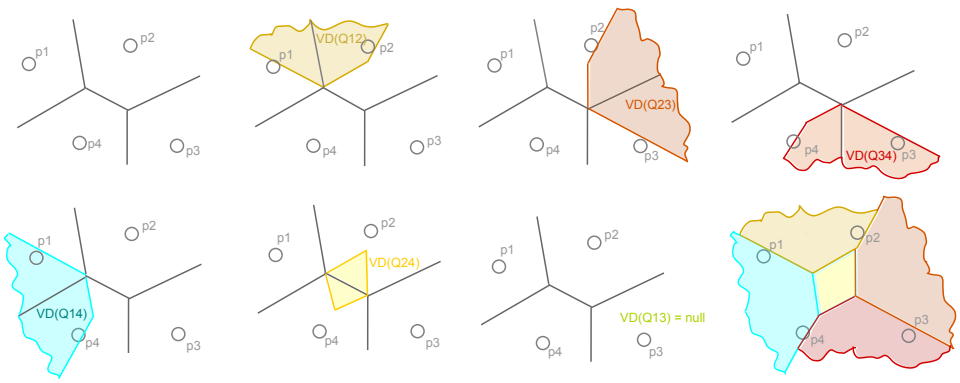
$$V(Q) = \{\mathbf{x} \in \mathbb{E}^2 : \max\{\|\mathbf{x} - \mathbf{p}_1\|, \|\mathbf{x} - \mathbf{p}_2\|\} \leq \min_{\mathbf{p}_j \in \bar{Q}} \|\mathbf{x} - \mathbf{p}_j\|\}.$$

Množinu $V(Q)$ by sme mohli skonštruovať napr. takto: v prvom kroku zostrojíme Voronoiov diagram rádu 1. Tak nám vzniknú oblasti, ktorých body sú najbližšie k jednotlivým bodom množiny P. Pre vybranú dvojicu bodov $Q = \{\mathbf{p}_i, \mathbf{p}_j\}, i \neq j$ určíme "maximálnu zónu dosahu": najprv skonštruujeme Voronoiov región bodu $\mathbf{p}_i$ bez uvažovania bodu $\mathbf{p}_j$ a potom opačne. Ak by sme tieto regióny zjednotili, vznikla by množina bodov, ktoré sú blízko *aspoň* jednému bodu $\mathbf{p}_i, \mathbf{p}_j$. Ak však urobíme ich prienik, zostanú body, ktoré sú blízko *iba* k $\mathbf{p}_i, \mathbf{p}_j$, a od každého ďalšieho generátora sú ďalej - čím vznikne práve hľadaná množina bodov $V(Q_{ij})$. Vyberieme novú kombináciu bodov $\mathbf{p}_k, \mathbf{p}_l, \{i, j\} \neq \{k, l\}$ a opakujeme predchádzajúci postup, až kým nezvolíme všetkých $\binom{n}{2}$ možných dvojíc:

(1) $VD(P)$
(2) $Q_{ij} := \{\mathbf{p}_i, \mathbf{p}_j\} \subset P$
(3) $V(Q_{xy})$:
  (a) $VD(P \backslash \{\mathbf{p}_i\})$
  (b) $VD(P \backslash \{\mathbf{p}_j\})$
  (c) $V(Q_{ij}) := VD(P \backslash \{\mathbf{p}_i\})[\mathbf{p}_j] \cap VD(P \backslash \{\mathbf{p}_j\})[\mathbf{p}_i]$.
(4) $Q_{ij} := \{\mathbf{p}_k, \mathbf{p}_l\} \subset P; \{i, j\} \neq \{k, l\}$



Obr. 4.77. Na obrázku vľavo je krok 1 a 2. Na ďalších troch je zobrazená konštrukcia $V(Q_{12})$ ako je popísaná v krokoch $3a - c$.



OBR. 4.78. Na obrázkoch sú postupne zobrazené množiny $V(Q_{ij}), i, j \in \{1, 2, 3, 4\}, i \neq j$. Posledný obrázok je výsledné delenie roviny $V(Q)$.

Pozn.: Zápisom $VD(P\backslash\{\mathbf{p}_i\})[\mathbf{p}_j]$ máme na mysli, že skonštruujeme Voronoiov diagram množiny $P\backslash\{\mathbf{p}_i\}$ a berieme do úvahy región, ktorý prislúcha bodu $\mathbf{p}_j$.

Pre $k = 2$ už otázka neprázdnosti Voronoiovej oblasti nie je taká priamočiara ako v prípade diagramu rádu 1. Pre $k = 1$ sme si ukázali, že vzniknutá množina je konvexná a jej vnútro je neprázdne. Je ľahké nahliadnuť, že i Voronoiov diagram rádu 2 je konvexná množina (vznikne ako ako konečné zjednotenie konečných prienikov konvexných množín). Vnútro takéhoto objektu však nemusí byť neprázdne, čiže musíme uvažovať všetky vyššie popísaným spôsobom skonštruovateľné konvexné množiny:

(1) $V(Q_{ij})$ je polygonálny objekt
(2) $V(Q_{ij})$ je bod
(3) $V(Q_{ij})$ je prázdna množina

Jednotlivé situácie nastávajú podľa toho, ako "ďaleko" sú od seba vzdialené $\mathbf{p}_i, \mathbf{p}_j$.

$k \geq 3$

Vyššie spomenutý algoritmus sa dá zovšeobecniť pre Q s $2 < k \leq (n - 1)$ prvkami. Prepisy pre body patriace do jednotlivých Voronoiových regiónov rádu k môžeme zapísať ako

$$V(Q) = \{\mathbf{x} \in \mathbb{E}^2 : \|\mathbf{x} - \mathbf{p}_{i_l}\| \leq \|\mathbf{x} - \mathbf{p}_j\|, \mathbf{p}_j \in \bar{Q}, \mathbf{p}_{i_l} \in Q, l = 1, \ldots, k\}$$

resp.

$$V(Q) = \{\mathbf{x} \in \mathbb{E}^2 : \max_{\mathbf{p}_{i_l} \in Q} \|\mathbf{x} - \mathbf{p}_{i_l}\| \leq \min_{\mathbf{p}_j \in \bar{Q}} \|\mathbf{x} - \mathbf{p}_j\|\}.$$

V algoritme potom pre počítame pre jednotlive k-tice $Q_i = \{\mathbf{p}_{i_1}, \ldots, \mathbf{p}_{i_k}\}$ bodov prislúchajúci región ako

$$V(Q_i) = \bigcap_{l=1}^{k} V(P \backslash Q_i \cup \{\mathbf{p}_{i_l}\})[\mathbf{p}_{i_l}].$$

Výsledné delenie roviny opäť tvoria (nie nutne neprázdne) konvexné množiny. □

Cvičenie 4.26 (50 bodov). *Nech* $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \subset \mathbb{E}^2$, $n \in \mathbb{N}$ *a nech* $W = \{w_1, \ldots, w_n\} \subset \mathbb{R}$. Mocnosťou bodu $\mathbf{x} \in \mathbb{E}^2$ *ku kružnici so stredom* $\mathbf{s} \in \mathbb{E}^2$ *a polomerom* $r \in \mathbb{R}_+$ *nazývame číslo* $p(\mathbf{x}, \mathbf{s}, r) = \|\mathbf{x} - \mathbf{s}\|^2 - r^2$. *Nájdite, čo je množinou* $V(\mathbf{p}_i) = \{\mathbf{x} \in \mathbb{E}^2 : p(\mathbf{x}, \mathbf{p}_i, w_i) \leq p(\mathbf{x}, \mathbf{p}_j, w_j), \quad \text{pre všetky} \quad j \neq i\}$. *Množiny* $\{V(\mathbf{p}_i), \mathbf{p}_i \in P\}$ *tvoria tzv.* mocnostný Voronoiov diagram. *Nakreslite ho pre niekoľko bodov.*

Riešenie (Matej Hudák, 25.11.2010):

Na začiatok určime pre danú n-prvkovú množinu P bodov roviny a pre každý bod $p_i \in P$ oblasť tých bodov roviny, ktoré sú bližšie k p_i ako k ľubovoľnému inému bodu z P. Označme hľadanú oblasť ako $V(p_i)$. Pre dané dva body p_i, p_j množinu bodov nie ďalej k p_i ako k p_j tvorí polrovina vyťatá osou spojnice $p_i p_j$, obsahujúca p_i. Označme túto polrovinu $H(p_i, p_j)$. Hľadaná oblasť $V(p_i)$ je prienikom $n-1$ polrovín a teda konvexnou oblasťou s najviac $n-1$ hranami.

$$V(p_i) = \bigcap_{\{j, j \neq i\}} H(p_i, p_j)$$

Oblasť $V(p_i)$ sa nazýva Voronoiovým mnohouholníkom združeným s bodom p_i (pozri obrázok 1 vľavo). Všetkých n takýchto oblastí $V(p_i)$, $i = 1 \ldots n$ rozkladá rovinu na konvexnú sieť označovanú ako Voronoiov diagram. Vzťah pre definovanie množiny všetkých bodov Voronoiovho mnohouholníka môžeme definovať aj pomocou nasledujúceho vzťahu:

$$V(p_i) = \{x \in E^2 : \, d(x, \, p_i) \leq d(x, \, p_j), \, \textit{pre všetky} \, i \neq j\}$$

Mocnostné Voronoiove diagramy (power Voronoi diagrams) sú jedným zo zovšeobecnení klasických Voronoiových diagramov (na obrázku 1 vpravo). Samotné zovšeobecnenie spočíva v zámene metriky d za vhodnú funkciu. V prípade MVD je tou funkciou mocnosť.

Mocnosť bodu x definovanú v zadaní si môžeme predstaviť geometricky. Z pohľadu ľubovoľného bodu $x \in E^2$ môžeme váhovaný bod p_i vidieť ako kružnicu so stredom p_i a polomerom $|w_i|$. Potom napríklad ak $p(x, \, p_i, \, w_i) > 0$, tak bod x leží vo vnútri kružnice so stredom p_i.

Ďalej pokračujeme podobne ako pri klasických VD. Množina bodov s menšou mocnosťou vzhľadom na p_i ako vzhľadom na p_j je opäť rovina ohraničená priamkou, pričom na hranici ležia body s rovnakou mocnosťou p vzhľadom na body p_i a p_j. Podobným postupom ako v riešení *príkladu* 4.27 sa dostaneme k rovnici hraničnej krivky v tvare $x = \frac{x_0}{2}$ pre $w_i = w_j$ a k hraničnej krivke v tvare

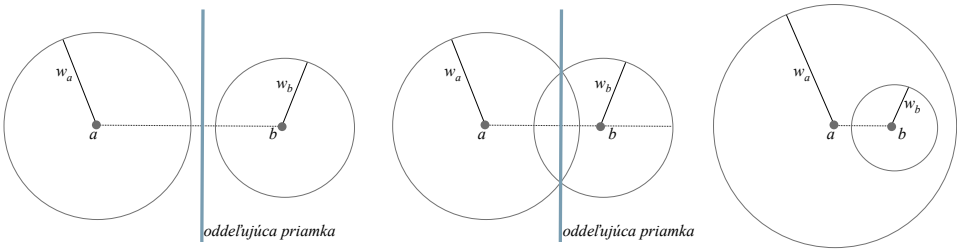
$$x = \frac{x_0^2+C}{2x_0},\ C = w_i^2 - w_j^2 \text{ pre } w_i \neq w_j,$$

čo ako vidíme je tiež priamka (x_0 je súradnica bodu), ale posunutá v závislosti na váhach bodov p_i a p_j.

Prienikom $n-1$ polrovín ohraničených priamkami dostaneme oblasť alebo región $V(p_i)$ v MVD, ktorá sa dá zapísať výrazom:

$$V(p_i) = \{x \in E^2 : p(x, p_i, w_i) \leq p(x, p_j, w_i), \textit{pre všetky } i \neq j\}$$

Aký vplyv majú váhy na vytváranie oddeľujúcej priamky? Nech máme v rovine zadané tri body a, b a ich váhy, zobrazené ako kružnice s polomerom w_a a w_b ako na obrázku 1. Tu môžme vidieť, že váhy bodov sú nenulové, pričom ak by váha daného bodu x bola nulová, polomer kružnice daný touto váhou by bol rovný nule. Z rovnice oddeľujúcej priamky (popísaná vyššie) vidíme, že aj keby bola váha určitého bodu záporná v rovnici by vystupovalo kladné číslo, keďže $(-1)^2 = 1$.



Obr. 4.79. Tri prípady tvorby oddeľujúcej priamky medzi dvoma bodmi

Na obrázku 1 vidíme viacero prípadov tvorenia oddeľovacej priamky medzi bodmi a a b. Tento proces závisí od súradníc bodov a ich váh. Nech $a = (0, 0)$ a $b = (x_0, 0)$.

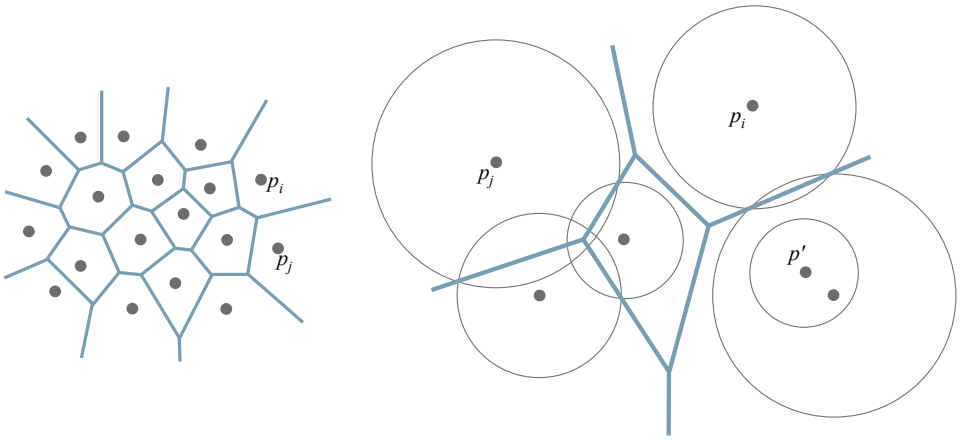
(1) V obrázku naľavo je $w_a > w_b$. Zároveň sú kružnice tvorené týmito váhami navzájom disjunktné. V tomto prípade sa oddeľujúca priamka vytvorí medzi týmito kružnicami podľa vyššie popísaného vzťahu. Vidíme, že oddeľujúca priamka je mierne posunutá k bodu b. Toto posunutie závisí od veľkosti rozdielu váh bodov, pričom bod s väčšou váhou „pohlcuje viac priestoru“ a oddeľujúca priamka sa vytvorí ako na obrázku, bližšie k bodu b.

(2) V obrázku v strede vidíme body, ktorých kružnice sa pretínajú. Platí rovnaký vzťah. V tomto prípade vidíme závislosť oddeľujúcej priamky od súradníc bodov a a b. Opäť je priamka posunutá k bodu b, keďže bod a má väčšiu váhu.

(3) V obrázku napravo sú dva body umiestnené tak, že bod b s kružnicou je „vo vnútri“ kružnice bodu a. V tomto prípade oddeľujúca priamka medzi bodmi a a b nevznikne. Môžeme to vidieť použitím hore uvedeného vzťahu pre konkrétne hodnoty. To znamená, že body a a b sú súčasťou jedného regiónu

MVD, pričom bod, ktorý má väčšiu váhu je generátorom tohto regiónu.

Ak sú všetky váhy $w_1 \dots w_n$ rovnaké VMD sa zhoduje s klasickým VD. V inom prípade je rozdiel medzi VMD a klasickým VD napríklad v tom, že pri VMD môže byť niektorá z jeho oblastí prázdna (na obrázku $2V(p')$).



Obr. 4.80. Klasický a mocnostný Voronoiov diagram

□

Cvičenie 4.27 (50 bodov). *Nech* $P = \{\mathbf{p}_1, \dots, \mathbf{p}_n\} \subset \mathbb{E}^2$, $n \in \mathbb{N}$ *a nech* $W = \{w_1, \dots, w_n\} \subset \mathbb{R}$. Aditívne váženou vzdialenosťou bodu $\mathbf{x} \in \mathbb{E}^2$ *ku bodu* $\mathbf{s} \in \mathbb{E}^2$ *váhou* $w \in \mathbb{R}$ *nazývame nazývame číslo* $p(\mathbf{x}, \mathbf{s}, w) = \|\mathbf{x} - \mathbf{s}\| - w$. *Nájdite, čo je množinou* $V(\mathbf{p}_i) = \{\mathbf{x} \in \mathbb{E}^2 : p(\mathbf{x}, \mathbf{p}_i, w_i) \leq p(\mathbf{x}, \mathbf{p}_j, w_j), \quad \text{pre všetky} \quad j \neq i\}$. *Množiny* $\{V(\mathbf{p}_i), \mathbf{p}_i \in P\}$ *tvoria tzv.* aditívne vážený Voronoiov diagram. *Nakreslite ho pre niekoľko bodov.*

Riešenie (Matej Hudák, 25.11.2010):

Použijeme definíciu klasického Voronoioveho diagramu z príkladu 4.25. Aditívne vážené Voronoiove diagramy(additive weighted Voronoi diagrams) sú ďalším zo zovšeobecnení klasických Voronoiových diagramov (na obrázku 1 vpravo). Samotné zovšeobecnenie spočíva v zámene metriky d za vhodnú funkciu. V prípade $AVVD$ je tou funkciou aditívne vážená vzdialenosť.

Množina bodov s menšou aditívnou vzdialenosťou vzhľadom na p_i ako vzhľadom na p_j je opäť rovina ohraničená krivkou, pričom na hranici ležia body s rovnakou aditívnou vzdialenosťou p vzhľadom na body p_i a p_j. Veľmi podobným postupom ako v riešení *príkladu* 4.27 sa dostaneme k rovnici hraničnej krivky v tvare $x = \frac{x_0}{2}$ pre $w_i = w_j$ a k hraničnej krivke v tvare:

$$0 = x^2(4x_0^2 - 2c^2) - x(4x_0^3) - y^2(2c^2) + x_0^4 \text{ pre } w_i \neq w_j,$$

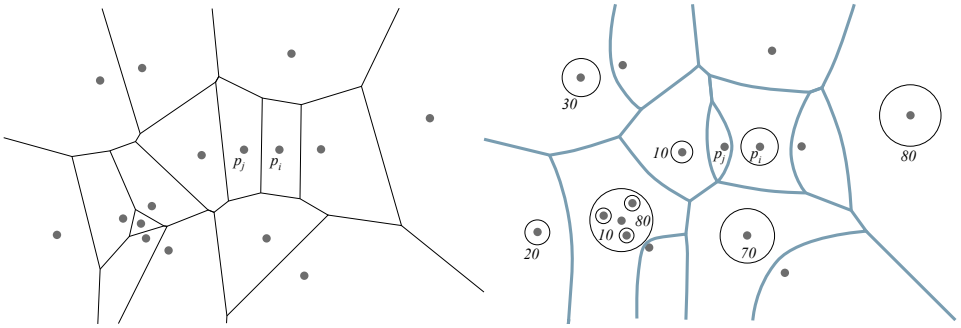
čo môžme prepísať do podoby:

$$0 = Ax^2 - Bx - Cy^2 - D$$

Prvý tvar hraničnej krivky nám napovedá že ide o hyperbolu, keďže y-ova časť rovnice je záporná (pred y^2 je mínus). Presnejšie je to časť hyperboly otvorená smerom k bodu s menšou váhou. Prienikom $n-1$ oblastí ohraničených takouto krivkou dostaneme oblasť $V(p_i)$ v $AVVD$, ktorá sa dá zapísať výrazom:

$V(p_i) = \{x \in E^2 : p(x, p_i, w_i) \leq p(x, p_j, w_i), \textit{pre všetky } i \neq j\}$

Ak sú všetky váhy $w_1 \dots w_n$ rovnaké $AVVD$ sa zhoduje s klasickým VD. V inom prípade rozdiel vidíme na obrázku 1 vpravo, kde modré krivky sú ohraničenia oblastí regiónov $AVVD$, kružnice znázorňujú váhu jednotlivých bodov v grafe, kde stredom danej kružnice x je generátor danej oblasti p $AVVD$. Polomerom je potom váha tohto generátora w_p. Čísla pri bodoch označujú veľkosť jednotlivých váh.



Obr. 4.81. Klasický a aditívne vážený Voronoiov diagram

□

Cvičenie 4.28 (50 bodov). *Nech* $P = \{\mathbf{p}_1, \dots, \mathbf{p}_n\} \subset \mathbb{E}^2$, $n \in \mathbb{N}$ *a nech* $W = \{w_1, \dots, w_n\} \subset \mathbb{R}_+$. Multiplikatívne váženou vzdialenosťou bodu $\mathbf{x} \in \mathbb{E}^2$ *ku bodu* $\mathbf{s} \in \mathbb{E}^2$ *váhou* $w \in \mathbb{R}_+$ *nazývame nazývame číslo* $p(\mathbf{x}, \mathbf{s}, w) = w\|\mathbf{x} - \mathbf{s}\|$. *Nájdite, čo je množinou* $V(\mathbf{p}_i) = \{\mathbf{x} \in \mathbb{E}^2 : p(\mathbf{x}, \mathbf{p}_i, w_i) \leq p(\mathbf{x}, \mathbf{p}_j, w_j), \quad \textit{pre všetky} \quad j \neq i\}$. *Množiny* $\{V(\mathbf{p}_i), \mathbf{p}_i \in P\}$ *tvoria tzv.* multiplikatívne vážený Voronoiov diagram. *Nakreslite ho pre niekoľko bodov.*

Riešenie (Viktória Bakurová, 2.1.2010): Chceme nájsť množinu $V(p_i)$, najprv teda nájdeme jej hranicu. Hranicou $V(p_i)$ bude množina všetkých takých bodov $x \in \mathbb{E}^2$, pre ktoré platí: $p(x, p_i, w_i) = p(x, p_j, w_j)$ pre všetky $j \neq i$. Presnejšie, pre každú dvojicu p_i a p_j, $j \neq i$ nájdeme krivku takú, že $p(x, p_i, w_i) = p(x, p_j, w_j)$ a hľadaná množina $V(p_i)$ bude prienikom týchto oblastí.

Nájdime hranicu pre dvojicu p_i a p_j, $j \neq i$, pričom $p_i = (x_i, y_i)$ a $p_j = (x_j, y_j)$. Urobme takú transformáciu roviny, aby $p_i = (0,0)$ a $p_j = (x_0, 0)$. Potom počítajme:

$p(x, p_i, w_i) = p(x, p_j, w_j)$
$w_i \|x - p_i\| = w_j \|x - p_j\|$
$w_i\sqrt{x^2+y^2} = w_j\sqrt{(x+x_0)^2+y^2}$

Môžu nastať 2 prípady: 1) $w_i = w_j$
2) $w_i \neq w_j$

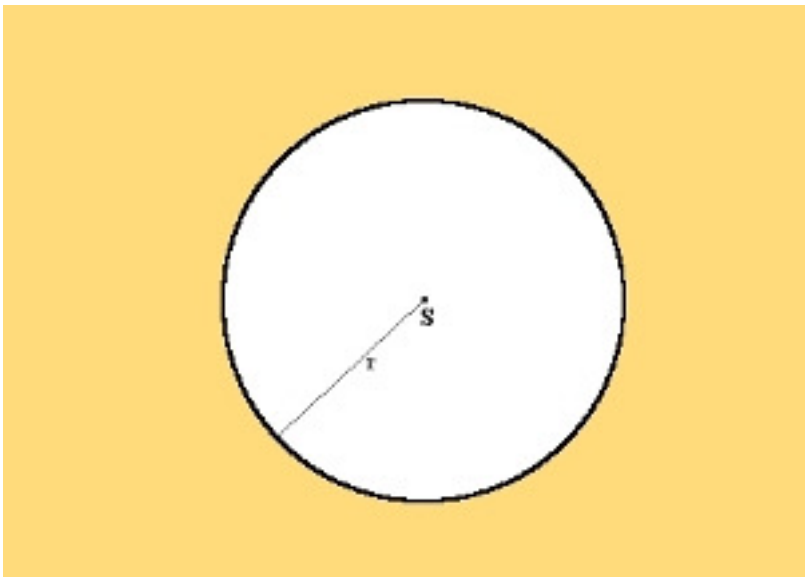
Najprv vyriešime prvý prípad, teda ak $w_i = w_j$.
$\sqrt{x^2+y^2} = \sqrt{(x-x_0)^2+y^2}$
$x^2+y^2 = (x-x_0)^2+y^2$
$x^2 = (x-x_0)^2$
$x^2 = x^2 - 2x_0x + x_0^2$
$0 = -2x_0x + x_0^2$
$x = \frac{x_0}{2}$
Z toho je jasné, že hranicou oblasti je priamka a výsledná oblasť je daná nerovnosťou $x \geq \frac{x_0}{2}$, ak $p(x, p_i, w_i) \geq p(x, p_j, w_j)$.

Teraz vyriešime prípad, ak $w_i \neq w_j$.
$w_i\sqrt{x^2+y^2} = w_j\sqrt{(x+x_0)^2+y^2}$
$w_i^2(x^2+y^2) = w_j^2((x+x_0)^2+y^2)$
$(x^2+y^2) = (\frac{w_j}{w_i})^2((x+x_0)^2+y^2)$
Označme $\frac{w_j}{w_i} = c$.
$(x^2+y^2) = c^2((x+x_0)^2+y^2)$
$x^2+y^2 = c^2(x+x_0)^2 + c^2y^2$
$x^2 - c^2(x+x_0)^2 + y^2 - c^2y^2 = 0$
$x^2 - c^2(x^2 - 2x_0x + x_0^2) + y^2 - c^2y^2 = 0$
$(1-c^2)x^2 + 2c^2x_0x - c^2x_0^2 + (1-c^2)y^2 = 0$

Vidíme, že sme dostali rovnicu tvaru $Ax^2 + Bx + Cy^2 + D = 0$. Táto rovnica je rovnicou elipsy. Nájdime jej stredový tvar:

$(1-c^2)\left(x^2 + \frac{2c^2x_0}{(1-c^2)}x\right) + (1-c^2)y^2 - c^2x_0^2 = 0$
$(1-c^2)\left(\left(x + \frac{c^2x_0}{(1-c^2)}\right)^2 - \left(\frac{c^2x_0}{(1-c^2)}\right)^2\right) + (1-c^2)y^2 - c^2x_0^2 = 0$
$(1-c^2)\left(\left(x + \frac{c^2x_0}{(1-c^2)}\right)^2 - \left(\frac{c^2x_0}{(1-c^2)}\right)^2\right) + (1-c^2)y^2 = c^2x_0^2$
$(1-c^2)\left(x + \frac{c^2x_0}{(1-c^2)}\right)^2 - \frac{(c^2x_0)^2}{(1-c^2)} + (1-c^2)y^2 = c^2x_0^2$

Označme výraz $\frac{c^2x_0}{(1-c^2)} = s_0$, potom:
$(1-c^2)(x+s_0)^2 - \frac{(c^2x_0)^2}{(1-c^2)} + (1-c^2)y^2 = c^2x_0^2$
$(1-c^2)(x+s_0)^2 + (1-c^2)y^2 = c^2x_0^2 + \frac{(c^2x_0)^2}{(1-c^2)}$



OBR. 4.82. Príslušná Voronoiova oblasť (vyfarbená) s hranicou kružnice

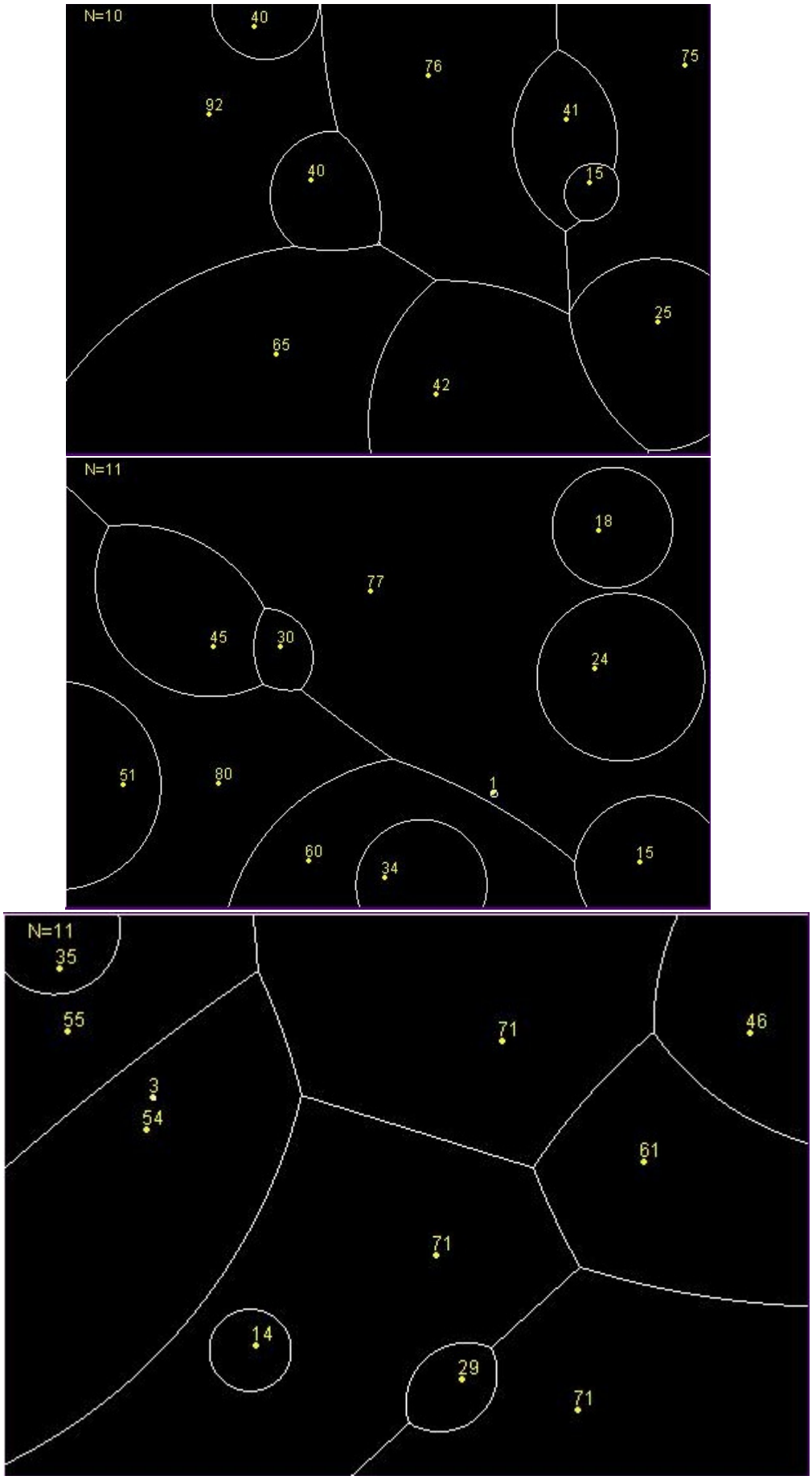
$$(1-c^2)\left(x+s_0\right)^2+(1-c^2)y^2=\frac{c^2x_0^2}{(1-c^2)}$$
$$\frac{(1-c^2)^2}{c^2x_0^2}\left(x+s_0\right)^2+\frac{(1-c^2)^2}{c^2x_0^2}y^2=1$$
$$\left(x+s_0\right)^2+y^2=\frac{(cx_0)^2}{(1-c^2)^2}$$

Výsledkom je teda kružnica so stredom v bode $S=\left[\frac{-c^2x_0}{1-c^2},0\right]$ a s polomerom $r=\frac{cx_0}{1-c^2}$, kde $c=\frac{w_j}{w_i}$.

V prípade, že c ide v limite k 1, dostávame špeciálny prípad, ktorý sme rozobrali v kroku 1).

Príslušná oblasť má rovnicu: $\left(x+\frac{c^2x_0}{1-c^2}\right)^2+y^2\geq\frac{(cx_0)^2}{(1-c^2)^2}$

Prienikom takýchto oblastí pre každé $j\neq i$ dostaneme príslušnú oblasť $V(p_i)$.



Obr. 4.83. Príklady multiplikatívne váženého Voronoiovho diagramu, pri bodoch sú uvedené ich váhy

□

Cvičenie 4.29 (25 bodov). *Klasifikujte prienik dvoch rôznych rotačných kužeľov s rovnobežnými osami rotácie a rovnakými vrcholovými uhlami.*

Cvičenie 4.30 (50 bodov). *Analyzujte, čo je množina bodov v* $\mathbb{E}^2$*, ktoré spĺňajú rovnicu*

$$\|\mathbf{x}-\mathbf{p}\|+a_{\mathbf{p}}=\|\mathbf{x}-\mathbf{q}\|+a_{\mathbf{q}},$$

kde $\mathbf{p},\mathbf{q}\in\mathbb{E}^2$ *sú ľubovoľné, pevne zvolené,* $a_{\mathbf{p}},a_{\mathbf{q}}\in\mathbb{R}$*.*

Riešenie (Michal Ferko, 30.10.2011):V prípade, že $\mathbf{p}=\mathbf{q}$, môžeme výraz upraviť takto:

$$\begin{aligned}||\mathbf{x}-\mathbf{p}||+a_{\mathbf{p}}&=||\mathbf{x}-\mathbf{q}||+a_{\mathbf{q}}\\ ||\mathbf{x}-\mathbf{p}||+a_{\mathbf{p}}&=||\mathbf{x}-\mathbf{p}||+a_{\mathbf{q}}\\ ||\mathbf{x}-\mathbf{p}||-||\mathbf{x}-\mathbf{p}||+a_{\mathbf{p}}&=a_{\mathbf{q}}\\ a_{\mathbf{p}}&=a_{\mathbf{q}}\end{aligned}$$

V tomto prípade ak $a_{\mathbf{p}}=a_{\mathbf{q}}$, tak je rovnica splnená pre všetky $\mathbf{x}\in\mathbb{E}^2$, teda riešením je celá rovina. Ak $a_{\mathbf{p}}\neq a_{\mathbf{q}}$, tak riešením je prázdna množina, lebo rovnica nebude nikdy splnená.

Ďalej budeme predpokladať, že $\mathbf{p}\neq\mathbf{q}$. Najprv si skúsime upraviť zjednodušený výraz, kde body $\mathbf{p}$ a $\mathbf{q}$ majú pevne dané súradnice. Do takejto situácie sa vieme dostať jednoduchou transformáciou súradníc. Nech teda $||\mathbf{p}-\mathbf{q}||=2d$ a po transformácii súradníc $\mathbf{x}=[x,y]$, $\mathbf{p}=[-d,0]$ a $\mathbf{q}=[d,0]$, teda transformáciou sa zachová vzdialenosť medzi bodmi $\mathbf{p}$ a $\mathbf{q}$.

Potom:

$$
\begin{aligned}
||\mathbf{x}-\mathbf{p}|| + a_{\mathbf{p}} &= ||\mathbf{x}-\mathbf{q}|| + a_{\mathbf{q}} && \\
\sqrt{(x+d)^2+y^2} + a_{\mathbf{p}} &= \sqrt{(x-d)^2+y^2} + a_{\mathbf{q}} && / - a_{\mathbf{p}} \\
\sqrt{(x+d)^2+y^2} &= \sqrt{(x-d)^2+y^2} + (a_{\mathbf{q}} - a_{\mathbf{p}}) && /^2, r = (a_{\mathbf{q}} - a_{\mathbf{p}}) \\
(x+d)^2+y^2 &= (x-d)^2+y^2+2r\sqrt{(x-d)^2+y^2}+r^2 && / - y^2 \\
x^2+2dx+d^2 &= x^2-2dx+d^2+2r\sqrt{(x-d)^2+y^2}+r^2 && / - x^2+2dx-d^2-r^2 \\
4dx-r^2 &= 2r\sqrt{(x-d)^2+y^2} && /^2 \\
16d^2x^2-8dr^2x+r^4 &= 4r^2((x-d)^2+y^2) && / : 4r^2 \\
\frac{4d^2}{r^2}x^2-2dx+\frac{r^2}{4} &= x^2-2dx+d^2+y^2 && / - x^2+2dx-\frac{r^2}{4}-y^2 \\
\frac{4d^2-r^2}{r^2}x^2-y^2 &= \frac{4d^2-r^2}{4} && / * \frac{4}{4d^2-r^2} \\
\frac{4}{r^2}x^2-\frac{4}{4d^2-r^2}y^2 &= 1 && \\
\frac{x^2}{(\frac{r}{2})^2}-\frac{y^2}{\sqrt{\frac{4d^2-r^2}{4}}^2} &= 1 && /a=\frac{r}{2}, b=\sqrt{\frac{4d^2-r^2}{4}} \\
\frac{x^2}{a^2}-\frac{y^2}{b^2} &= 1 &&
\end{aligned}
$$

Dopracovali sme sa ku rovnici hyperboly. Počas našich úprav sme ale spravili zopár neekvivalentných operácii, a teda musíme dať na naše konštanty d a r nejaké obmedzenia. Delili sme $4r^2$, preto $r \neq 0$. Ak by $r = 0$ v riadku, kde sme delili $4r^2$, dostali by sme:

$$
\begin{aligned}
16d^2x^2 &= 0 \\
x^2 &= 0 \\
x &= 0
\end{aligned}
$$

Čiže výsledkom by bola priamka $x = 0$, čo je os úsečky $\overline{\mathbf{pq}}$. Toto je prípad 1.

Ďalšiu neekvivalentnú úpravu sme robili, keď sme násobili zlomkom $\frac{4}{4d^2-r^2}$. Z toho dostávame:

$$
\begin{aligned}
4d^2-r^2 &\neq 0 \\
(2d)^2-r^2 &\neq 0 \\
(2d)^2 &\neq r^2 \\
2d &\neq r
\end{aligned}
$$

Teda ak $r = 2d = ||\mathbf{p}-\mathbf{q}||$, tak nastávajú prípady 2(b) alebo 3(b), lebo dostaneme:

$$\begin{aligned}\frac{4d^2-r^2}{r^2}x^2 - y^2 &= \frac{4d^2-r^2}{4}\\ \frac{0}{r^2}x^2 - y^2 &= \frac{0}{4}\\ -y^2 &= 0\\ y &= 0\end{aligned}$$

Posledná neekvivalentná úprava bola vloženie výrazu $\frac{4d^2-r^2}{4}$ pod odmocninu. Musí platiť:

$$\begin{aligned}\frac{4d^2-r^2}{4} &> 0\\ 4d^2-r^2 &> 0\\ (2d)^2 &> r^2\\ 2d &> r\end{aligned}$$

Dostávame sa ku prípadom 2(c) a 3(c), keď $r > 2d = ||\mathbf{p}-\mathbf{q}||$. Vtedy rovnica naozaj nie je splnená, lebo neplatí podmienka, ktorú sme si odvodili.

Celkovo teda nastávajú už spomínané prípady:

(1) Ak $a_{\mathbf{q}} = a_{\mathbf{p}}$, tak dostávame:

$$||\mathbf{x}-\mathbf{p}|| = ||\mathbf{x}-\mathbf{q}||$$

Teda hľadáme tie body $\mathbf{x}$, pre ktoré vzdialenosť od bodov $\mathbf{p}$ a $\mathbf{q}$ je rovnaká. V rovine sú tieto body práve osou úsečky $\overline{\mathbf{pq}}$.

(2) Ak $a_{\mathbf{q}} > a_{\mathbf{p}}$, tak naša rovnica sa veľmi podobá na alternatívnu definíciu hyperboly, ktorá hovorí: Bod $\mathbf{x}$ patrí do hyperboly s ohniskami $\mathbf{p}$ a $\mathbf{q}$, ak absolútna hodnota rozdielu vzdialeností je konštanta. Toto by sedelo na našu rovnicu až na tú absolútnu hodnotu. Problém je, že konštanty $a_{\mathbf{p}}$ a $a_{\mathbf{q}}$ nemôžu byť ľubovoľné, aby to bola hyperbola. Nastávajú tieto prípady:
   (a) Ak $0 < a_{\mathbf{q}} - a_{\mathbf{p}} < ||\mathbf{p}-\mathbf{q}||$, tak výsledkom je naozaj hyperbola (teda iba komponent súvislosti hyperboly bližšie k bodu $\mathbf{q}$).
   (b) Ak $a_{\mathbf{q}} - a_{\mathbf{p}} = ||\mathbf{p}-\mathbf{q}||$, tak rozdiel vzdialeností od bodov $\mathbf{p}$ a $\mathbf{q}$ má byť rovný ich vzdialenosti. Toto je ale maximálny rozdiel vzdialeností, aké môžu body v rovine od týchto bodov nadobudnúť (kvôli trojuholníkovej nerovnosti): jediné takéto body $\mathbf{x}$, pre ktoré platí rovnosť sú tie body, ktoré ležia na polpriamke určenej bodom $\mathbf{q}$ a vektorom $\mathbf{q}-\mathbf{p}$ (pre tieto body triviálne rozdiel vzdialeností od $\mathbf{p}$ a $\mathbf{q}$ je práve $||\mathbf{p}-\mathbf{q}||$).

(c) Ak $a_{\mathbf{q}} - a_{\mathbf{p}} > ||\mathbf{p}-\mathbf{q}||$, tak výsledkom je prázdna množina, kvôli trojuholníkovej nerovnosti: Ak $||\mathbf{x}-\mathbf{p}|| = a$ a $||\mathbf{q}-\mathbf{p}|| = b$, tak potom $||\mathbf{x}-\mathbf{q}|| \leq a+b$.

(3) Ak $a_{\mathbf{q}} < a_{\mathbf{p}}$, tak výsledné množiny sú rovnaké ako v predchádzajúcom prípade, akurát sú body $\mathbf{p}$ a $\mathbf{q}$ vymenené. Teda:

(a) Ak $a_{\mathbf{p}} - a_{\mathbf{q}} < ||\mathbf{p}-\mathbf{q}||$, tak výsledkom je naozaj hyperbola (teda iba komponent súvislosti hyperboly bližšie k bodu $\mathbf{p}$).

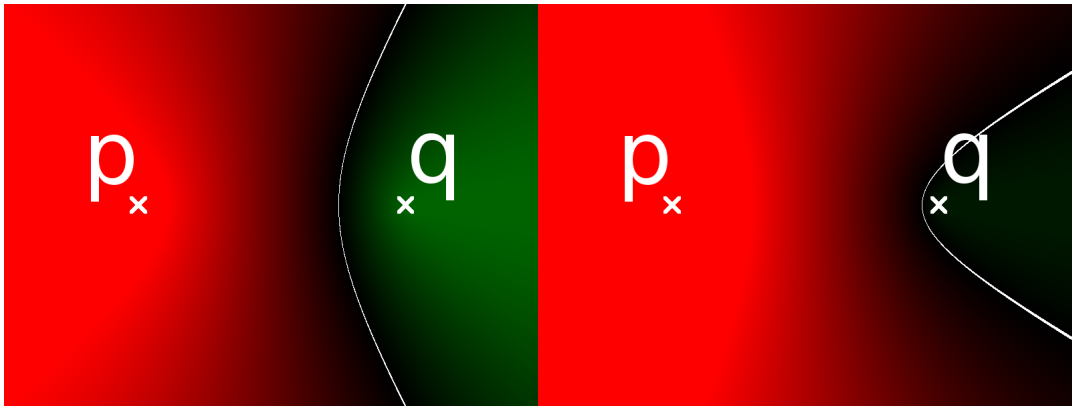
(b) Ak $a_{\mathbf{p}} - a_{\mathbf{q}} = ||\mathbf{p}-\mathbf{q}||$, tak body $\mathbf{x}$, pre ktoré platí rovnosť sú tie body, ktoré ležia na polpriamke určenej bodom $\mathbf{p}$ a vektorom $\mathbf{p}-\mathbf{q}$ (pre tieto body triviálne rozdiel vzdialeností od $\mathbf{p}$ a $\mathbf{q}$ je práve $||\mathbf{p}-\mathbf{q}||$).

(c) Ak $a_{\mathbf{p}} - a_{\mathbf{q}} > ||\mathbf{p}-\mathbf{q}||$, tak výsledkom je prázdna množina, kvôli trojuholníkovej nerovnosti: Ak $||\mathbf{x}-\mathbf{p}|| = a$ a $||\mathbf{q}-\mathbf{p}|| = b$, tak potom $||\mathbf{x}-\mathbf{q}|| \leq a+b$.

Pre lepšie objasnenie sú zobrazené vizualizácie na obrázku 4.84. V týchto obrázkoch sú vyznačené body $\mathbf{p}$ a $\mathbf{q}$ a majú súradnice (v pixeloch) $\mathbf{p} = [200, 300]$ a $\mathbf{q} = [600, 300]$, teda $||\mathbf{p}-\mathbf{q}|| = 400$. V obrázkoch sa farebne vizualizuje hodnota výrazu:

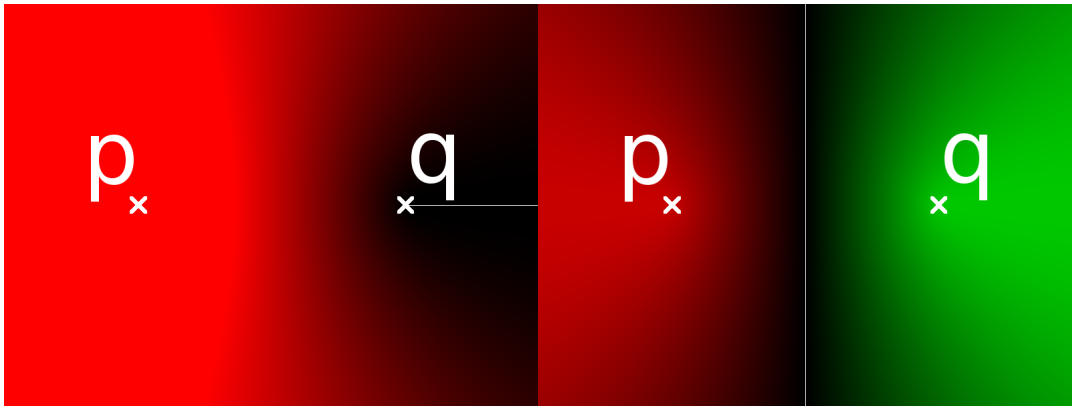
$$f : \mathbb{E}^2 \to \mathbb{R}$$
$$f(\mathbf{x}) = ||\mathbf{x}-\mathbf{p}|| - ||\mathbf{x}-\mathbf{q}|| - r$$

Kde $r := a_{\mathbf{q}} - a_{\mathbf{p}}$ je konštanta, ktorej hodnota je v každom obrázku iná (a teda určuje výsledné body našej rovnice). Červené pixely sú záporné hodnoty ($f(\mathbf{x}) < 0$), zelené sú kladné hodnoty ($f(\mathbf{x}) > 0$) a biele sú blízko k nule ($|f(\mathbf{x})| < \varepsilon$) - toto sú body spĺňajúce našu rovnicu. Intenzita červených a zelených bodov určuje ich hodnotu (zelenšia $\Rightarrow$ vyššie hodnoty, červenšia $\Rightarrow$ nižšie hodnoty).



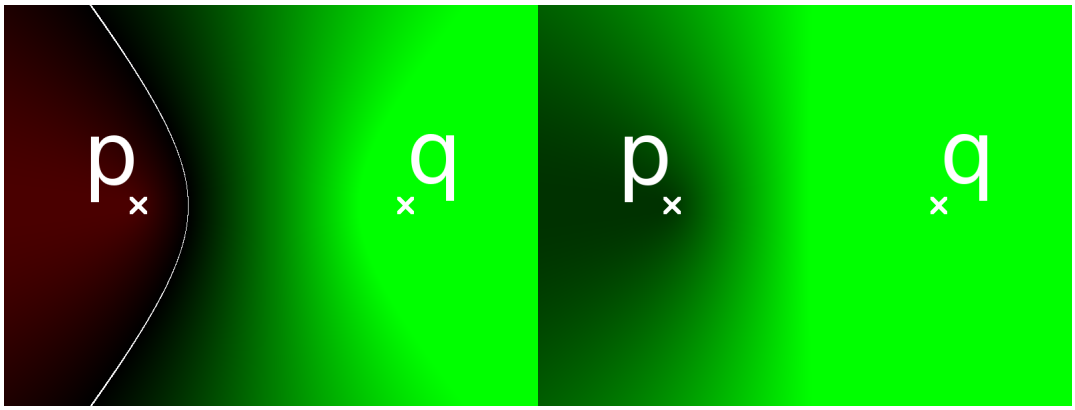
(A) r = 200, prípad 2(b)

(B) r = 350, prípad 2(b)



(C) r = 400, prípad 2(a)

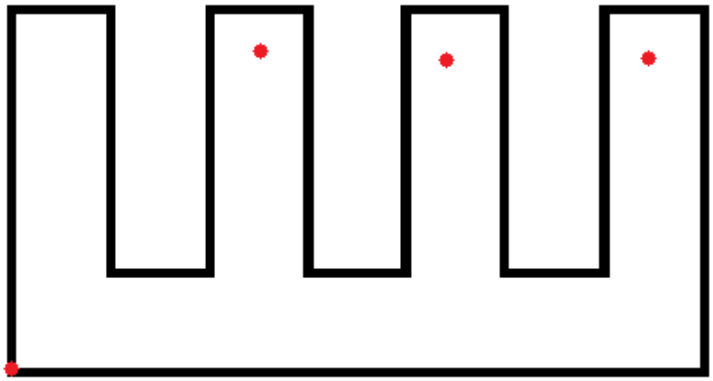
(D) r = 0, prípad 1



(E) r = -250, prípad 3(b)

(F) r = -500, prípad 3(c)

OBR. 4.84. Farebná vizualizácia našej rovnice - Vyobrazené všetky prípady, ktoré môžu nastať



Obr. 4.85. Príklad dosahujúci hornú hranicu

□

Cvičenie 4.31. *Ortogonálny mnohouholník je taký, ktorého strany sú buď zvislé alebo horizontálne. Nech P je ortogonálny mnohouholník s n vrcholmi. Dokážte, že $\lfloor \frac{x}{4} \rfloor$ kamier je niekedy nutný, ale vždy postačujúci počet na jeho stráženie.*

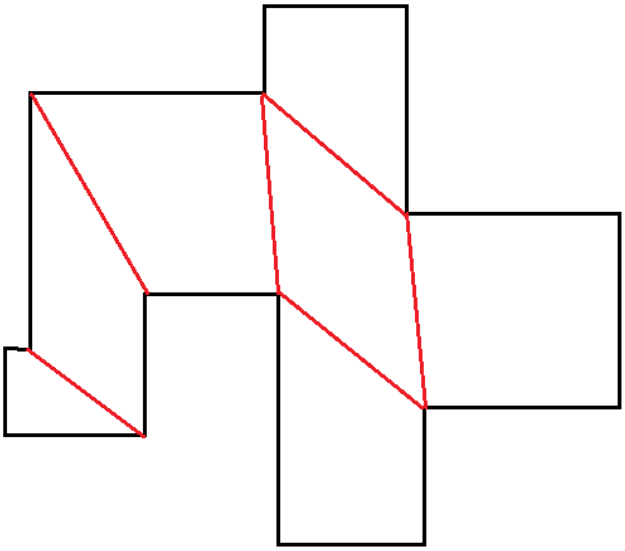
Riešenie (Kiss Gábor, 28.11.2010): Najprv si ukážme prípad, keď je počet strážcov(kamier) práve $\lfloor \frac{x}{4} \rfloor$ (poz. obr. 4.85)

Dôkaz toho, že na stráženie je postačujúci počet strážcov $\lfloor \frac{x}{4} \rfloor$ budeme konštruovať podobne ako pri dôkaze všeobecného problému stráženia galérie. Rozložme si daný mnohouholník P na konvexné štvoruholníky. Budeme postupovať postupným odstraňovaním štoruholníkov nasledovne: Vyberieme si taký štvoruholník, že tri z jeho strán sú hranami mnohouholníka a štvrtá je jeho diagonála. Následne odstránime tento štvoruholník. Na novovzniknutý mnohouholník použijeme predchádzajúci postup. Algoritmus končí vtedy, keď z mnohouholníka ostane už len štvoruholník. Týmto postupom sme určili také diagonály, ktoré nám pôvodný mnohouholník rozdelili na konvexné štvoruholníky-označme si túto množinu štvoruholníkov ako $Kvad(P)$ (poz. obr. 4.86).

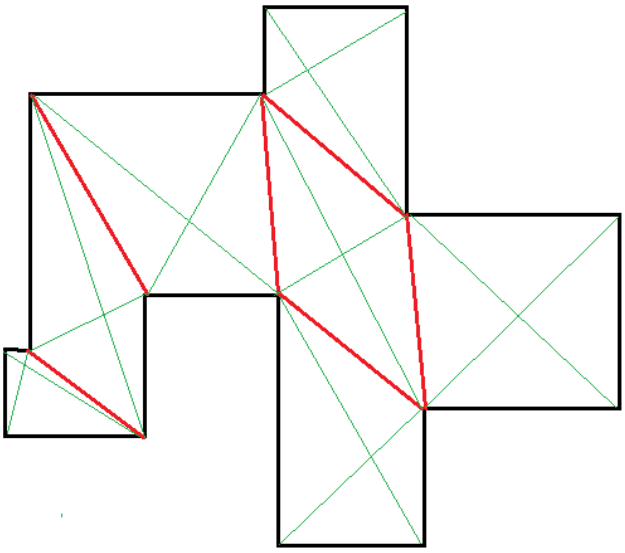
Teraz si zostavme graf G tak, že každému štvoruholníku pridáme obe jeho diagonály. Týmto nám vznikol planárny graf, ktorý je štvorzafarbiteľný (poz. obr. 4.87).

Dôkaz štvorzafarbiteľnosti mnohouholníka P s pridanými uhlopriečkami v tomto prípade sa dá urobiť aj nasledovne.

Vytvorme duálny graf Q ku $Kvad(P)$ tak, že každému štroruholníku z $Kvad(P)$ bude priradený práve jeden vrchol grafu Q. Dva vrcholy budú prepojené hranou jedine vtedy, ak štvoruholníky, ktorým sú priradené, majú spoločnú hranu. Tento graf je stromom (za predpokladu, že P neobsahuje diery). Následne budeme postupovať indukciou.

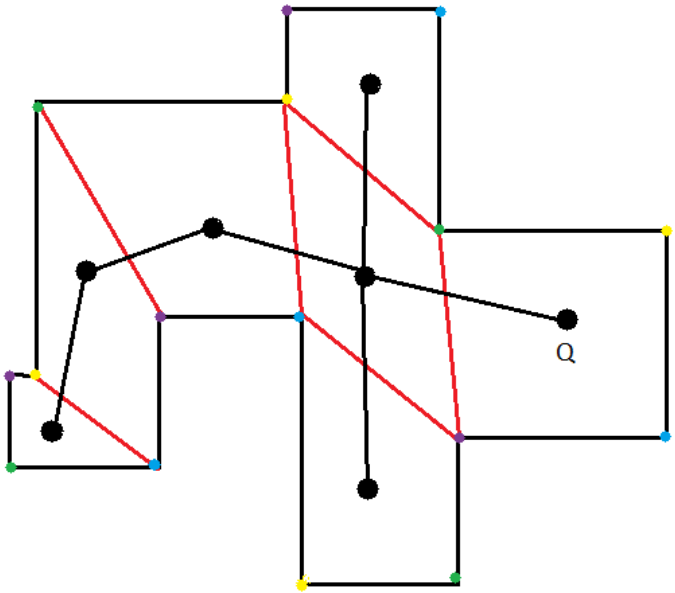


Obr. 4.86. Rozdelenie ortogonálneho mnohouholníka na konvexné štvorsteny



Obr. 4.87. Rovinný (a teda štvorzafarbiteľný) graf

Predpokladajme, že je štvorzafarbiteľný graf s m vrcholmi. Vezmime si teraz $n > m$. Vyberme si z Q list stromu. Ten je určite spojený s ostanými časťami grafu páve jednou hranou, ktorá zodpovedá diagonále v mnohouholníku. Odstráňme teda tento vrchol z grafu a aj



Obr. 4.88. Umiestnenie strážcov do vrchol ofarbeného grafu

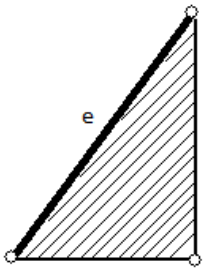
štvoruholník z $Kvad(P)$, ktorému prislúchal. Vznikne nám mnohouholník s menším počtom vrcholov ako mal P – označme ho P'. Pre všetky mnohouholníky s menším počtom vrcholov ako n máme už dokázané, že sú štvorzafarbitelné. Ostáva nám zafarbiť už len vrcholy z P, ktoré sme odstránili. Vzhľadom na to, že odstránený štvoruholník má dva spoločné vrcholy s P', tak musíme zafarbiť už len dva vrcholy. Týmto priradíme farby rôzne od spoločných vrcholov štvoruholníka a P'.

Strážcov umiestnime do tých vrcholov, ktoré sú označené najmenej používanou farbou.

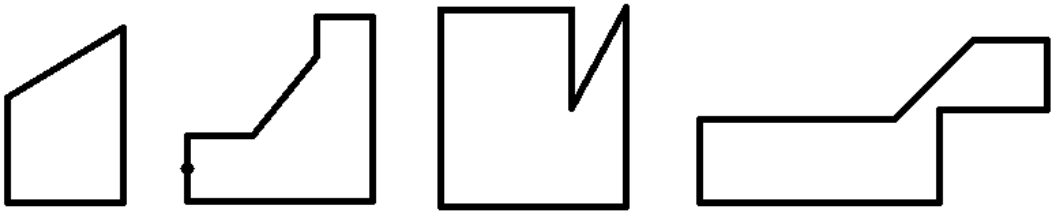
Ešte musíme zodpovedať jednu otázku. Existuje vždy rozdelenie takéhoto mnohouholníka na konvexné štvoruholníky?

Dôkaz, ktorý si naznačíme je z knihy *Art Gallery theorems and algorithms od Jaseph O'Rourke* . Základnou myšlienkou dôkazu je jednoduché pozorovanie- po oddelení štvoruholníka z ortogonálneho mnohouholníka nám zostane zvyšok, ktorý nie je nutne ortogonálny- ale napriek tomu je naďalej naštvoruholníkovatelný. Teda by mala existovať širšia trieda mnohouholníkov na ktorej by sa dalo dokázať, že existuje rozdelenie na konvexné štvoruholníky. Túto triedu mnohouholníkov ako prvá definovala Lubiw, ktorej dôkaz tu načrtneme.

Nech P je náš ortogonálny mnohouholník. Duálny graf k danému $Kvad(P)$ je strom. Odobratím jedného štvoruholníka z $Kvad(P)$ sa nám teda musí P rozdeliť na časti, ktoré sú naďalej ortogonálne až na jednu hranu. Toto pozorovanie nám pomôže definovať 1-ortogonálne



Obr. 4.89. Nos naklonenej hrany



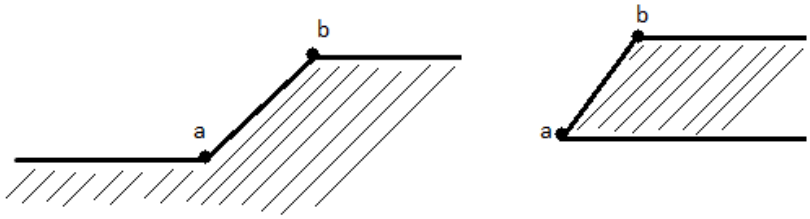
Obr. 4.90. Kontrapíklady porušenia podmienok 1-ortogonálnosti

mnohouholníky. Sú to také mnohouholníky, ktoré neobsahujú žiadnu dieru a majú najviac jednu hranu – e, ktorá nie je nutne zvislá, ani horizontálna (nazývame ju naklonená hrana) a ďalej spĺňa nasledujúce podmienky:

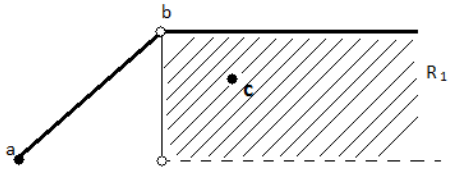
(1) Má párny počet hrán
(2) Okrem hrany e sú všetky hrany striedavo horizontálne, alebo vertikálne.
(3) Všetky vnútorné uhly sú menšie, alebo rovné 270°
(4) Nos naklonenej hrany neobsahuje žiadne vrcholy

Nosom naklonenej hrany e nazývame pravouhlý trojuholník smerujúci do vnútra mnohouholníka, ktorého preponou je hrana e- nos zahráňa vnútro e, ale nie jeho hraničné body (poz. obr. 4.89).

Každý ortogonálny mnohouholník je aj 1-ortogonálny mnohouholník, kde e môže byť ľubovoľná hrana. Porušenie ktorejkoľvek podmienky z 1 až 4 môže viesť k tomu, že mnohouholník nebude štvoruholníkovatelný. Na obrázku (Obr. 6) vidieť mnohouholníky, ktoré nespĺňajú dané pravidlá.



Obr. 4.91. Dve situácie pre susedné hrany k hrane ab



Obr. 4.92. Možnosti umiestnenia vrchola c

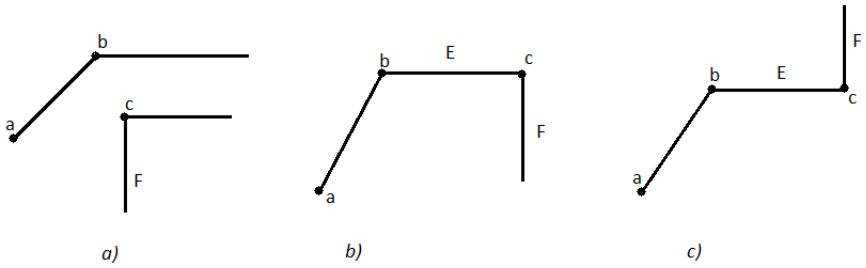
Tvrdenie: Každý 1-ortogonálny mnohouholník je možné rozdeliť na konvexné štvoruholníky.

Dôkaz:

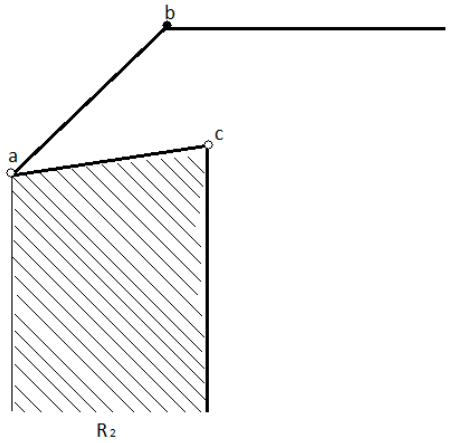
Dôkaz budeme robiť indukciou. Ak má P len 4 hrany, tak má dve horizontálne a jednu vertikálnu hranu (možno bude treba vykonať rotáciu o 90°, aby to platilo). Bez ohľadu na orientáciu e musí byť P konvexné. Toto bude tvoriť základ našej indukcie. Ďalej predpokladajme teda, že P má viac ako 4 hrany. Ukážeme, že existuje konvexný štvoruholník, ktorého odobratím z P dostaneme opäť 1-ortogonálny mnohouholník s menším počtom hrán. Vlastnosti 1 a 2 nám zabezpečujú, že susedné hrany k e sú buď obe horizontálne, alebo vertikálne. Označme si koncové body e ako a a b. Teda $e = ab$. Z vlastnosti 3 vieme, že nám môžu nastať dve situácie (poz. obr. 4.91).

V oboch prípadoch je bod b koncovým bodom hornej hrany. Body a a b sú zároveň aj vrcholmi štvoruholníka, ktorý chceme odstrániť. Potrebujeme teda určiť ešte ďalšie dva vrcholy- označme si ich c a d. Vrchol c nech je najľavejší a najvyššie umiestnený vrchol z oblasti R_1 (poz. obr. 4.92).

R_1 je zľava a zhora uzavretá oblasť, neobsahuje ľavé vrcholy a inde je otvorená. Keďže R_1 obsahuje aspoň vrchol na druhom konci hrany



Obr. 4.93. Bod c ako vrch vertikálnej hrany

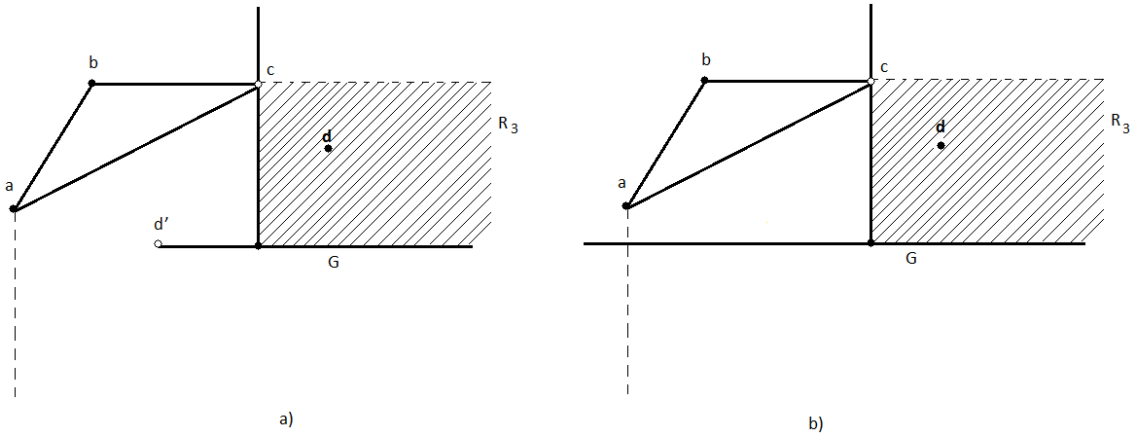


Obr. 4.94. Definícia oblasti R_2

susediacej k b, tak c musí existovať. V prípade, že c nie je týmto bodom, tak vzhľadom na to, že musí byť najľavejší bude na vrchu vertikálnej hrany (poz. obr. 4.93).

Vrchol d štvoruholníka je možné určiť analýzou dvoch prípadov. Definujme si región R_2 tak, ako vidíme na obr. 4.94.

Oblasť R_2 je uzavretá zhora a sprava, neobsahuje body a a c, inak je otvorená. Bod d nech je teda najvyššie umiestnený, a najpravejší vrchol v R_2. Ak nastane jedna zo situácií obr. 4.93a alebo obr. 4.93b, tak určite obsahuje aspoň spodný vrchol hrany F, ktorej patrí aj vrchol c. V prípade, keď $F \neq cd$, tak je d najvyšší – teda je na vrchole vertikálnej hrany, a zároveň je aj najviac v pravo – teda je na pravej strane horizontálnej hrany – teda je v rohu. V prípade zobrazenom na obr. 4.93c môže byť d ľavým koncom hrany G v R_2 (označený ako d' na obr. 4.95a), alebo R_2 nemusí obsahovať žiaden vrchol (poz. obr. 4.95b).



Obr. 4.95. Definícia oblasti R_3

V oboch týchto prípadoch si definujme oblasť R_3 tak, ako je znázornený na obr. 4.95. Teda je uzavretý zľava a zospodu, neobsahuje ľavé vrcholy, inak je otvorený. Teraz definujme d ako najľavejší, najnižšší vrchol nachádzajúci sa v R_3. Keďže R_3 musí obsahovať aspoň pravý vrchol hrany G, tak d musí existovať.

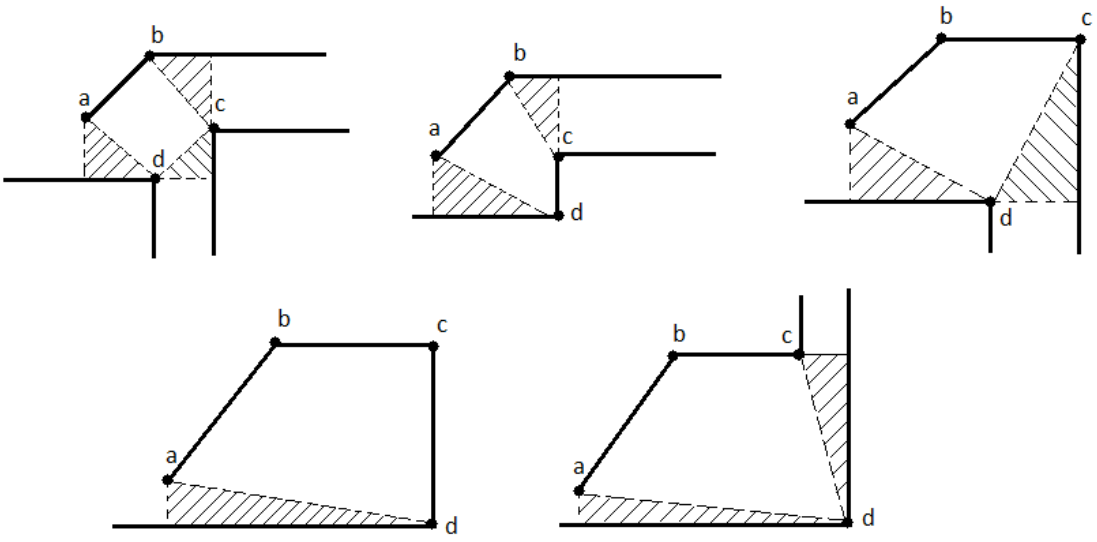
Teraz, keď máme definované vrcholy $abcd$ za všetkých okolností, odobrerme štvoruholník $abcd$. Tento štvoruholník musí byť konvexný, lebo uhol pri b je konvexný, keďže c je v R_1. Uhol pri a je konvexný, lebo d je v R_2, alebo R_3. Nakoniec uhly pri c a d sú konvexné, ak R_2 a R_3 sú platné. Odstránením $abcd$ nám môže zanechať 3 mnohouholníkové regióny a hraničené hranami bc, cd a ad. Každá z týchto diagonál je jedinou naklonenou hranou v prislúchajúcich mnohouholníkoch- tým sa zachovala vlastnosť 2 z definície 1-ortogonálnych mnohouholníkov. Tretiu podmienku si môžeme ľahko overiť nakreslením všetkých rozpísaných vyššie (poz. obr. 4.96).

Štvrtá podmienka je zaručená voľbou vrchola c, ako najľavejšieho a najvyššieho, a podobne je to aj pri vrchole d (obr. 4.96 – vyšrafované časti).

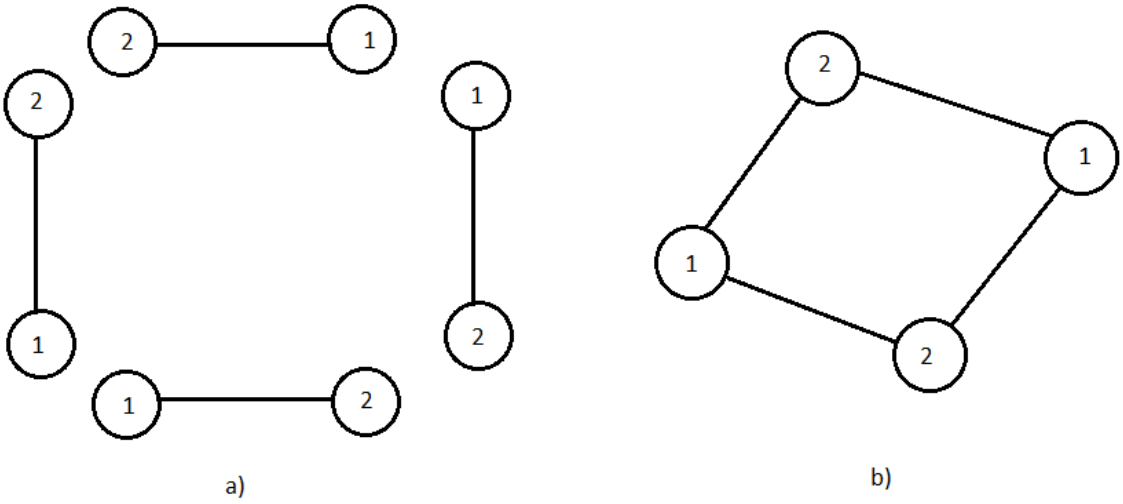
Napokon nám ostala prvá podmienka- teda, že počet vrcholov je párny.

Definujme si koncové body vrchných, spodných, ľavých a pravých hrán na dva typy, ako je ukázané na obr. 4.97a.

Koncovým bodom naklonených hrán je priradený typ podľa prislúchajúcej horizontálnej, alebo vertikálnej hrany. Je jasné, že každému vrcholu 1-ortogonálneho mnohouholníka je priradený typ jednoznačne. Napríklad pri konvexnom rohu, kde sa stretávajú vrchná a ľavá strana môže byť jedine vrchol typu 1. Pri prechádzaní hranice každého 1-ortogonálneho mnohouholníka dochádza k striedaniu typu vrcholov. Nech je vrchol a typu 1 a b typu 2 (obr. 4.91), prípady ukázané na



Obr. 4.96. Rozloženie 1-ortogonálneho mnohouholníka na maximálne tri menšie



Obr. 4.97. Párnosť počtu strán nových mnohouholníkov

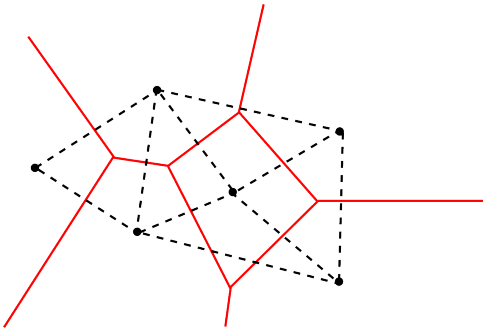
obr. 4.96 ukazujú, že c je vždy typu 1 a d typu 2. Teda každá diagonála bc, cd a ad ma koncové body oboch typov (obr. 4.97b) a teda k nim prislúchajúce mnohouholníky dodržujú striedanie vrcholov typu 1 a 2. Teda musia mať párny počet vrcholov. Týmto sme ukázali, že každý mnohouholník, ktorý ostane po oddelení štvoruholníka $abcd$ je 1- ortogonálny mnohouholník. Je jasné, že majú menej vrcholov, ako P. Pomocou indukdcie je teda jasné, že P je rozdelitelný na konvexné štvoruholníky. □

Cvičenie 4.32 (10 bodov). *Zistite, či sa Voronoiov diagram dá pre ľubovoľnú neprázdnu množinu generátorov opísať rovinným grafom. Ak sa pre nejaké množiny nedá, charakterizujte ich.*

Cvičenie 4.33 (15 bodov). *Zdôvodnite podrobne, prečo generátor ohraničeného $V(\mathbf{p}_i)$ neleží na $\partial\,\mathrm{conv}(P)$.*

Cvičenie 4.34 (10 bodov). *Ukáž, že Voronoiov diagram n prvkovej množiny P má najviac $2n-5$ vrcholov a najviac $3n-6$ hrán.*

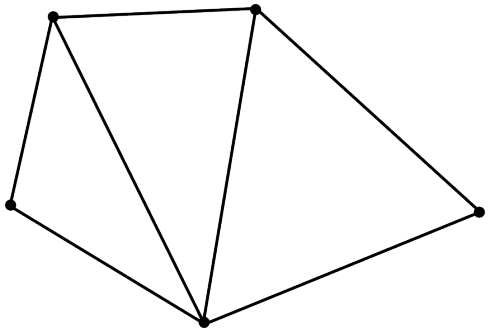
Riešenie (Jakub Mišún, 30.11.2011): Vieme, že ak vytvoríme duálnu štruktúru k Voronoiovmu diagramu, dostaneme triangluláciu množiny P. Vďaka dualizmu taktiež vieme, že počet hrán triangulácie sa rovná počtu hrán Voronoiovho diagramu a rovnako počet trojuholníkov triangulácie sa rovná počtu vrcholov Voronoiovho diagramu. To je spôsobené tým, že generátory (reprezentujúce oblasti) vymeníme s vrcholmi triangulácie (susedné generátory tvoria hranu rovnako ako susedné vrcholy triangulácie) a opačne vrcholy Voronoia vymeníme za trojuholníky. V dôsledku tohoto zistenia nám teda stačí nájsť maximálny počet trojuholníkov a hrán triangulácie.



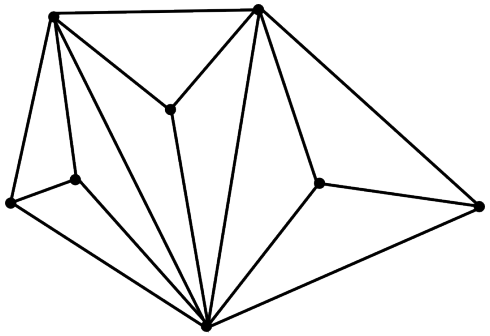
Obr. 4.98. Duálna štruktúra k Voronoiovmu diagramu je triangukácia

Najprv ukážeme počet trojuholníkov. Máme n bodov, pre ktoré vytvoríme triangulácie. Tieto vieme rozdeliť na body, ktoré tvoria konvexný obal (označme ich O), nech ich je o a tie čo sú v jeho vnútri (označme ich I), nech ich je i. Nutne teda $n = i+o$. Poďme si teraz zostrojiť triangluláciu tých to bodov. Najprv zoberieme iba body O. Tieto tvoria konvexný mnohouholník a ten vieme rozdeliť na $o-2$ trojuholníkov, čo je triviálne zistenie. Máme teda $o-2$ trojuholníkov. Pozrime sa teraz, čo sa stane ak pridáme jeden z bodov I. Je zjavné, že jeden takýto bod nám pridá najviac 2 trojuholníky. Potom, pridaním i bodov teda pridáme najviac $2i$ trojuholníkov. To znamená, že pre n bodov môžeme dostať až $(o-2)+2i$ trojuholníkov. Je zjavné, že $2i$ narastá rýchlejšie ako $(o-2)$ a teda ak chceme maximalizovať počet trojuholníkov musíme maximalizovať počet i. Avšak, treba si uvedomiť, že o musí byť minimálne 3 inač nemá zmysel hovoriť o trojuholníkoch. Preto, $i = n-3$.

Dostávame teda $(o-2)+2i = (3-2)+2(n-3) = 2n--5$ trojuholníkov. V dôsledku dualizmu Teda Voronoiov diagram nadobúda najviac $2n-5$ vrcholov.



Obr. 4.99. Body o nám tvoria o-2 trojuholníkov



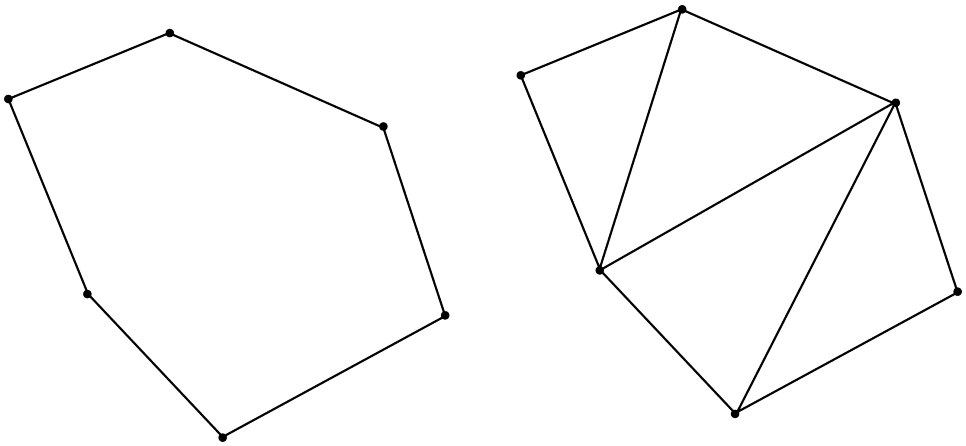
Obr. 4.100. Pridaním vnútorných bodu získame ďalších 2i trojuholníkov

Počet hrán ukážeme veľmi podobne. Opäť $n = i + oi$. Tentoraz ale body O tvoria v tringulácií $(o-3)+o$ bodov. To preto, že mnohouholník tvorený bodmi z O má o vonkajších hrán a jeho triangulácia tvorená diagonálami má vnútri $o-3$ hrán, keďže diagonál je $o-3$. Ak teraz pridáme bod z I ten nám pridá najviac 3 hrany. Preto, i bodov pridá najviac $3i$ hrán. Celkový počet hrán je teda nutne $(o-3)+o+3i$. Opäť, ak chceme maximalizovať počet hrán musíme maximalizovať i, lebo $3i$ narastá rýchlejšie ako $2o-3$. Samozrejme o opäť musí byť najmenej 3. Potom, hrán bude najviac $(o-3)+o+3i = (3-3)+3+3(n-3) = 3n-6$. Z duality má aj Voronoiov diagram najviac $3n-6$ hrán.

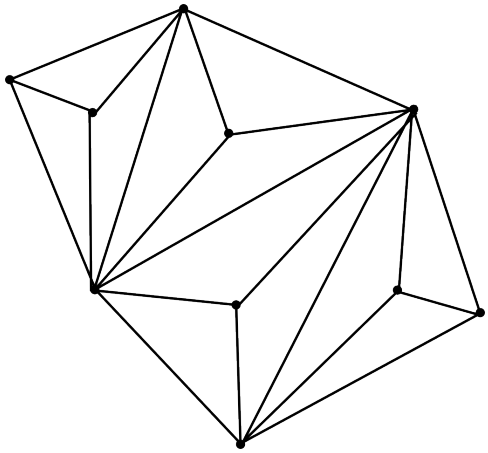
□

Cvičenie 4.35 (10 bodov). *Nájdi mnohouholník pre ktorý existuje rovnostranná triangulácia.*

Riešenie (Jakub Mišún, 30.11.2011): Rovnostranná triangulácia je taká triangulácia, ktorej každý trojuholník je rovnostranný. Keďže trojuholníky triangulácie susedia je jasné, že všetky trojuholníky takejto

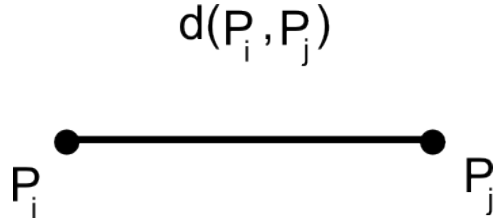


Obr. 4.101. Body o nám dájú 2o-3 hrán



Obr. 4.102. Pridaním i bodov dostávame ďalších 3i hrán

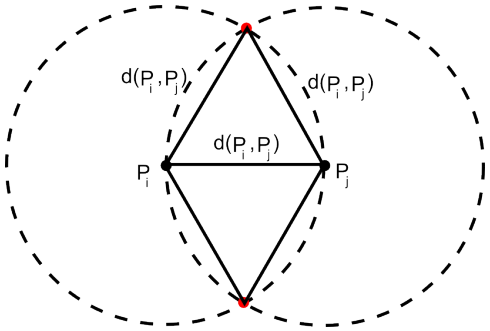
triangulácie sú rovnaké. Poďme si zostrojiť takúto množinu. Vezmime dva body P_i, P_j v rovine, pomocou nich vygenerujeme rovnostranné triangulácie, ktorých trojuholníky majú strany dĺžky $d(P_i, P_j)$



Obr. 4.103. Tieto dva body nám vygenerujú celú triangulaciu

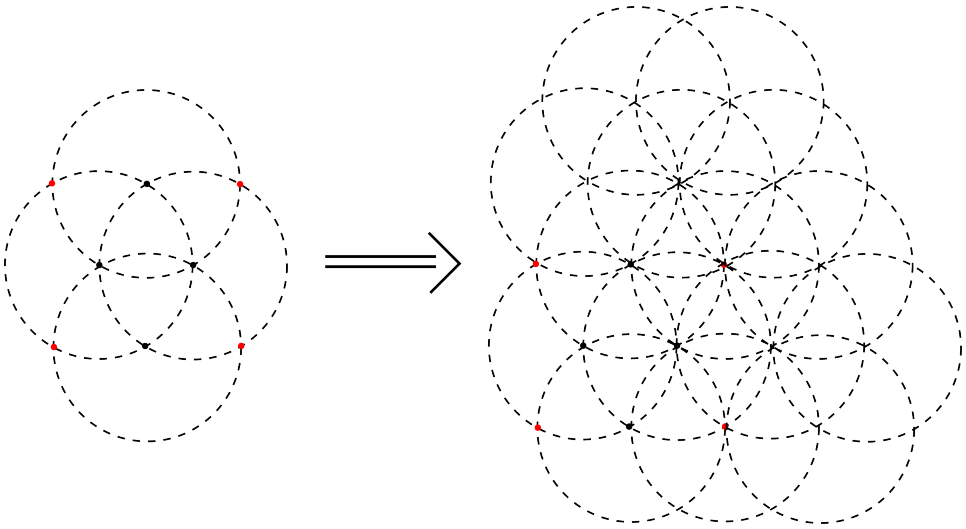
Tieto body tvoria jednu z hrán trojuholníka. Zostrojíme mu zvyšné dve hrany. Keďže má byť rovnostranný urobíme to tak, že jeden z nich

vezmeme za stred kružnice s polomerom $d(P_i, P_j)$, a rovnako aj druhý. Prienik oboch kružníc nám určí dva rovnoramenné trojuholníky.



Obr. 4.104. Tvorba rovnoramenných trojuholníkov

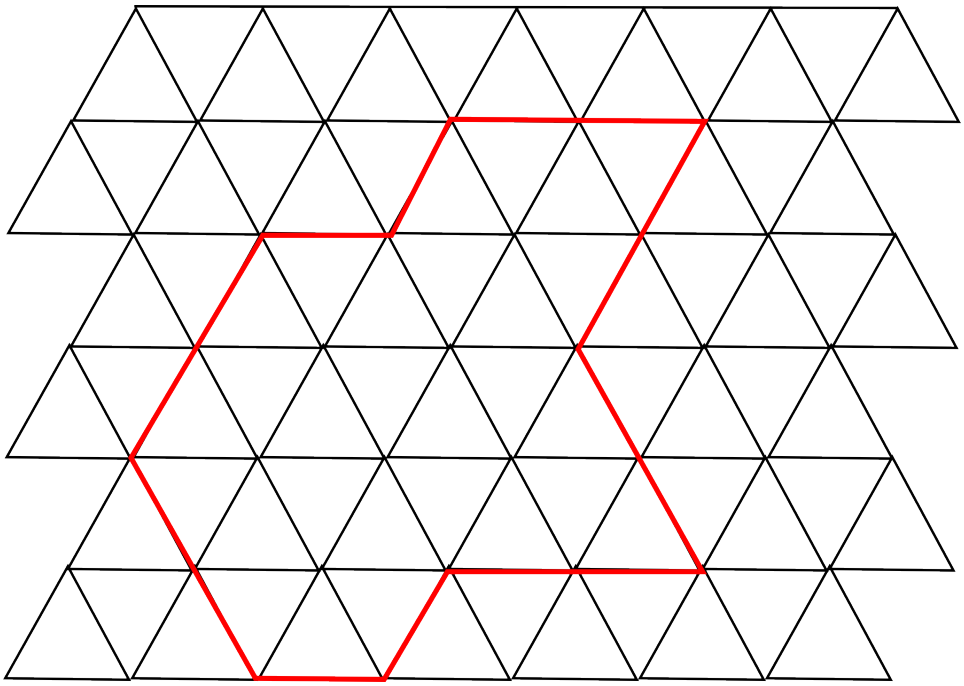
Dostali sme nové dva body, ktoré nám pomôžu pri generácií ďalších trojuholníkov. Jednoducho zopakujeme tvorbu trojuholníkov pre všetky nové ramená. Toto opakujeme ľubovolný počet krát.



Obr. 4.105. Trojuholníky generujeme donekonečna

Výsledne dostávame body v priestore, pre ktoré existuje rovnoramenná triangulácia. Tieto body vlastne tvoria trojuholníkové dláždenie roviny. Teraz nám už len stačí vziať ľubovolnú uzavretú lomenú čiaru tvorenú hranami trojuholníkov a ako výsledok dostávame mnohouholník, pre ktorý je možné zostrojiť rovnoramennú trianguláciu

Keďže sme tieto mnohouholníky generovali len pomocou dvoch bodov, dokážeme takto zostrojiť každý mnohouholník s rovnoramennou trianguláciou aký existuje, samozrejme až na škálovanie, pretože zmenou vzdialenosti počiatočných bodov dostávame rovnaké mnohouholníky, ibaže zmenšené, respektíve zväčšené. □



Obr. 4.106. Môžeme zostrojiť ľubovolný mnohouholník s rovnoramennou trianguláciou

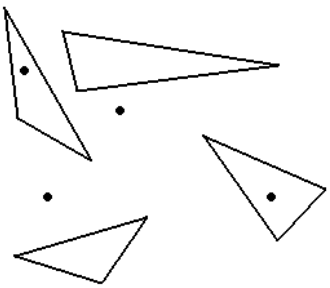
KAPITOLA 5

Geometrické vyhľadávanie

CVIČENIE 5.1 (20 bodov). *Nech S je množina n trojuholníkov v $\mathbb{E}^2$, ktorých hranice sú po dvoch disjunktné. Nech P je množina n bodov v $\mathbb{E}^2$. Navrhnite algoritmus, ktorý v čase $\mathcal{O}(n \log n)$ nájde všetky bod P, ktoré neležia v žiadnom z trojuholníkov z S.*

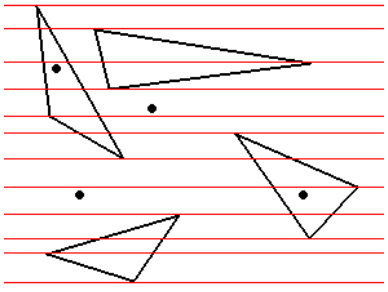
Riešenie (Dalibor Hrtánek, 5.12.2009):

Úlohu vyriešime metódou rovinných pásov (metódou vrstiev).



OBR. 5.1. Vstupná konfigurácia

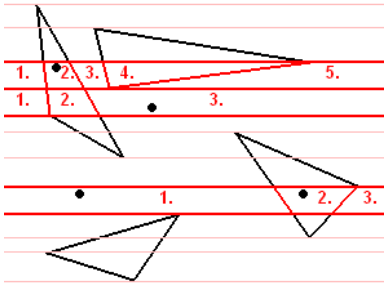
Všetkými vrcholmi trojuholníkov z S vedieme horizontálne priamky. Takýmto rozdelením roviny nám vznikne $3n + 1$ horizontálnych pásov (*Obr.2*). Na zistenie, v ktorom z týchto pásov sa nachádza bod $\mathbf{p}_i \in P$, si vo fáze predspracovania pásy usporiadame podľa rastúcej y-ovej súradnice. Vďaka tomuto usporiadaniu vieme nájsť požadovaný pás v čase $T : \mathcal{O}(\log n)$ pomocou binárneho vyhľadávania.



OBR. 5.2. Vytvorenie horizontálnych pásov

Teraz musíme určiť, kde sa bod $\mathbf{p}_i \in P$ nachádza v rámci nájdeného pásu. Rozdelíme si teda pás na oblasti. Pretože pracujeme v

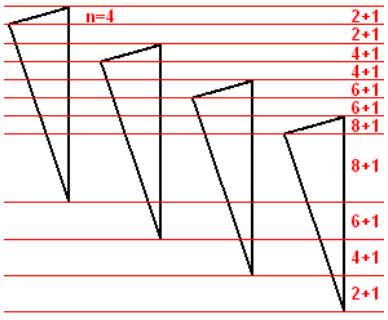
rovine a všetky hrany trojuholníkov z S sú úsečky, prienik pásu s množinou trojuholníkov z S môže byť len prázdna množina alebo množina lichobežníkov, ktoré môžu prejsť do trojuholníkov. Ak v predspracovaní zoradíme hrany týchto lichobežníkov podľa x-ovej súradnice zľava doprava, pomocou binárneho vyhľadávania vieme zistiť oblasť v ktorej sa bod $\mathbf{p}_i \in P$ nachádza v rámci tohto pásu v čase $T : \mathcal{O}(\log n)$. Ak je poradové číslo nájdenej oblasti v rámci pásu nepárne, bod $\mathbf{p}_i \in P$ neleží v žiadnom z trojuholníkov z S(*Obr.3*).



Obr. 5.3. Zoradenie oblastí v rámci pásu

Tento algoritmus má časovú zložitosť $T : \mathcal{O}(\log n) + \mathcal{O}(\log n) = \mathcal{O}(\log n)$. Keď ho použijeme pre zistenie polohy každého bodu z P, dostávame celkovú časovú zložitosť algoritmu $T : \mathcal{O}(n \log n)$.

Vo fáze predspracovania musíme triediť $3n+1 \in \mathcal{O}(n)$ pásov, pričom každý môže mať až $2n + 1 \in \mathcal{O}(n)$ oblastí (*Obr.4*). Pamäťová zložitosť je teda $M : \mathcal{O}(n^2)$ a s použitím optimálneho triediaceho algoritmu dostávame časovú zložitosť predspracovania $P : \mathcal{O}(n^2 \log n)$.

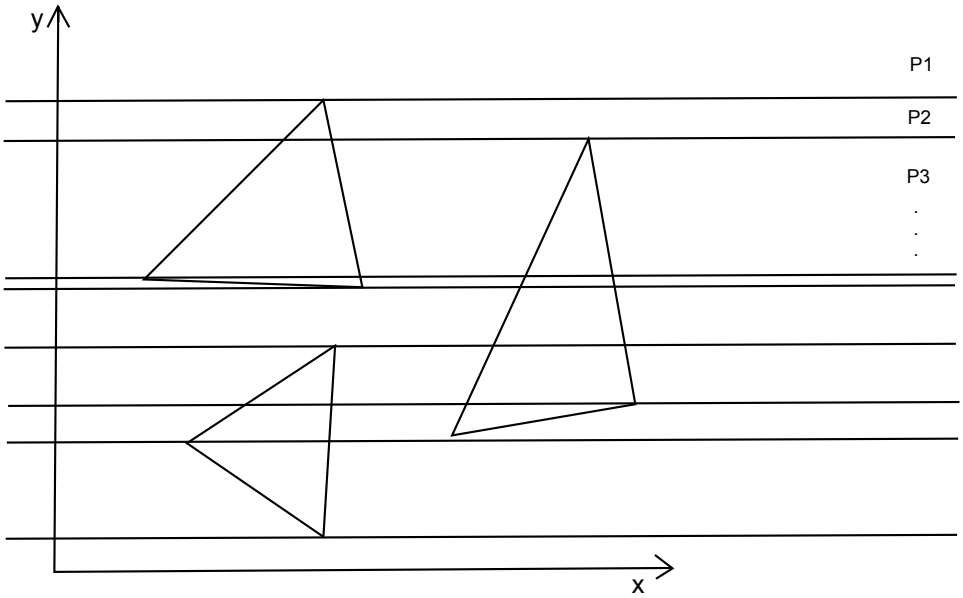


Obr. 5.4. Počty oblastí v jednotlivých pásoch

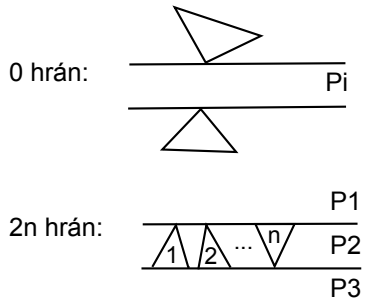
□

Riešenie (Barbora Gallusová, 4.12.2009):

V scéne máme n trojuholníkov, teda $3n$ vrcholov. Rozdeľme scénu tak, že z každého vrchola budeme viesť priamku rovnobežnú s osou x, poz. obr. 5.5.



Obr. 5.5. Rozdeľme scénu tak, že z každého vrchola budeme viesť priamku rovnobežnú s osou x.



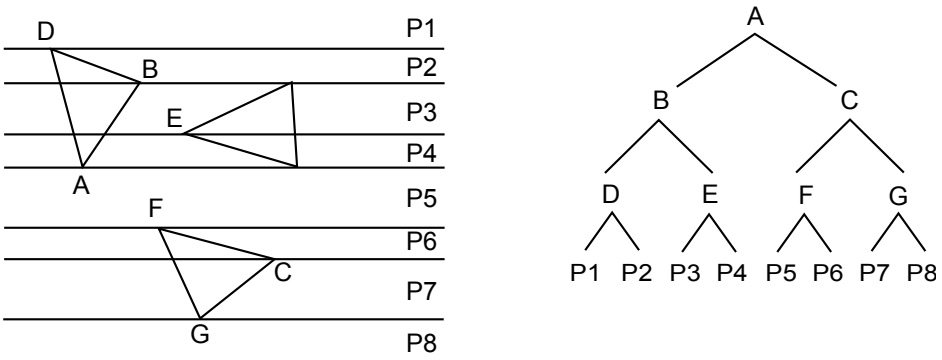
Obr. 5.6. Každý pás obsahuje najviac $2n$ hrán (buď celú hranu alebo jej časť), najmenej 0 hrán.

Dostaneme tak najviac $3n$ priamok (pre vrcholy s rovnakou y-súradnicou vedieme len jednu priamku), ktoré scénu rozdelia na najviac $3n+1$ pásov $P_1, P_2, \ldots$. Každý pás obsahuje najviac $2n$ hrán (buď celú hranu alebo jej časť), najmenej 0 hrán, poz. obr. 5.6.

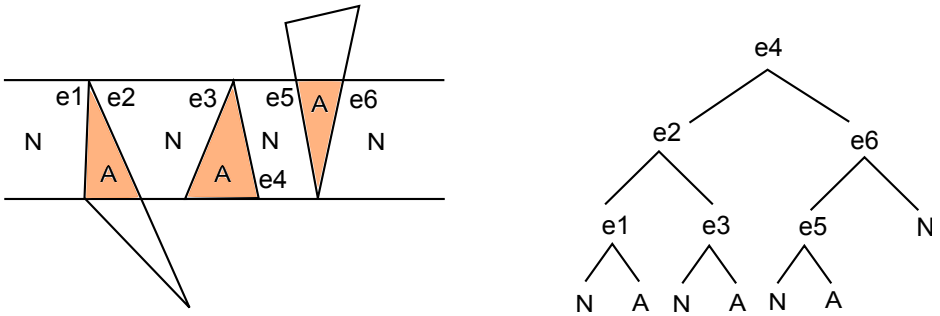
Vytvorme binárny vyhľadávací strom (vyvážený) pre pásy tak, že utriedime do tohto stromu vrcholy trojuholníkov, z ktorých robíme priamky podľa y-súradnice a v listoch budú pásy podľa toho, či sa nachádzajú nad daným vrcholom (ľavý syn) alebo pod vrcholom (pravý syn), poz. obr. 5.7.

Pre každý z n bodov vieme povedať v čase $O(\log n)$, v ktorom z pásov sa nachádza ($\log n$ - výška váženého binárneho stromu s $O(n)$ prvkami; priamok je najviac $3n$, pásov najviac $3n+1$).

Teraz potrebujeme určiť polohu bodu v rámci pásu a hneď budeme vedieť, či sa nachádza v trojuholníku alebo nie. Vytvorme si teda pre



Obr. 5.7. Tvorba binárneho vyhľadávacieho stromu pre pásy - utriedenie vrcholov.



Obr. 5.8. Tvorba binárneho vyhľadávacieho stromu pre pásy - utriedenie hrán.

pás tiež vyvážený binárny strom, že utriedime do stromu hrany (alebo ich časti) podľa x-súradnice bodu, čo leží v ich strede (v strede časti hrany) a v listoch majme uložené, či je napravo (naľavo) od hrany trojuholník, alebo nie, poz. obr. 5.8.

Pre každý z n bodov vieme povedať v čase $\mathcal{O}(\log n)$, v ktorej časti pásu sa nachádza, a teda či leží v nejakom trojuholníku alebo nie ($\mathcal{O}(\log n)$ – výška váženého binárneho stromu s $\mathcal{O}(n)$ prvkami, hrán v páse je najviac $2n$, regiónov $2n+1$). Časová zložitosť zistenia počtu bodov, ktoré neležia v žiadnom z trojuholníkov, je teda:

T: $\mathcal{O}(n).[\mathcal{O}(\log n) + \mathcal{O}(\log n)] = \mathcal{O}(n \log n)$

$\mathcal{O}(n)$ – pre každý bod

$\mathcal{O}(\log n)$ – v ktorom páse sa nachádza

$\mathcal{O}(\log n)$ – kde v páse sa nachádza (a teda či v trojuholníku alebo nie) podľa x-súradnice bodu

Túto zložitosť vieme dosiahnuť, len ak už máme predspracované stromy, ktoré používame. Aká je zložitosť predspracovania? Vytvorenie stromu pre pásy je $\mathcal{O}(n \log n)$ - tvorba vyváženého binárneho stromu a vytvorenie stromu pre každý pás je $\mathcal{O}(n)$(počet pásov). $\mathcal{O}(n \log n)$ (tvorba vyváženého binárnehostromu)$= \mathcal{O}\left(n^2 \log n\right)$. □

Cvičenie 5.2 (20 bodov). *V rovine je daných n bodov. Popíšte algoritmus, ktorý zistí počet bodov vnútri zvoleného obdĺžnika, ktorého strany sú rovnobežné so súradnicovými osami. Pri algoritme je povolené predspracovanie tak, aby pri viacnásobných požiadavkách bola doba vyhľadania menšia ako $\mathcal{O}(n)$.*

Riešenie (Martina Bátorová, 16.11.2008):

Ukážeme si metódu, ktorá používa predspracovanie bližšie popísané v skriptách *Zložitosť geometrických algoritmov*, str. 100-102 a dokáže pre pevnú vstupnú množinu a ľubovoľný obdĺžnik odpovedať v čase $\mathcal{O}(\log n)$.

Bez ujmy na všeobecnosti predpokladajme, že sa všetky body nachádzajú v prvom kvadrante (ak to neplatí, zmenou sústavy súradníc dosiahneme požadovaný stav). Základom predpracovania je rozdelenie roviny na časti, v ktorých je odpoveď na opakované požiadavky konštantná, tzv. *bunky*. Tieto vytvoríme tak, že každým bodom vstupnej množiny vedieme horizontálnu a vertikálnu priamku. Takto nám vznikne $\mathcal{O}(n^2)$ buniek. Opakovanou požiadavkou je otázka: "koľko bodov vstupnej množiny sa nachádza juhozápadne od bodu $\mathbf{p}$?". Ak je odpoveď k, hovoríme, že $\mathbf{p}$ *dominuje* k bodom, zapisujeme $N(\mathbf{p}) = k$. Zároveň vidíme, že ak vezmeme dva body z jednej bunky, odpoveď sa nezmení.

Vďaka predspracovaniu vieme problém príslušnosti do obdĺžnika popísať jazykom dominancie jeho vrcholov. Stačí zistiť, v ktorej bunke sa nachádzajú body $\mathbf{q}_1, \ldots, \mathbf{q}_4$. Na túto otázku vieme odpovedať v čase $\mathcal{O}(\log n)$, pretože na každú bunku sa vieme pozrieť ako na karteziánsky súčin intervalov. Využitím princípu inklúzie a exklúzie vyjadríme počet bodov vo vnútri obdĺžnika ako

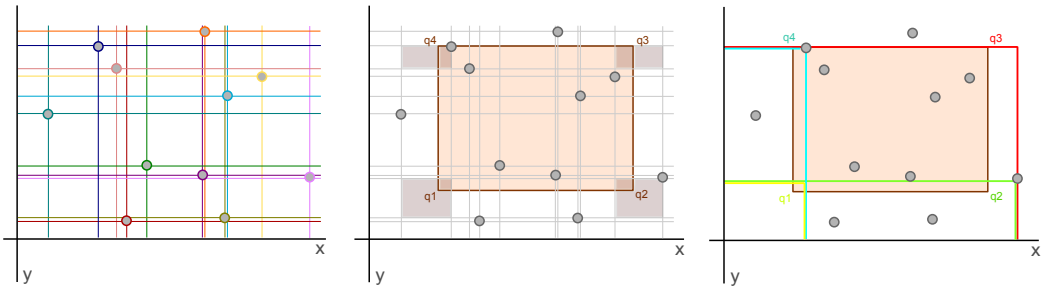
$$k = N(\mathbf{q}_3) - [N(\mathbf{q}_2) + N(\mathbf{q}_4)] + N(\mathbf{q}_1)$$

(viď obrázok vpravo, hodnotu $N(\mathbf{q}_1)$ musíme pripočítať, pretože sme ju odpočítali dvakrát v $N(\mathbf{q}_2), N(\mathbf{q}_4)$).

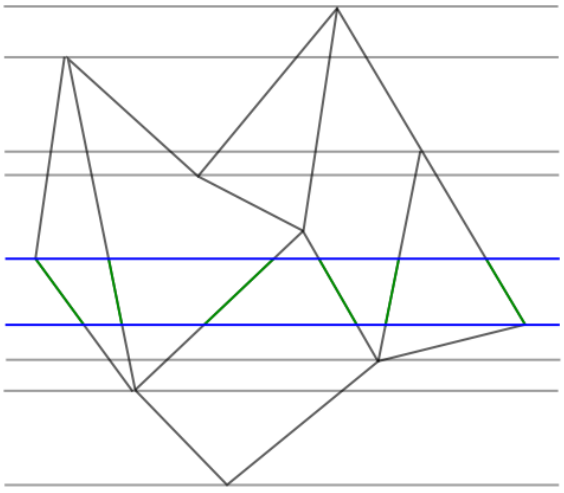
Metóda pri uvedenom predspracovaní vyžaduje $4\mathcal{O}(\log n) + \mathcal{O}(1) = \mathcal{O}(\log n)$ operácií. □

Cvičenie 5.3 (30 bodov). *V rovine je daný rovinný graf s n vrcholmi popíšte algoritmus, ktorý zistí pre zadaný bod, do ktorej z oblastí grafu patrí. Uveďte zložitosť predspracovania, zložitosť vyhľadania ako aj pamäťovú náročnosť algoritmu.*

Riešenie (Júlia Kučerová, 12.12.2010): Na riešenie zadaného problému použijeme metódu vrstiev. Majme zadaný rovinný graf s priamymi hranami ($RGPH$), označme si ho G. Tento graf je rovinný, čo znamená, že hrany sa pretínajú len vo vrcholoch, ktoré sú ich koncovými bodmi. Hľadaný bod si označme A. Každým vrcholom grafu G



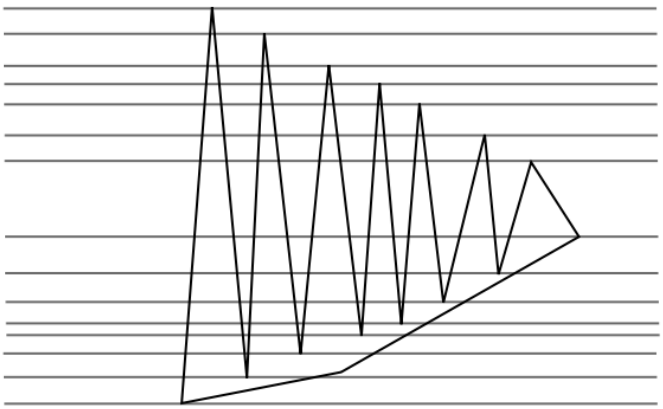
OBR. 5.9. Na obrázku vľavo je rozdelenie roviny na bunky. Na obrázku v strede je lokalizácia buniek, v ktorých ležia vrcholy obdĺžnika. Vpravo pre jednotlivé vrcholy berieme hodnoty $N(\mathbf{q}_i)$ podľa severozápadného bodu bunky, do ktorej sme daný vrchol zaradili.



OBR. 5.10. Rovinný graf s priamymi hranami rozdelený na pásy. Jeden je zvýraznený.

budeme viesť horizontálnu priamku, čím rovinu rozdelíme na $n+1$ oblastí, ktoré nazývame vrstvy. Vo vytvorených vrstvách je prienik ľubovoľnej oblasti grafu G buď prázdna množina, alebo lichobežník, ktorý môže degenerovať na trojuholník.

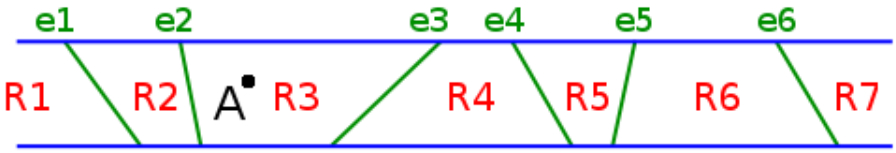
Časti hrán obsiahnuté vo vrstvách budeme nazývať hranové segmenty.Hranové segmenty v každej vrstve môžme zoradiť zľava doprava. Označme si L ako zoznam vrstiev, ktoré máme zoradené podľa y-ovej súradnice. To znamená, že akýkoľvek bod vo vrstve $L[j]$ má väčšiu y-ovú súradnicu, ako bod vo vrstve $L[i]$, kde $i < j$. Každý prvok $L[i]$ v zozname L má pointer na zoznam hranových segmentov l_i v zodpovedajúcej vrstve, zoradených zľava doprava. Graf G má n vrcholov, teda



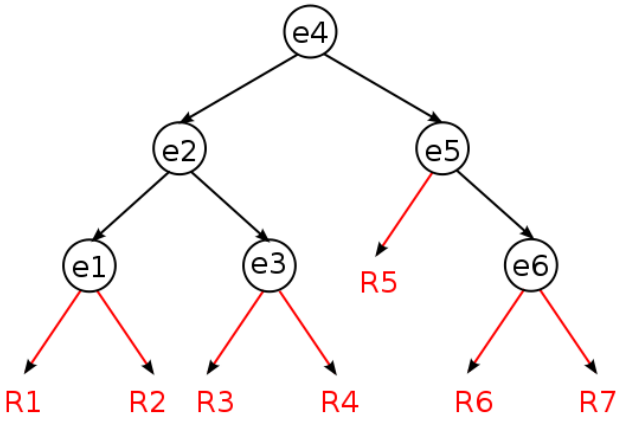
Obr. 5.11. Príklad, kde pamäťová zložitosť stromov pre jednotlivé pásy dosahuje zložitosť $\Omega(n^2)$

vieme lokalizovať hľadaný bod v čase $\mathcal{O}(\log n)$. Ak je G planárny graf, má $\mathcal{O}(n)$ hrán. Každá hrana G môže poskytnúť najviac jeden hranový segment danej vrstve. Teda ohraničenie počtu hranových segmentov každej vrstvy je $\mathcal{O}(n)$. Dva za sebou nasledujúce hranové segmenty vo vrstve jednoznačne určujú oblasť grafu G. Ak každej oblasti medzi hranovými segmentami v každej vrstve priradíme označenie oblasti, ktorú určujú, potom vieme presne určiť oblasť, ktorej patrí hľadaný bod A. Teda vyhľadávanie vo vrstve je v $\mathcal{O}(\log n)$. Teraz si určime, ako vytvoríme a budeme ukladať utriedený zoznam L a utriedené podzoznamy l_i. Podľa analýzy vyššie vieme, že l_i obsahuje najviac $\mathcal{O}(n)$ hranových segmentov, teda priestor na uloženie zoznamu L a podzoznamov l_i je ohraničený $\mathcal{O}(n^2)$. Môže ale nastať situácia, kde má rovinný graf štruktúru ako na obr. 5.11 a hranových segmentov vo vrstvách je $\Omega(n^2)$.

Priama metóda na vytvorenie týchto zoznamov je utriedenie vrcholov grafu G podľa y-ovej súradnice, vytvorenie zoznamu L a potom utriedenie každého podzoznamu l_i. Potom je časová náročnosť získania zoznamu L určená ako $\mathcal{O}(n \log n)$ a časová náročnosť získania podzoznamov l_i, $i = 1, 2, \ldots, n+1$ bude $\mathcal{O}((n+1)n \log n) = \mathcal{O}(n^2 \log n)$. Toto ale môžme zlepšiť. Pri zlepšovaní časovej náročnosti použijeme fakt, že každá hrana grafu G prechádza susediacimi vrstvami a teda sa jej poradie vzhľadom na ostatné susediace hrany nemení. Zametaciu priamku budeme posúvať zdola nahor. Stav zametacej priamky bude dátová štruktúra, ktorá si bude pamätať hranové segmenty vrstvy. Zmeny stavu zametacej priamky budú nastávať len vo vrcholoch grafu G, keďže len tu hrany končia, alebo začínajú. Hrany si orientujeme zdola nahor, čím vylúčime horizontálne hrany. Zmena stavu pozostáva



Obr. 5.12. Jeden pás grafu s priamymi hranami



Obr. 5.13. Binárny strom reprezentujúci pás grafu s priamymi hranami

z vynechania hrán, ktoré v danom vrchole končia, prípadne pridania hrán, ktoré v ňom začínajú. Pri zachovaní vyváženosti stromu každé zaradenie, alebo vynechanie hrany bude mať časovú zložitosť $\mathcal{O}(\log n)$.

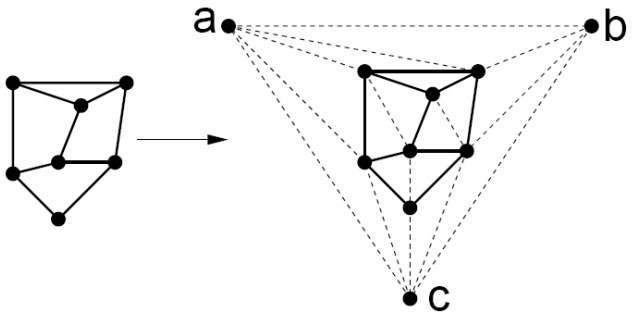
Príklad

Majme na vstupe graf G (obr. 5.10), kde máme zvýraznenú jednu vrstvu. Túto vrstvu si popíšeme detailnejšie. Označme hranové segmenty, ktoré sa nachádzajú v tejto vrstve ako $e1, \ldots, e6$ a oblasti, ktoré nám tieto segmenty orčujú označme $R1, \ldots, R7$. Na obrázku 5.12 máme vyznačený bod A. Chceme určiť, v ktorej oblasti sa tento bod nachádza. Majme binárny vyhľadávací strom (poz. obr. 5.13), kde vo vrcholoch budú hranové segmenty a v listoch oblasti. Takto môžeme jednoducho zistiť, v ktorej oblasti sa zadaný bod nachádza. Časová náročnosť pre lokalizáciu bodu vzhľadom na rovinný graf s n vrcholmi je pre túto metódu $\mathcal{O}(n \log n)$, pamäťová náročnosť je $\mathcal{O}(n^2)$ a náročnosť predspracovania je tiež $\mathcal{O}(n^2)$.

(Zdroj: Computational Geometry: Methods and Applications, Jianer Chen) □

Riešenie (Kristína Lidayová, 29.11.2011): Riešime pomocou metódy postupných triangulácií, ktorej autorom je Kirkpatrick. Je to metóda dosahujúca optimálnu pamäťovú aj časovú zložitosť. Pre praktické použitie sa ale príliš nehodí, a tak slúži len na porovnanie ostatných metód.

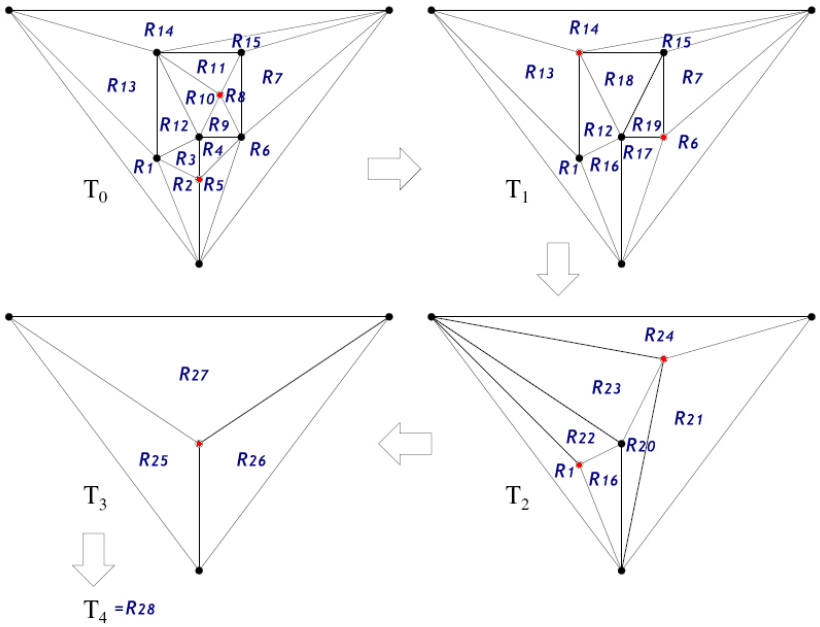
Bez ujmy na všeobecnosti predpokladáme, že vstupom je triangulácia s troma vonkajšími hranami. Pokiaľ nie je splnený predpoklad triangulácie, pridáme do grafu hrany tak, aby rovinný graf bol trianguláciou. Pokiaľ nie je splnená podmienka práve troch vonkajších hrán, najprv vypočítame konvexný obal všetkých vrcholov a následne obkolesíme tento konvexný polygón veľkým trojuholníkom s vrcholmi a, b, c. Nakoniec ešte pridáme hrany medzi vrcholy konvexnej množiny a vrcholy a, b, c patriace veľkému trojuholníku. Keďže trianguláciu polygónu vieme vykonať v čase $\mathcal{O}(n \log n)$, prípadne v čase $\mathcal{O}(n)$, pokiaľ použijeme sofistikovanejšiu metódu, dokážeme splniť vstupné podmienky v čase nanajvýš $\mathcal{O}(n \log n)$. Toto je teda zložitosť predspracovania.



Označme túto vstupnú trianguláciu T_0. Naša popisovaná metóda vytvorí postupnosť triangulácií $T_0, T_1, T_2, ..., T_k$, pričom triangulácia T_k bude pozostávať už len z jedného trojuholníka. Každý trojuholník z triangulácie T_{i+1} prekrýva istý počet trojuholníkov v triangulácií T_i. Pokiaľ zistíme, v ktorom trojuholníku z triangulácie T_{i+1} sa nachádza hľadaný bod, budú nás v ďalšom kroku zaujímať len tie trojuholníky z triangulácie T_i, ktoré tento trojuholník prekrýva. Postupnosť triangulácií $T_0, T_1, T_2, ..., T_k$ zostrojíme nasledovne. Predpokladajme, že máme trianguláciu T_i a chceme vytvoriť trianguláciu T_{i+1}. Aby sme mohli zaručiť, že tento proces bude po $\mathcal{O}(\log n)$ opakovaniach ukončený, musíme ešte zaistiť, že počet vrcholov v T_{i+1} bude zakaždým menší ako počet vrcholov v T_i.

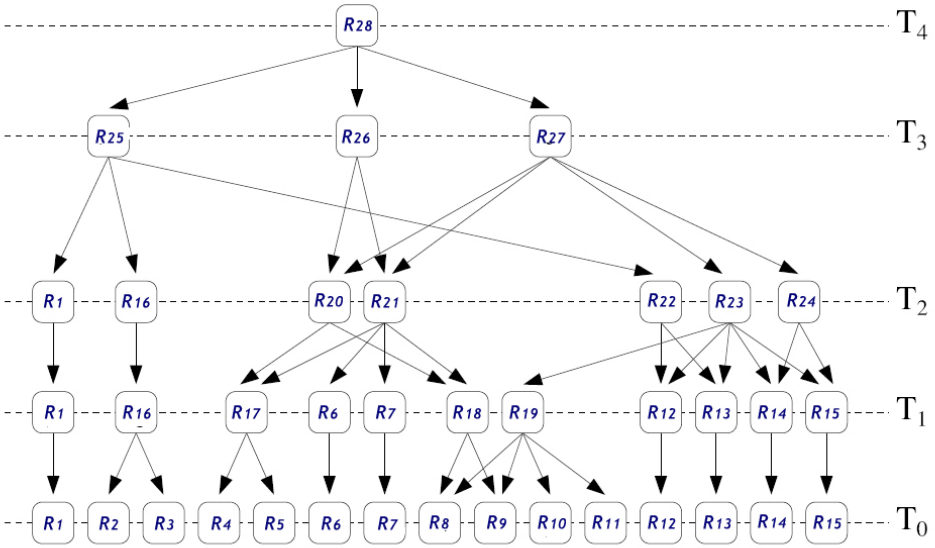
To zabezpečíme starostlivým výberom a zmazaním nezávislej podmnožiny vrcholov z T_i. Spoločne s vrcholmi zmažeme aj všetky hrany incidentné s týmito vrcholmi. Následne mnohouholníky vzniknuté vynechaním vrcholov opäť triangulujeme a dostávame tak trianguláciu T_{i+1}. Končíme, keď dostaneme trianguláciu T_k pozostávajúcu už len z

jedného trojuholníka.

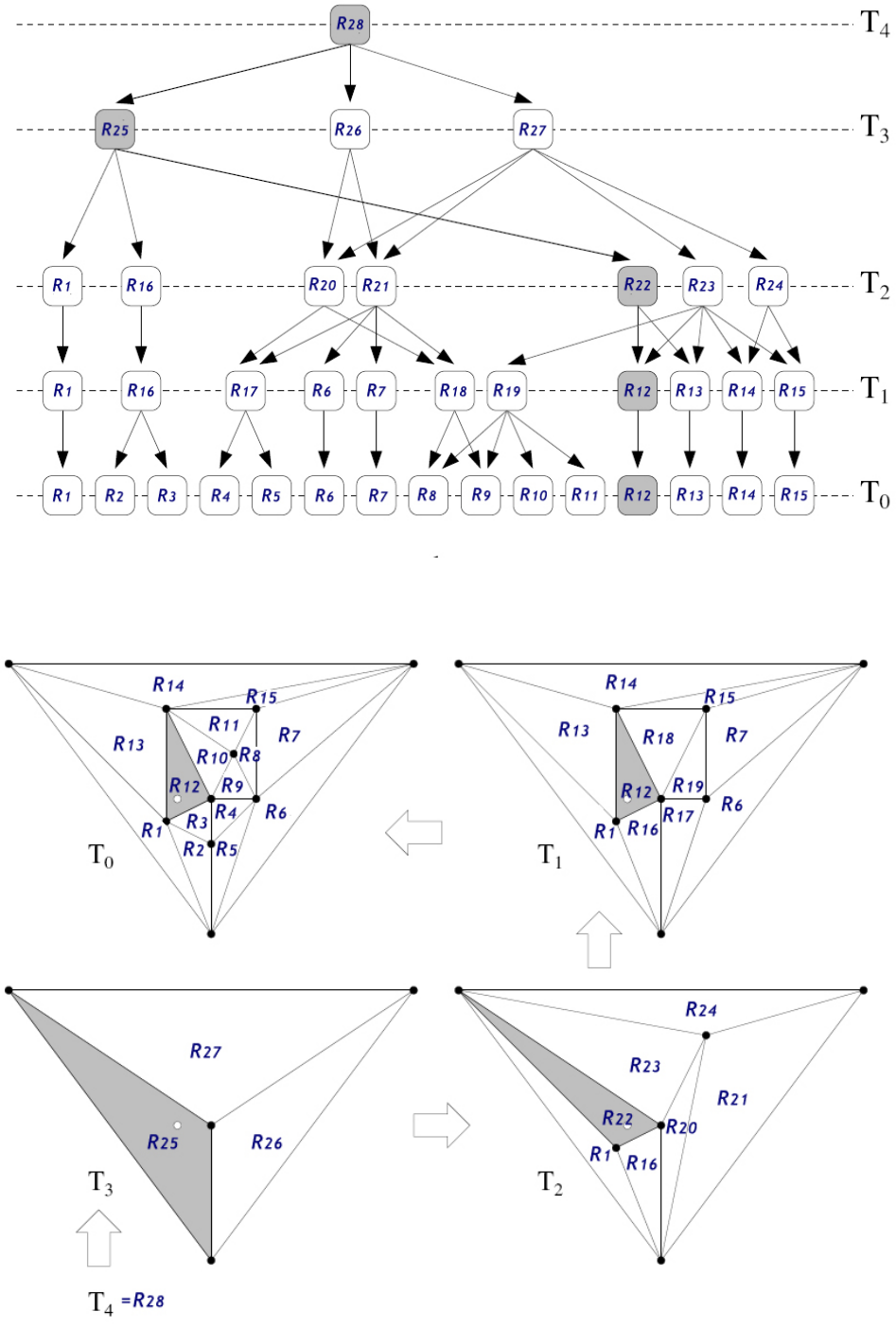


Vrcholy určené na zmazanie vyberáme tak, aby každý vrchol mal stupeň $< d$ (mal nanajvýš d hrán). Zmazaním takéhoto vrchola vznikne nanajvýš d-uholník, ktorý sa dá triangulovať $d-2$ trojuholníkmi. Každý nový trojuholník tak prekrýva nanajvýš $d-2$ trojuholníkov z predchádzajúcej triangulácie. Ďalšou podmienkou na výber vrcholov určených na zmazanie je, že to nesmú byť hraničné vrcholy a žiadne dva nesmú byť navzájom susedné. Čiže musia byť z nezávislej množiny vrcholov v aktuálnom planárnom grafe T_i. To zaručí jednoduchšiu opätovnú trianguláciu.

Vytvoríme prehľadávaciu štruktúru s topológiou orientovaného koreňového acyklického grafu. Vrcholy tohto grafu odpovedajú trojuholníkom v jednotlivých trianguláciach. Tieto trojuholníky vystupujúce v triangu láciach označujeme R_j *j=1..počet trojuholníkov* . Trojuholník môže súčasne patriť do viacerých triangulácií a preto ho označíme hneď v tej triangulácií, v ktorej vznikol (teda tej, ktorá má menší index). Z trojuholníka R_j pôjde orientovaná hrana do R'_j, ak R_j je z triangulácie T_i, R'_j je z triangulácie T_{i-1} a $R_j \cap R'_j \neq \emptyset$ (vnútra sú neprázdne).



Koreň štruktúry zodpovedá jednoduchému trojuholníku T_k a na najnižšej vrstve sú listy, ktoré zodpovedajú trojuholníkom vstupnej triangulácie T_0. Pri lokalizácií bodu začneme v koreni T_k. Pokiaľ hľadaný bod neleží v jednoduchom trojuholníku triangulácie T_k, tak sme skončili. Bod leží v exteriéry, mimo trojuholníka. Pokiaľ leží vo vnútri prehľadávame každý trojuholník z triangulácie T_{k-1} , ktorý má s trojuholníkom z T_k neprázdne vnútro. Môže ich byť maximálne $d-2$. Keď nájdeme ten, v ktorom sa nachádza náš hľadaný bod, prehľadávame každý trojuholník z triangulácie T_{k-2}, ktorý má s týmto neprázdne vnútro a tak ďalej. Napokon nájdeme trojuholník, ktorý obsahuje hľadaný bod v poslednej triangulácii T_0 a to je požadovaný výsledok.



Zložitosť metódy postupných triangulácií:
Označme n_i počet vrcholov triangulacie T_i, e_i počet jej hrán a f_i počet jej stien. Celková zložitosť algoritmu ako aj dátovej štruktúry závisí od výberu množiny nezávislých vrcholov v triangulácii T_i. Chceme, aby platili dve podmienky:

(1) Oblasť R_j má nanajvýš h synov a nanajvýš h otcov. Ak graf má n_0 vrcholov, tak hrán je nanajvýš hn_0
(2) Počet vrcholov klesá aspoň geometricky rýchlo. Nech $n_{i+1} = \alpha_i n_i$, kde $0 < \alpha_i \leq \alpha < 1$ pre $i = 0, \ldots, k-1$,

$$n_0 + n_1 + \cdots + n_k \leq n_0 + \alpha n_0 + \alpha^2 n_0 + \cdots + \alpha^{k-1} n_0 = n_0 \tfrac{1-\alpha^k}{1-\alpha}$$
$$\tfrac{1-\alpha^k}{1-\alpha} > 1$$

kde $\alpha \in (0, 1)$

Obidve podmienky spolu zabezpečujú odhad $\mathcal{O}(n)$ pre veľkosť štruktúry. Počet všetkých vrcholov všetkých triangulácií počas celého algoritmu je $\mathcal{O}(n)$, $n = n_0$. Všetkých hrán v grafe je $\mathcal{O}(hn)$=$\mathcal{O}(n)$, lebo h je konštanta. Z Eulerovej vety pre rovinný graf vieme, že $e \leq 3n-6$. V našom prípade, keď je rovinný graf plne triangulovaný platí rovnosť. Ak chceme sčítať stupne všetkých vrcholov, každá hrana sa tam vyskytne 2x a dostávame:

$$\textstyle\sum_v deg(v) = 2e = 6n - 12$$

V rovinnom grafe existuje nanajvýš $\frac{n}{2}$ vrcholov, ktorých stupeň je aspoň 12. Položme $d = 11$, čiže budeme vynechávať vrcholy, ktorých stupeň je nanajvýš 11. Vynechaním vrcholu vznikne nanajvýš 11-uholník, ktorý sa dá triangulovať h=9 trojuholníkmi. Teda každý trojuholník incidentný s vynechaným vrcholom bude mať nanajvýš 9 synov v grafe. Vynechávame maximálnu takú množinu nezávislých vrcholov.

Nech v je počet vynechaných vrcholov, potom $3 + 11v \geq \frac{n}{2} - v$ (3 hraničné vrcholy, 11 susedných vrcholov k vynechaným vrcholom).

$$3 + 11v \geq \tfrac{n}{2} - v$$
$$12v \geq \tfrac{n}{2} - 3$$
$$v \geq \lfloor \tfrac{1}{24} n - \tfrac{3}{12} \rfloor$$

Pre dostatočne veľké n vynecháme aspoň $\lfloor \frac{1}{24} n \rfloor$ vrcholov, teda $n_{i+1} \leq (1 - \frac{1}{24}) n_i$, z toho $\alpha = (1 - \frac{1}{24}) = \frac{23}{24}$. Kvôli hodnote α blízke 1 sa táto metóda reálne nepoužíva a slúži len na porovnávanie ostatných metód.

Výška našej dátovej štruktúry je úmerná $\mathcal{O}(\log n)$, pretože počet vrcholov triangulácií klesá geometricky. Keďže posunúť sa z úrovne triangulácie T_i na úroveň T_{i-1} trvá konštantný čas, celkový čas behu algoritmu je $T : \mathcal{O}(\log n)$. Pamäťová náročnosť je $M : \mathcal{O}(n)$ a predspracovanie trvá $P : \mathcal{O}(n \log n)$.

Zdroj:

- *Lecture notes for the course Computational Geometry prepared by David Mount, Dept. of Computer Science, University of Maryland*
- *Poznámky z predmetu Výpočtová geometria*

□

CVIČENIE 5.4 (10 bodov). *Zistite, či je mnohouholník monotónny vzhľadom na daný smer.*

Riešenie (Lukáš Tencer, 6.11.2008):

Majme mnohouholník P s vrcholmi $(p_1, \ldots, p_n)$ a chceme zistiť, či je tento mnohouholník monotónny(tj. prienik mnohouholníku a ľubovoľnej priamky ortogonálnej k priamke $\bar{l}$ so smerom $\overrightarrow{v}$ je jeden súvislý komponent) na smer daný vektorom $\overrightarrow{v}$.

Algoritmus, ktorý rozhodne o monotónnosti P bude vyzerať nasledovne:

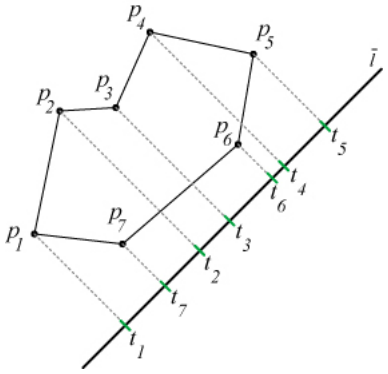
(1) Nájdeme si priamku $\bar{l}$ určenú vektorom $\overrightarrow{v}$ a ľubovoľným bodom v rovine.

(2) Urobíme kolmý priemet vrcholov mnohouholníka na priamku $\bar{l}$. Nech priemet vrcholu p_i je bod $\widetilde{p_i}$, potom pre každý bod $\widetilde{p_i}$ máme hodnotu t_i, ktorá nám hovorí o polohe bodu $\widetilde{p_i}$ na priamke $\bar{l}$. Hodnoty t_i, $i = 1, \ldots, n$, vieme získať pri premietaní vrcholov mnohouholníka P. Čiže hodnotu t_i vieme aj jednoznačne priradiť vrcholu mnohouholníka p_i. Pozri obr. 5.14.

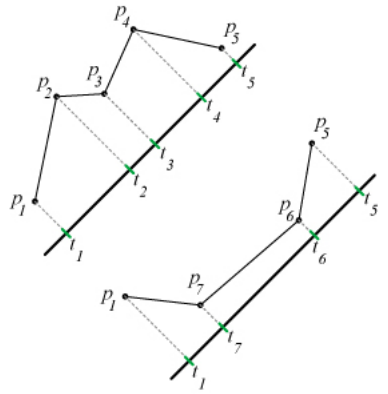
(3) Nájdeme minimálne a maximálne hodnoty t_i, $i = 1, \ldots, n$. Označme ich t_j a t_k, tieto zodpovedajú vrcholom mnohouholníka, pomocou ktorých ho rozdelíme na dve reťaze $R_1 = (p_j, p_{j+1}, \ldots, p_{k-1}, p_k)$ a $R_2 = (p_j, p_{j-1}, \ldots, p_{k+1}, p_k)$. Pre každú z týchto reťazí vieme nájsť postupnosť hodnôt t $T_1 = (t_j, t_{j+1}, \ldots, t_{k-1}, t_k)$ a $T_2 = (t_j, t_{j-1}, \ldots, t_{k+1}, t_k)$ zodpovedajú bodom v reťazi. Pozri obr. 5.15.

(4) Pokiaľ platí, že postupnosti T_1 a T_2 sú neklesajúce tak mnohouholník je monotónny vzhľadom na smer $\overrightarrow{v}$.

```
boolean jeMonotonny ( vrcholy[] , vektor v ) {
  p= priamka(v, bod(0,0));
  parametrePriemetov[] = kolmePriemety(vrcholy[]);
  tmin= minimum(parametrePriemetov);
  tmax= maximum(parametrePriemetov);
  retaz1= vrcholy[tmin, tmin.nasledovnik .. tmax.predchodca,
tmax];
  retaz2= vrcholy[tmin, tmin.predchodca .. tmax.nasledovnik,
tmax];
  for(int i=0, i<pocetPrvkov(retaz1)-1, i++){
    if(retaz1[i] > retaz1[i+1]){
      return false;
  }   for(int i=0, i<pocetPrvkov(retaz2)-1, i++){
    if(retaz2[i] > retaz2[i+1]){
      return false;
  }
  return true;
```



OBR. 5.14. Mnohouholník a kolmé priemety jeho vrcholov na priamku $\bar{l}$.



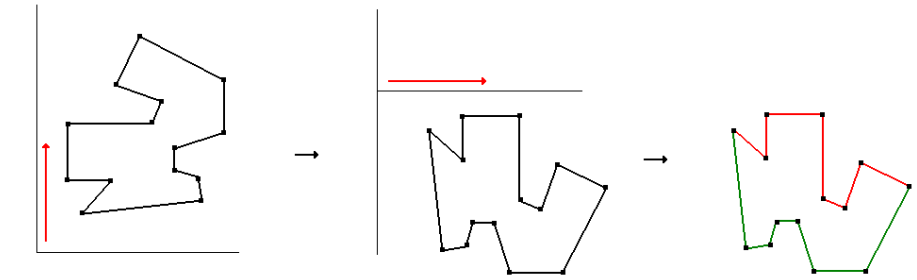
OBR. 5.15. Monotónne reťaze a k nim korešpondentné postupnosti parametrov t_i.

}

□

Riešenie (Jana Hlinková, 13.11.2008): Mnohouholník M je monotónny vzhľadom na daný smer, ak jeho hranicu δM môžeme rozdeliť na dve lomené čiary, ktoré sú monotónne vzhľadom na tento smer. Lomená čiara je monotónna vzhľadom na daný smer, ak jej prienik s každou priamkou kolmou na tento smer tvorí maximálne jeden súvislý komponent (prázdna množina, jeden bod alebo úsečka, pozri obr. 5.16).

Existuje jednoduchý spôsob ako overiť, či je mnohouholník M monotónny vzhľadom na x-ovú os (v lineárnom čase). Rovnako v lineárnom čase vieme uskutočniť rotáciu n bodov mnohouholníka v rovine okolo počiatku sústavy (jeden bod v konštantnom čase). V konštantnom čase vieme vypočítať uhol medzi dvomi vektormi. Teda aby sme zistili, či je mnohouholník monotónny vzhľadom na daný smer (u, v), vypočítame si uhol medzi týmto vektorom a x-ovou osou ϕ, zrotujeme



Obr. 5.16. Monotónnosť vzhľadom na y-ovú os. Po rotácii monotónnosť vzhľadom na x-ovú os.

každý vrchol mnohouholníka o uhol ϕ okolo bodu $(0, 0)$ a overíme, či je zrotovaný mnohouholník monotónny vzhľadom na os x.

Ešte popíšem algoritmus overenia monotónnosti vyhľadom na os x. Nájdeme vrchol s najmenšou x-ovou súradnicou a vrchol s najväčšou x-ovou súradnicou. Tieto dva vrcholy rozdeľujú mnohouholník na hornú a dolnú lomenú čiaru. Postupne prechádzame po vrcholoch týchto lomených čiar zľava doprava a overujeme, či ich x-ové súradnice nasledujú v neklesajúcom poradí.

Celý algoritmus beží v čase $\mathcal{O}(n)$:

- $\mathcal{O}(1)$ vypočítanie uhlu rotácie
- $\mathcal{O}(n)$ rotovanie n bodov
- $\mathcal{O}(n)$ nájdenie vrcholu s min a max x-ovou súradnicou
- $\mathcal{O}(n)$ overenie monotónnosti lomených čiar

Pamäťová zložitosť algoritmu je $\mathcal{O}(n)$, keďže využíva iba konštantný počet pomocných premenných.

```
//polygon is represented by array of points: Point2d[]

boolean isMonotone(Point2d[] M, Vector2d u)
{
        double fi = rotationAngle(u);
        Point2d[] rM = rotatePolygon(M, fi);
        return isXMonotone(rM);
}

boolean isXMonotone(Point2d[] M)
{
        Point2d minXPoint = getMinX(M);
        Point2d maxxPoint = getMaxx(M);
        return ( isLineMonotone(minXPoint, maxxPoint, M)
               & isLinMeonotone(maxxPoint, minXPoint, M) );
}
```

□

Cvičenie 5.5 (30 bodov). *Opíšte algoritmus triangulácie konvexného mnohouholníka a hviezdicového mnohouholníka.*

Riešenie (Lukáš Tencer, 17.11.2008):

Triangulácia konvexného mnohouholníka je triviálna a môžeme ju spraviť v lineárnom čase a to pridaním hrán z jedného vrcholu ku všetkým ostatným vrcholom. Pozri obr. 5.17.

Aby sme mohli korektne definovať algoritmus triangulácie hviezdicového mnohouholníku, objasnime si najprv niektoré pojmy.

Hviezdicový mnohouholník (star-shaped) je ten, v ktorom existuje bod z ktorého je vidno celý mnohouholník. Množina týchto bodov sa nazýva jadro (kernel). Pozri obr. 5.18.

Mnohouholník s hranovou viditeľnosťou (edge-visible) je taký, v ktorom existuje hrana e, na ktorej existuje bod p taký, že ľubovoľný bod mnohouholníku je viditeľný z bodu p. Pozri obr. 5.19.

Triviálnym prípadom pri triangulácii hviezdicového mnohouholníku je pokiaľ jadro obsahuje niektorý vrchol v mnohouholníku, v takomto prípade nám stačí spojiť vrchol v so všetkými ostatnými vrcholmi. Pozri obr. 5.20.

Ak chceme triangulovať ľubovoľný hviezdicový mnohouholník, postupujeme jeho rozdelením na hranovo viditeľné mnohouholníky a triangluáciu algoritmom troch mincí (three-coin algorithm) a následným spojením týchto častí pôvodne rozdeleného mnohouholníku. Pôvodný mnohouholník nemusel byť hranovo viditeľný(ak by bol použijeme algoritmus troch mincí priamo naň), avšak časti na ktoré ho rozdelíme už budú a tieto vieme triangulovať v lineárnom čase. Dôkaz algoritmu ako i podrobnejšie opísaný postup možno nájsť v *A. A. Schoone, J. van-Leeuwen "Triangulating a star-shaped polygon"*.

Teraz popíšem algoritmus a uvediem jeho jednotlivé kroky. Povedzme, že chceme triangulovať hviezdicový mnohouholník P.

(1) Majme bod k jadra P (algoritmus *Lee, Prepata 1979* nám ho nájde v čase $\mathcal{O}(n)$). Rozdelíme si mnohouholník P na 2 mnohouholníky s hranovou viditeľnosťou pridaním hrany e, ktorá prechádza bodom k a niektorým vrcholom p mnohouholníku P. Táto hrana môže pretínať niektorú z hrán mnohouholníku P čím nám vznikne nový bod s (nazýva sa aj Steinerov bod). Takto sme dostali dva nové mnohouholníky P_1 a P_2.
(2) Mnohouholníky s hranovou viditeľnosťou P_1 a P_2 triangulujeme pomocou algoritmu troch mincí.
(3) Spojíme mnohouholníky P_1 a P_2 a odstránime Steinerov bod p ako i všetky hrany triangulácie ktoré s ním súvisia.
(4) Diera v triagulácii, ktorá nám vynikla po odstránení bodu p, bude taktiež tvoriť mnohouholník s hranovou viditeľnosťou a tak na jej trianguláciu použijeme algoritmus troch mincí. To

že diera ktorá vznikla po odstránení bude tiež tvoriť mnohouholník s hranovou viditeľnosťou vyplýva s toho, že bod p, ktorý sme odstraňovali ležal na hrane a tak ľubovoľný vrchol diery z neho bude viditeľný.

Ako ilustráciu pozri obr. 5.21. Teraz si ešte uveďme algoritmus troch mincí na trianguláciu mnohouholníku s hranovou viditeľnosťou. V algoritme musíme počítať aj s orientáciou, my budeme mať ako východziu kladnú orientáciu.

V algoritme používame dve procedúry **posunmince()** a **odsekni()**. Procedúra **posunmince()** nám posunie poslednú z troch mincí za prvú mincu. Formálne ak máme ako mince vrcholy k_i, k_{i+1}, k_{i+2}, potom sa nám zmení postupnosť na $k_{i+1}, k_{i+2}, k_{i+3}$. Pozri obr. 5.22.

Procedúra **odsekni()** nám pre postupnosť vrcholov k_i, k_{i+1}, k_{i+2} ako mincí do triangulácie pridá hranu (k_i, k_{i+2}), odstráni vrchol k_{i+1} zo vstupnej postupnosti a presunie mincu ktorá na ňom bola na vrchol k_{i+3}. Pozri obr. 5.23.

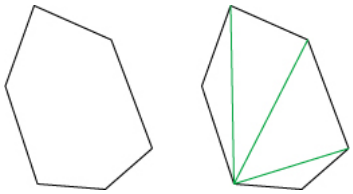
Samotný algoritmus bude vyzerať nasledovne:

Vrcholy u a v sú vrcholy hrany s ktorej je mnohouholník viditeľný. Ako vstup máme postupnosť v, k, $k+1$, ... , $k+n-2$, u.

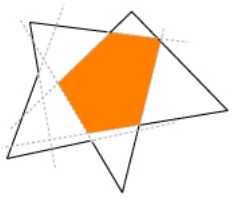
```
ThreeCoinsTriangulate ( mnohouholník P ) {
  minca1=v;
  minca2=k;
  minca3=k+1;
  while( minca3 != v ){
    if ( jeKonvexny( minca2 ) )
      odsekni(); // pridá hranu do triangulácie a posunie mincu
    else
      posunmince(); // posunie mince v smere orientácie
  }
}
```

Ak sa pozrieme na algoritmus neformálne, tak vidíme, že sa z mnohouholníku s hranovou viditeľnosťou postupne budú odrezávať trojuholníky pri vrcholoch, ktoré majú vnútorný uhol konvexný. Toto nám zaručí, že hrany pridané do triangulácie budú vždy vo vnútri trojuholníku a nikdy sa nebudú prekrývať. Vzhľadom na to, že trianguluje me mnohouholník s hranovou viditeľnosťou, tak každý trojuholník k_{i-1}, k_i, k_{i+1}, kde vnútorný uhol pri k_i je konvexný, bude uchom(pozri Ear cutting algoritmus) a teda hrana k_{i-1}, k_{i+1} bude celá vo vnútri mnohouholníku. Taktiež po odrezaní trojuholníka sa nám vlastnosť hranovej viditeľnosti zachová, čiže orezanie bude vždy korektné. Algoritmus ide po vrcholoch postupne a jediný prípad, kedy by neskončil je pokiaľ by mal mnohouholník všetky vnútorné uhly nekonvexné, avšak toto nemôže nastať.

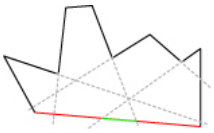
Pre ilustráciu algoritmu troch mincí pozri obr. 5.24. □



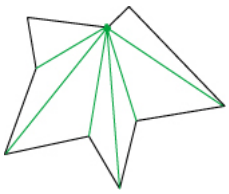
OBR. 5.17. Triangulácia konvexného mnohouholníku.



OBR. 5.18. Hviezdicový mnohouholník so zvýrazneným jadrom.



OBR. 5.19. Mnohouholník s hranovou viditeľnosťou.



OBR. 5.20. Triangulácia hviezdicového mnohouholníka pokiaľ jadro obsahuje vrchol.

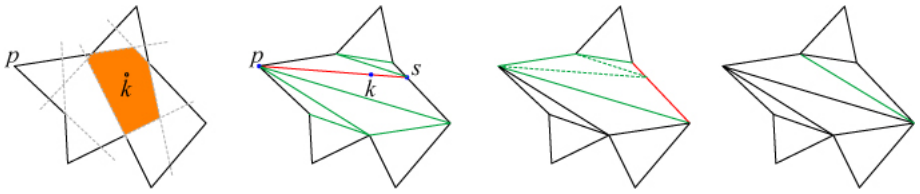
CVIČENIE 5.6 (30 bodov). *Nájdite algoritmus triangulácie jednoduchého mnohouholníka.*

Riešenie (Pavol Beluško, 23.11.2008): Predpokladajme, že jednoduchý mnohouholník má aspoň 4 vrcholy. V ďalšom texte sa pod pojmom mnohouholník chápe jednoduchý mnohouholník s aspoň 4 vrcholmi.

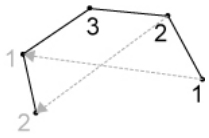
Definícia (ucho mnohouholníka): trojuholník $V_{i-1}V_iV_{i+1}$ sa nazýva ucho mnohouholníka určené vrcholom V_i, ak jeho strany $V_{i-1}V_iaV_iV_{i+1}$ sú súčasťou hranice mnohouholníka a ak celá strana $V_{i-1}V_{i+1}$ leží v mnohouholníku (obrázok 5.25).

Definícia (prekrývajúce sa uši): dve uši sa prekrývajú, ak ich vnútra majú neprázdny prienik.

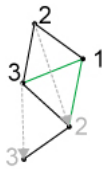
VETA 5.1. *Každý mnohouholník má aspoň 2 uši.*
Dôkaz:[indukcia]



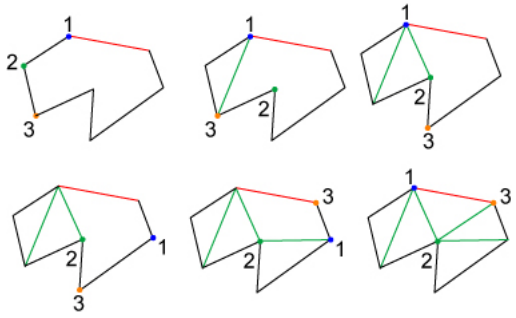
OBR. 5.21. Algoritmus triangulácie hviezdicového mnohouholníka.



OBR. 5.22. Posúvanie vrcholov.



OBR. 5.23. Odrezávanie a posúvanie vrcholov.



OBR. 5.24. Triangulácia algoritmom troch mincí.

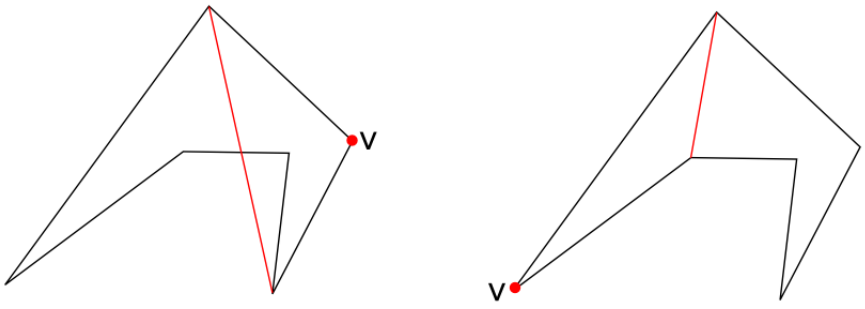
(1) Pre $n = 4$ tvrdenie platí, triangulácia štvoruholníka pozostáva z dvoch neprekrývajúcich sa uší.
(2) Mnohouholník má $n > 4$ vrcholov. Analyzujme nejaký vrchol V_i, pri ktorom je vnútorný uhol konvexný. Môžu nastať dve situácie.
   (a) Vrchol V_i určuje ucho. Odoberme ho a pridajme hranu určenú vrcholmi V_{i-1} a V_{i+1}. Pre mnohouholník s $n - 1$ vrcholmi platí indukčný predpoklad, t.j. má aspoň 2 uši. Dve z nich nemôžu byť určené vrcholmi V_{i-1} a V_{i+1} (bol by to spor - vnútra uší by mali neprázdny prienik). Teda existuje aspoň jedno ucho určené iným vrcholom V_x. Čiže náš pôvodný mnohouholník má aspoň dve neprekrývajúce sa uši určené vrcholmi V_x a V_i.

(b) Vrchol V_i neurčuje ucho. Keďže uhol $V_{i-1}V_iV_{i+1}$ je konvexný, tak existuje aspoň jeden vrchol mnohouholníka vo vnútri trojuholníka $V_{i-1}V_iV_{i+1}$.
Vyberme spomedzi týchto vrcholov jeden, označme ho P. Ak spojnica V_iP nie je uhlopriečka, pretne aspoň jednu hranu mnohouholníka. Vezmime jednu z nich a P označme ten jej koniec, ktorý je vnútri trojuholníka $V_{i-1}V_iV_{i+1}$ (Treba ukázať, že takáto hrana má vždy aspoň jeden koniec v trojuholníku $V_{i-1}V_iV_{i+1}$. Oba konce danej hrany nemôžu byť v opačnej polrovine k $V_{i-1}V_{i+1}V_i$, lebo potom by túto hranu nemohla preťať spojnica V_iP. Čiže aspoň jeden z nich sa musí nachádzať v polrovine $V_{i-1}V_{i+1}V_i$. Keďže V_iP leží celá v trojuholníku $V_{i-1}V_iV_{i+1}$ a hrany jednoduchého mnohouholníka sa nepretínajú, takýto bod musí byť vo vnútri trojuholníka $V_{i-1}V_iV_{i+1}$. Situácia je znázornená na obrázku 5.34). Opakujme test na prienik s hranicou až kým spojnica V_iP nebude uhlopriečka. Algoritmus skončí, pretože skúmame konečnú množinu bodov, pričom zakaždým spomedzi nich vylúčime tie body, ktoré neprešli testom.
Uhlopriečka V_iP rozdelí mnohouholník na dva disjunktné mnohouholníky s menej než n vrcholmi.
Pre tieto dva mnohouholníky platí indukčný predpoklad, t.j. oba majú aspoň dve uši. Dve z nich však nemôžu byť určené vrcholmi V_i a P, pretože ich vnútra by mali neprázdny prienik, čo by bol spor s predpokladom. Teda pre oba mnohouholníky existuje ešte nejaký vrchol V_x (resp. V_y), ktorý určuje ucho. Spolu teda majú minimálne 2 uši. Ak by bol jeden z nich trojuholník, platí to isté, akurát že trojuholník sám by bol jedno ucho "veľkého"mnohouholníka.

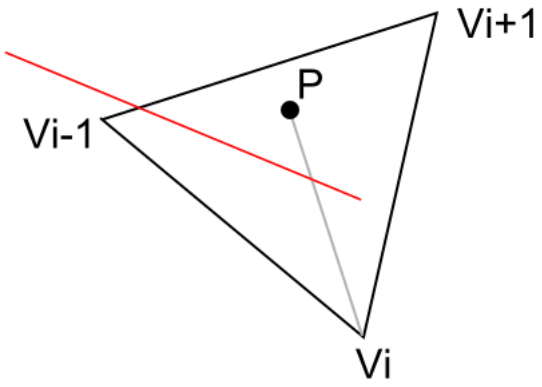
Tvrdenie platí. □

Algoritmus triangulácie a analýza časovej náročnosti:

(1) Nájdime všetky uši mnohouholníka. To znamená, že pre každý vrchol musíme otestovať, či určuje ucho. Tento test pozostáva z testu konvexnosti uhla a prejdenia všetkých lineárnych častí hranice mnohouholnika kvôli testu na prienik s potenciálnym uchom (čo je ekvivalentné prejdeniu všetkých vrcholov). Táto časť algoritmu si teda vyžiada $\mathcal{O}(n^2)$ krokov.
(2) Vezmime jedno ucho mnohouholníka. Toto bude tvoriť prvý trojuholník triangulácie. Odoberme vrchol V_i, ktorý ho určuje a pridajme hranu $V_{i-1}V_{i+1}$. Získame tak jednoduchý mnohouholník s menej vrcholmi (v čase $\mathcal{O}(1)$). Po odobratí vrchola V_i musíme znovu otestovať vrcholy V_{i-1} a V_{i+1}, či sa nezmenil



OBR. 5.25. Vrchol V vľavo neurčuje ucho, vrchol V vpravo určuje.



OBR. 5.26. Aspoň jeden koniec hrany, ktorá má prienik so spojnicou V_iP sa nachádza vo vnútri trojuholníka $V_{i-1}V_iV_{i+1}$.

ich stav (vlastnosť, že určujú ucho, môžu získať, ale aj stratiť). Tento krok vyžaduje $\mathcal{O}(n)$ operácií.

Je dôležité mať na zreteli, že vzhľadom na indukčný predpoklad aj pre nový mnohouholník platí vyššie uvedené tvrdenie, t.j. môžme predpokladať existenciu aspoň jedného ucha v našom udržiavanom zozname. Ak sa jedná o trojuholník, máme trianguláciu hotovú.

Krok 2 vyžaduje spolu $\mathcal{O}(n)$ operácií.

Krok 1 pobeží jedenkrát s náročnosťou $\mathcal{O}(n^2)$, krok 2 pobeží $\mathcal{O}(n)$-krát s náročnosťou $\mathcal{O}(n)$, celková naročnosť algoritmu je teda $\mathcal{O}(n^2)$.

Analýza pamäťovej náročnosti:
V algoritme je okrem vstupnej štruktúry pre mnohouholník veľkosti $\mathcal{O}(n)$ použitá štruktúra pre uchovávanie uší, taktiež veľkosti $\mathcal{O}(n)$. Celková pamäťová náročnosť je teda $\mathcal{O}(n)$. □

Riešenie (Martin Manduch, 6.12.2009): Najprv rozložíme jednoduchý mnohouholník na monotónne podľa y-ovej osi podľa cvičenia 5.8. Potom spravíme triangluláciu monotónnych mnohouholníkov nasledovným algoritmom.

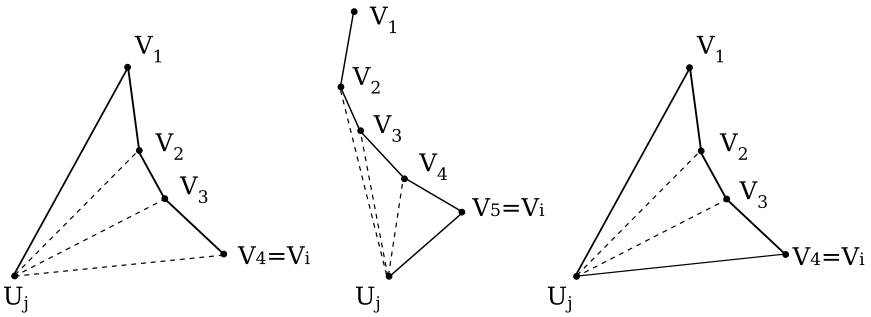
Najprv utriedime vrcholy mnohouholníka M podľa y-ovej súradnice do postupnosti $U_1, \ldots, U_n$. Využijeme štruktúru zásobník na uloženie vrcholov, ktorými sme už prešli, ale ešte k nim možno nájsť vnútornú diagonálu. Ak sa nachádzajú v zásobníku vrcholy $V_1, \ldots, V_i$, tak ich y-ové súradnice tvoria nerastúcu postupnosť. Ak $i \geq 3$, tak uhly $\angle V_j V_{j+1} V_{j+2} \geq \pi$ pre $j = 1, \ldots, i-2$.

```
    TriangulaciaMonotonnehoMnohouholnika()
{
    vlož U_1, U_2 do zásobníka
    for(j = 3; j ≤ n; j++)
    {
        if (U_j je sused V_1, ale nie je sused V_i)      //obr. 5.27 vľavo
        {
            pridaj diagonály U_j V_2, ..., U_j V_i
            //v zásobníku bude iba V_i, U_j
        }
        else if (U_j je sused V_i, ale nie je sused V_1)      //obr. 5.27 v strede
        {
            while(i > 1 a ∠U_j V_i V_{i-1} < π)
                    pridaj diagonálu U_j V_{i-1} a vyber V_i zo zásobníka
            pridaj U_j do zásobníka
        }
        else      //obr. 5.27 vpravo
            pridaj diagonály U_j V_2, ..., U_j V_{i-1}
    }
}
```

Správnosť algoritmu sa ukazuje tak, že pre každú vetvu, **if**, **else if** a **else**, sa dokáže, že pridaná diagonála leží v mnohouholníku a nepretína sa s inými.

Utriedenie vrcholov M podľa y-ovej súradnice trvá $\mathcal{O}(n)$ kvoli monotónnosti M. Každý vrchol je najviac raz pridaný do zásobníka a raz odobraný. Vrchol môže byť sprístupnený v zásobníku pri jeho odoberaní alebo ak na ňom skončí while cyklus. Pri spracovaní nového prvku U_i while cyklus skončí najviac raz. Preto je zložitosť $\mathcal{O}(n)$.

Predpokladajme, že sme mnohouholník M rozložili na $l+1$ monotónnych mnohouholníkov $M_1, \ldots, M_{l+1}$ pridaním l diagonál. Platí: $l < n-3$, lebo ak by nastala rovnosť, mali by sme hotovú triangluláciu M. Nech číslo N označuje súčet počtov bodov všetkých monotónnych mnohouholníkov. Pridaním každej diagonály mnohouholníka M sa toto



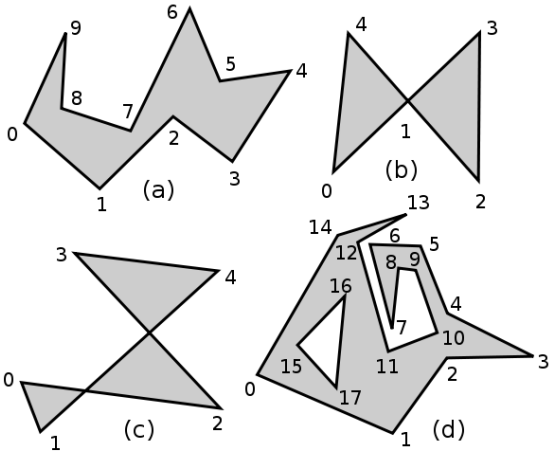
OBR. 5.27. Triangulácia monotónneho mnohouholníka.

číslo zvýši o 2. Diagonál je menej ako $n-3$. Preto $N < (n+2(n-3))$. Triangulácia každého monotónneho trojuholníka má zložitosť $\mathcal{O}(n)$ a spolu majú menej ako $3n-6$ bodov, preto celková zložitosť všetkých je $\mathcal{O}(n)$. Zložitosť rozkladu mnohouholníka na monotónne je $\mathcal{O}(n \log n)$, preto je celkový čas potrebný na trianguláciu jednoduchého mnohouholníka $\mathcal{O}(n \log n)$. □

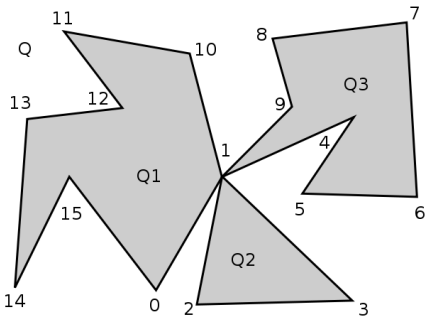
CVIČENIE 5.7 (35 bodov). *Nájdite algoritmus triangulácie mnohouholníka (nie nutne jednoduchého).*

Riešenie (Júlia Kučerová, 12.12.2010): Najprv si zadefinujme pojem mnohouholníka. Mnohouholnk je uzavretý rovinný útvar pozostávajúci z konečného počtu navzájom nadväzujúcich úsečiek. Jednoduchý mnohouholník je taký, v ktorom sa žiadne dve nesusediace hrany nepretínajú. Jednoduché mnohouholníky delia rovinu na dve časti, z nich jedna je ohraničená (vnútro) a druhá neohraničená (vonkajšok mnohouholníka). Jednoduché mnohouholníky môžme rozdeliť na konvexné a nekonvexné. Vnútrom konvexného mnohouholníka je konvexná množina. V jednoduchých mnohouholníkoch patrí vrchol práve dvom susedným hranám. Nejednoduché mnohouholníky môžu obsahovať diery, samoprieseky ako aj vrcholy, ktoré patria viac, ako dvom hranám. Na obrázku obr. 5.28 sú znázornené mnohouholníky: (a) jednoduchý nekonvexný mnohouholník (b) mnohouholník, ktorý obsahuje bod (ozn. 1), ktorý patrí až 4 hranám (c) mnohouholník so samopriesekmi (e) mnohouholník s dierou

Na trianguláciu takýchto mnohouholníkov môžeme použiť algoritmus ear-clipping. V prípade, keď máme na vstupe jednoduchý mnohouholník môžme použiť algoritmus popísaný v úlohe 5.6(riešiteľ Pavol Beluško). Ak máme na vstupe mnohouholník, ktorý obsahuje jeden, alebo viac bodov, ktoré patria viac, ako dvom hranám, ako v príklade (b) môžme pokračovať tak, že si tento mnohouholník rozdelíme na viac mnohouholníkov v týchto bodoch. Toto rozdelenie docielime duplikovaním daného bodu, kde nové body budú mať rovnakú pozíciu, ako



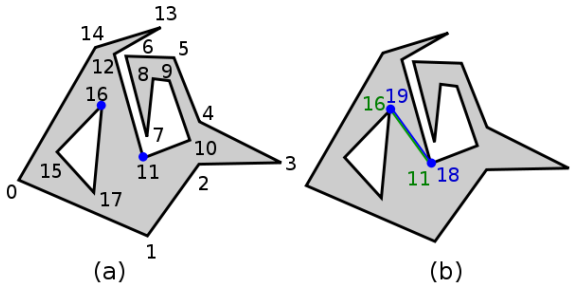
Obr. 5.28. Typy uzavretých lomených čiar



Obr. 5.29. Lomená čiara so samopriesekmi sa dá pridaním vrcholov zmeniť na niekoľko jednoduchých mnohouholníkov.

pôvodný, avšak každý bude patriť inej časti (jednoduchému mnohouholníku) vstupného mnohouholníka. Novovzniknuté mnohouholníky môžme následne triangulovať algoritmom ear-clipping. Na obr. 5.29 môžeme vidieť takýto typ mnohouholníka, kde bod 1 patrí až 6 hranám. Označme ho Q. V tomto bode môžme mnohouholník rozdeliť na 3 menšie. Tri preto, že vrchol patrí 6 hranám, ale ak chceme vytvoriť jednoduché mnohouholníky má patriť práve dvom hranám. Teda nám vznikli 3 mnohouholníky: $Q1 = \{0, 1, 10, 11, 12, 13, 14, 15\}$, $Q2 = \{1, 2, 3\}$ a $Q3 = \{1, 4, 5, 6, 7, 8, 9\}$.

V prípade mnohouholníka s dierami môžme postupovať nasledovne. Uvažujme mnohouholník s jednou dierou. Vrcholy si označme $v_0, \ldots, v_n$. Tento mnohouholník pozostáva z vonkajšieho a vnútorného mnohouholníka. Označme si vrcholy vonkajšieho mnohouholníka v protismere chodu hodinových ručičiek. Vrcholy vnútorného mnohouholníka musia byť označené opačne, teda v smere chodu hodinových ručičiek. Vyberme



Obr. 5.30. Formálna konverzia mnohouholníka s dierou na jednoduchý mnohouholník

si vrchol z vnútorného mnohouholníka a následne vrchol z vonkajšieho. Vrcholy vyberáme tak, aby boli navzájom viditeľné obr. 5.30(a)). Môžme konvertovať topológiu tohto mnohouholníka na topológiu jednoduchého mnohouholníka vložením súhlasných hrán spájajúcich vyznačené vrcholy. Obrázok 5.30(b) ukazuje dve nové hrany, zelenú a modrú. Označené vrcholy musíme zdvojiť kvôli tomu, že dáta vrcholov sú jednoznačné a teda nemôžme použiť rovnaký vrchol na dvoch rôznych miestach mnohouholníka. Zdvojený vrchol má rovnakú pozíciu, ako pôvodný, jeden z nich je konvexný a druhý reflexný. Napríklad spodný vyznačený vrchol má dva asociované vrcholy označené v_{11} a v_{18}. Pôvodný vrchol patril vonkajšiemu mnohouholníku. Po zdvojení a pridaní hrán je v_{11} reflexný a v_{18} konvexný.

Pôvodný mnohouholník bol zložený z vonkajšieho a vnútorného, kde vonkajší bol tvorený vrcholmi

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$$

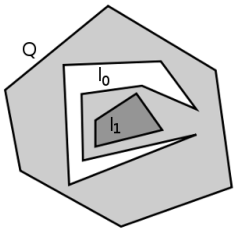
vnútorný bol tvorený vrcholmi

$$\{15, 16, 17\}$$

Po zduplikovaní vrchola v_{11} vznikol vrchol v_{18} a po zduplikovaní vrchola v_{16} vznikol vrchol v_{19}. Mnohouholník po pridaní dvoch nových hrán a vrcholov je tvorený nasledovne

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 16, 17, 15, 19, 18, 12, 13, 14\}$$

Takto vzniknutý mnohouholník je už jednoduchý a teda ho môžeme triangulovať algoritmom ear-clipping. V prípade, že má mnohouholník viacero dier postupujeme nasledovne. Predpokladajme, že vnútorné mnohouholníky ležia vnútri vonkajšieho a nepterínajú sa. Na obr. 5.31 môžete vidieť, že žiaden z vrcholov vnútorného mnohouholníka I_1 nie je viditeľný zo žiadneho vrchola vonkajšieho mnohouholníka Q. Ale niektoré vrcholy mnohouholníka I_0 sú viditeľné z mnohouholníka Q.



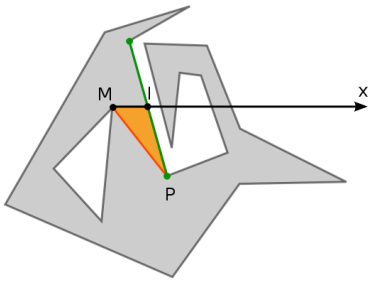
Obr. 5.31. Prekážajúce si vnútorné mnohouholníky

Teda môžme použiť predchádzajúci algoritmus na vytvorenie jednoduchého mnohouholníka (označme si ho Q_1) z mnohouholníkov I_0 a Q. Následne môžeme použiť rovnaký algoritmus na novovzniknutý mnohouholník Q_1 a I_1. Týmto nám vznikne jednoduchý mnohouholník, ktorý už môžeme jednoducho triangulovať. Analogicky budeme pokračovať aj pri väčšom množstve vnútorných mnohouholníkov.

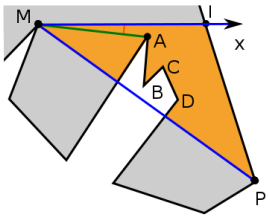
Algoritmus na nájdenie navzájom viditeľných bodov

Na obr. 5.30(a) môžeme vidieť, že vrcholy v_{11} a v_{16} sú navzájom viditeľné. Avšak exisutujú aj ďalšie vrcholy s takouto vlastnosťou. Z vrcholov vnútorného mnohouholníka si vyberieme vrchol s najväčšou x súradnicou, označme si ho M. V prípade, že ich je viac, tak si z nich vyberieme ten s menšou y súradnicou. Veďme z tohto bodu polpriamku rovnobežnú s x-ovou osou kladným smerom. Táto polpriamka sa nám pretne s vonkajším mnohouholníkom minimálne v jednom bode. Zoberieme si bod najbližšie k M. Tento priesečník je buď vrchol mnohouholníka, alebo patrí jeho hrane. V prípade, že je to vrchol vonkajšieho mnohouholníka, našli sme hľadaný bod. V opačnom prípade (ktorý je viac pravdepodobný) pokračujeme nasledovne. Na obr. 5.32 môžeme vidieť bod I, ktorý je priesečníkom polpriamky s vonkajším mnohouholníkom. Hrana, na ktorej leží je zafarbená nazeleno. Koncový bod tejto hrany s väčšou x súradnicou si označme P. Tento bod je možný kandidát na náš hľadaný bod. Trojuhlník (M, I, P) je na obrázku obr. 5.32 vyznačený oranžovou farbou. Celý trojuholník patrí vnútru vonkajšieho mnohouholníka a spojnica bodov M a P nepretína vonkajší mnohouholník, a teda sme našli požadovaný bod. Avšak môže nastať situáca, ktorá je znázornená na obrázku obr. 5.33, kde nám táto spojnica pretne vonkajší mnohouholník. Ako môžeme vidieť 4 vrcholy mnohouholníka ležia v trojuholníku (M, I, P). V takomto prípade najmenej jeden vrchol bude reflexný a zo všetkých reflexných aspoň jeden musí byť viditeľný z vrchola M. V prípade znázornenom na obrázku sú 3 vrcholy reflexné a to A, C a D. Z M je viditeľný vrchol A, pretože uhol medzi spojnicou AM a polpriamkou vedenou z vrchola M je najmenší uhol. Môže nastať situácia, kedy viac bodov zviera rovnaký, a to minimálny, uhol. V tom prípade vyberieme bod najbližšie k M.

Pseudokód



Obr. 5.32. Priesečník polpriamky s vonkajším mnohouholníkom



Obr. 5.33. Prípad prieniku polpriamky a vonkajšieho mnohouholníka, kde MP pretína hranicu vonkajšieho mnohouholníka

```
NÁJDI VIDITEĽNÉ VRCHOLY{
    urči bod M;
    veď bodom polpriamku a urči prienik I;
    IF (I je vrchol){
        viditeľný bod = I;
    }ELSE{
        urči bod P;
        IF (prienik MIP s mnohouholníkom je prázdny){
            viditeľný bod = P;
        }ELSE{
            urči reflexné body patriace MIP;
            urči bod A ktorého spojnica s M má minimálny uhol s polpriamkou;
            viditeľný bod = A;
        }
    }
}
```

(Zdroj: http://www.geometrictools.com/Documentation/TriangulationByEarClipping.pdf)

□

Cvičenie 5.8 (25 bodov). *Nájdite algoritmus rozkladu mnohouholníka na monotónne mnohouholníky vzhľadom na daný smer.*

Riešenie (Jana Hlinková, 4.12.2008):

Popíšem $\mathcal{O}(n \log n)$ algoritmus rozkladu mnohouholníka na monotónne mnohouholníky (vzhľadom na x-ovú os) založený na zametacom princípe. Rozklad na monotónne mnohouholníky sa dosiahne pridaním vhodne zvolených nepretínajúcich sa diagonál.

Mnohouholník M je monotónny vzhľadom na x-ovú os, ak prienikom M s ľubovoľnou vertikálnou priamkou je jeden súvislý komponent. Monotónnosť mnohouholníku porušuje taký vrchol, pri ktorom je vnútorný uhol väčší ako 180° a obe hrany s ním incidentné ležia buď naľavo, alebo obe napravo od neho. Vrchol, ktorého obe hrany ležia naľavo od neho, budeme označovať ako spájajúci vrchol (vzhľadom na prechod zametacej priamky v smere osi x sa v ňom hrany spájajú). Vrchol, ktorého hrany ležia napravo nazveme rozdeľovací vrchol.

Pozrime sa na prípad rozdeľovacieho vrcholu v. Takýto vrchol potrebujeme spojiť diagonálou s vhodným vrcholom naľavo od neho alebo nad ním (teda s vrcholom, ktorého x-ová súradnica je menšia nanajvýš rovná x-ovej súradnici rozdeľovacieho vrcholu). Takáto diagonála rozdelí mnohouholník na dva mnohouholníky, v ktorých už daný rozdeľovací vrchol nebude rozdeľovací (hrany s ním incidentné už nebudú obe ležať napravo od neho).

Ako nájsť vhodný vrchol? Vhodný vrchol musí byť samozrejme viditeľný z rozdeľovacieho vrcholu, aby sme ich mohli spojiť diagonálou. Vhodný vrchol pre rozdeľujúci vrchol definujeme pomocou hrany e_{nad} mnohouholníka, ktorá je bezprostredne nad rozdeľovacím vrcholom a hrany e_{pod}, ktorá je beprostredne pod rozdeľovacím vrcholom (keďže ide o rozdeľovací vrchol, takého hrany existujú). Za vhodný vrchol budeme považovať najpravejší vrchol, ktorý je vertikálne viditeľný z oboch hrán e_{nad} a e_{pod} (vertikálne viditeľný znamená, že je viditeľný pre každý bod pozdĺž vertikálnej zametacej priamky medzi e_{nad} a e_{pod}, prechádzajúcej týmto vrcholom). Vhodný vrchol prislúchajúci hrane e_{nad} budem označovať $v(e)$.

Zametací algoritmus:

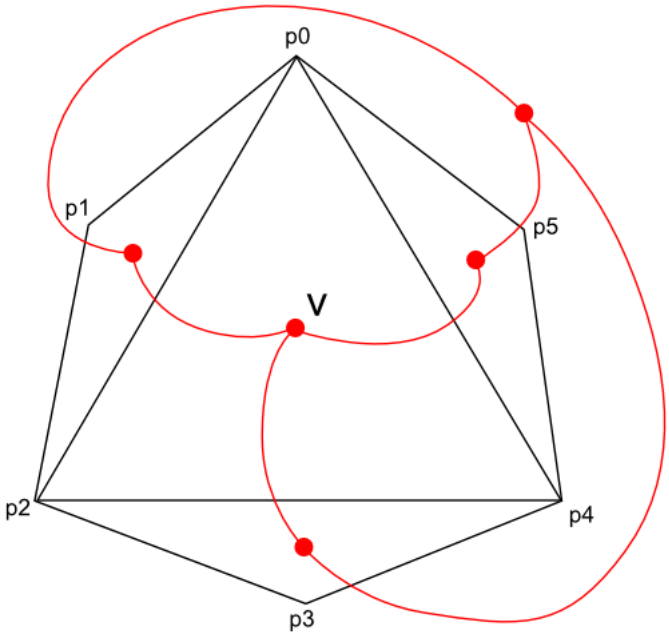
- Vertikálna zametacia priamka sa bude pohybovať od najľavejšieho vrcholu mnohouholníka k najpravejšiemu.
- *Udalosti:* koncové body hrán mnohouholníka (teda vrcholy mnohouholníka), ktoré sú utriedené podľa x-ovej súradnice. V priebehu algoritmu sa negenerujú žiadne nové udalosti.
- *Stav zametacej priamky:* zoznam hrán, ktoré pretínajú zametaciu priamku, utriedený odhora nadol (pre jednoduché prehľadávanie môže byť uložený vo vyváženom binárnom strome). Spolu s hranou si pamätáme jej vhodný vrchol $v(e)$.

- *Spracovanie udalosti:* spracovanie závisí od typu udalosti (typu vrcholu):
  - *rozdeľovací vrchol:* prehľadaním zoznamu prislúchajúcemu súčastnému stavu zametacej priamky nájdeme hranu e_{nad} ležiacu bezprostredne nad týmto rozdeľovacím vrcholom v. v spojíme s vhodným vrcholom $v(e)$ prislúchajúcim hrane e_{nad}. Pôvodné hrany incidentné s v pridáme do zoznamu prislúchajúcemu stavu zametacej priamky, a v zvolíme za vhodný vrchol prislúchajúci spodnej hrane a za nový vhodný vrchol hrany e_{nad}.
  - *spájajúci vrchol:* vymažeme zo zoznamu stavu zametacej priamky hrany incidentné s týmto vrcholom, nájdeme hranu e_{nad} a zvolíme v za vhodný vrchol tejto hrany. item*najľavejší (začiatočný) vrchol:* hrany incidentné s týmto vrcholom pridíme do zoznamu stavu zametacej priamky a vrchol zvolíme za vhodný vrchol hornej hrany.
  - *najpravejší (koncový) vrchol:* vymaž incidentné hrany zo zoznamu stavu zametacej priamky. Koniec algoritmu.
  - *vrchol horného reťazca (jedna incidentná hrana je od neho naľavo, druhá napravo a vnútro mnohouholníka je pod ním):* v zozname stavu zametacej priamky zameníme ľavú hranu za pravú a vhodný vrchol ľavej zvolíme za vhodný vrchol pravej.
  - *vrchol dolného reťazca (jedna incidentná hrana je od neho naľavo, druhá napravo a vnútro mnohouholníka je nad ním):* v zozname stavu zametacej priamky zameníme ľavú hranu za pravú. Nájdeme hranu ležiacu bezprostredne nad spracovávaným vrcholom a zvolíme ho za jej vhodný vrchol.

  V posledných troch prípadoch overíme, či pôvodným vhodným vrcholom bol spájajúci vrchol, ak hej, tieto vrcholy spojíme diagonálou.

Zložitosť spracovania udalosti je $\mathcal{O}(\log n)$ (prehľadávanie vo vyváženom vyhľadávacom strome a k tomu konštantne veľa operácií). Udalostí je n (jedna pre každý vrchol mnohouholníka), a teda celková zložitosť je $\mathcal{O}(n \log n)$.

Rozklad mnohouholníka na monotónne mnohouholníky vzhľadom na x-ovú os sa môže využiť pri rozklade na monotónne mnohouholníky vzhľadom na ľubovoľný smer. Postup je nasledovný: najprv mnohouholník zrotujeme tak, aby zadaný smer bol totožný so smerom x-ovej osi, následne prevedieme rozklad na monotónne mnohouholníky vzhľadom na x-ovú os, na záver mnohouholník zrotujeme do pôvodnej polohy. □



OBR. 5.34. Triangulácia monotónneho mnohouholníka a jej duálny graf

Riešenie (Erika Pálešová, 5.12.2009): Majme v rovine mnohouholník, ktorý chceme rozložiť na monotónne mnohouholníky vzhľadom na daný smer.

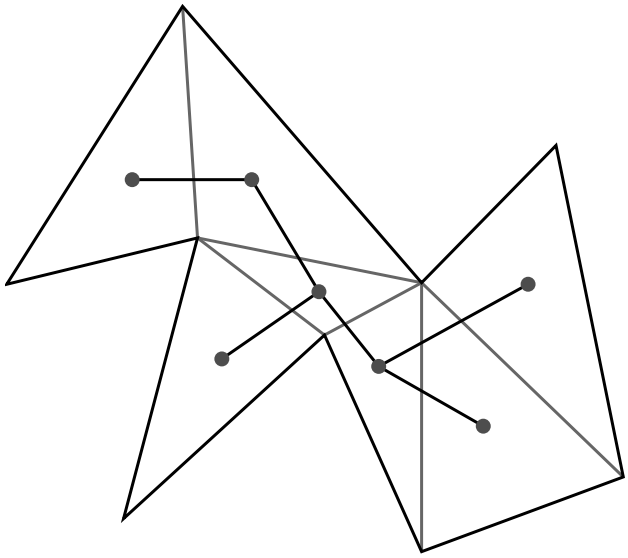
Uvažujme nad trojuholníkom. Každý typ trojuholníka považujeme za monotónny mnohouholník vzhľadom na každý možný smer.

Preto pre rozklad mnohouholníka na monotónne mnohouholníky vzhľadom na daný smer nám stačí tento mnohouholník triangulovať (napr. spôsobmi z úloh 5.5, 5.6, 5.7). Týmto spôsobom ho budeme mať rozdelený na monotónne mnohouholníky vzhľadom na každý smer, a teda aj na daný smer. □

CVIČENIE 5.9 (20 bodov). *Dokážte alebo vyvráťte tvrdenie: Duálny graf triangulácie monotónneho mnohouholníka je reťazec (t.j. graf, ktorého každý vrchol má stupeň nanajvýš 2).*

Riešenie (Pavol Beluško, 17.11.2008): Tvrdenie vo všeobecnosti neplatí. Na obrázku 5.34 je kontrapríklad: monotónny mnohouholník (v ľubovoľnom smere) a jeho triangulácia. Vrchol v jej duálneho grafu má stupeň 3. □

CVIČENIE 5.10 (25 bodov). *Opíšte algoritmus, ktorý nájde pre daný jednoduchý mnohouholník M uhlopriečku, ktorá ho rozdelí na mnohouholníky M_1 a M_2 s nanajvýš $\lfloor \frac{2n}{3} \rfloor + 2$ vrcholmi.*

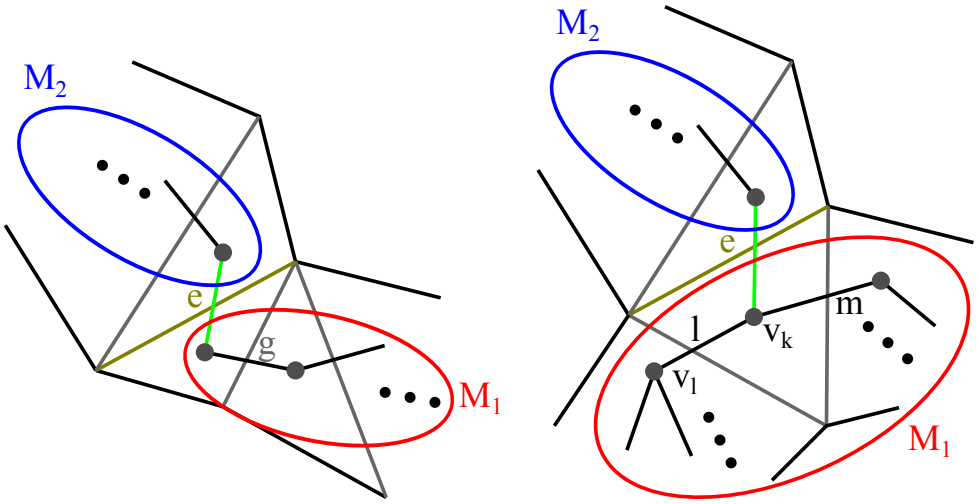


Obr. 5.35. Jednoduchý mnohouholník s 9 vrcholmi, jeho triangulácia a duálny graf tejto triangulácie.

Riešenie (Michal Ferko, 20.11.2011): Algoritmus najprv vytvorí trianguláciu mnohouholníka M pridaním $n-3$ uhlopriečok, a teda rozdelí mnohouholník na $n-2$ trojuholníkov ako na obr. 5.35. Následne sa skonštruuje duálny graf ku triangulácii - vrcholy grafu budú trojuholníky v triangulácii a hrana bude medzi každými dvoma trojuholníkmi, ktoré majú spoločnú hranu. To znamená, že v tomto grafe je každý vrchol stupňa nanajvýš 3. Keďže triangulácia iba pridáva uhlopriečky a nie body, tak každý vrchol mnohouholníka je na jeho hranici. Teda každá hrana duálneho grafu zodpovedá uhlopriečke - je to hrana spájajúca dva vrcholy na hranici a je to hrana patriaca do vnútra mnohouholníka (lebo je obsiahnutá v dvoch trojuholníkoch, teda nemôže byť na hranici). Navyše duálny graf je strom, keďže všetky vrcholy sa nachádzajú na hranici (a teda nemôže vzniknúť cyklus) a ide nám o jednoduché mnohouholníky, teda nemá diery.

Pomocou duálneho grafu G teraz nájdeme takú uhlopriečku, ktorá vyhovuje požiadavkám zo zadania. Hľadáme uhlopriečku e, teda hranu duálneho grafu, pričom chceme, aby z hľadiska počtu vrcholov ani jeden z mnohouholníkov nebol extrémny (a nepresiahol počtom vrcholov $\lfloor \frac{2n}{3} \rfloor + 2$, čiže počtom trojuholníkov $\lfloor \frac{2n}{3} \rfloor$). Ak by sa nám podarilo nájsť hranu, ktorá rozdelí mnohouholník približne na polovicu (čo sa počtu vrcholov týka), tak máme vyhrané.

Hranu e nájdeme nasledovným postupom. Pre každú hranu grafu G zistíme, koľko vrcholov je v podstrome na jednej strane hrany a koľko je vrcholov na druhej strane hrany. Keď máme túto informáciu pre všetky hrany (ako dvojicu hodnôt (i_e, j_e) pre hranu e), vyberieme



(A) Prípad 1 (B) Prípad 2

OBR. 5.36. Možnosti umiestnenia hrany e v grafe G

z nich tú hranu, pre ktorú je hodnota funkcie $V(e) = |i_e - j_e|$ najmenšia zo všetkých. Takto dostaneme v ideálnom prípade uhlopriečku, ktorá rozdeľuje mnohouholník na 2 mnohouholníky s rovnakým počtom trojuholníkov. A teda tieto mnohouholníky budú mať približne $\frac{n}{2} + 2$ vrcholov.

Teraz treba ukázať, že e nemôže rozdeliť M tak, že by jeden z mnohouholníkov M_1, M_2 mal viac ako $\lfloor \frac{2n}{3} \rfloor + 2$ vrcholov. Postupujeme sporom: Nech M_1 po vybratí hrany má viac ako $\lfloor \frac{2n}{3} \rfloor + 2$ vrcholov, t.j. viac ako $\lfloor \frac{2n}{3} \rfloor$ trojuholníkov. To znamená, že M_2 má menej ako $\lfloor \frac{n}{3} \rfloor + 1$ trojuholníkov. Trojuholník T z M_1, ktorý obsahuje hranu e môže byť dvoch typov: buď jedna z jeho zvyšných hrán je na hranici M_2 alebo obe jeho zvyšné hrany sú uhlopriečky M. Oba tieto prípady sú znázornené na obrázku 5.36. Prvá možnosť nastať nemôže, lebo hrana, ktorá je uhlopriečkou M (označme ju g) by mala $V(g) < V(e)$.

Zostáva ukázať, že ak nastane druhý prípad - hrany T sú 2 uhlopriečky M (označené ako l a m v obrázku 5.36) a hrana e, tak jedna z tých uhlopriečok mala nižšiu hodnotu $|i - j|$ ako e. Ak vezmeme graf (strom) triangulácie prislúchajúci M_1 - označme ho G_1, vieme, že má viac ako $\lfloor \frac{2n}{3} \rfloor$ vrcholov. Koreň v_k stromu G_1 je stupňa dva a to znamená, že aspoň jeden z podstromov určených jedným zo synov v_k má viac ako $\lfloor \frac{n}{3} \rfloor$ vrcholov - označme tento vrchol v_l. Nech hrana l je určená v_k a v_l. Vyjadrime hodnoty $V(e)$ a $V(l)$:

$$V(e) = |i_e - j_e| = \left| \lfloor \frac{2n}{3} \rfloor + a - (\lfloor \frac{n}{3} \rfloor + 1 - a) \right| = \lfloor \frac{n}{3} \rfloor + 2a - 1$$

$$V(l) = |i_l - j_l| = \left| (n - (\lfloor \frac{n}{3} \rfloor + b)) - (\lfloor \frac{n}{3} \rfloor + b) \right| = \left| \lfloor \frac{n}{3} \rfloor - 2b \right|$$

Kde $a, b \in \{1, 2, ...\}$. Vidno, že $V(e) > V(l)$, dostávame sa k sporu zvolenia e, lebo l by bol lepší kandidát. Preto M_1 ani M_2 nemôžu mať viac ako $\lfloor \frac{2n}{3} \rfloor + 2$ vrcholov.

Jednotlivé kroky algoritmu sú znázornené v obrázku 5.37.

Dôležitá otázka je, ako rýchlo bude náš algoritmus bežať. Najprv treba vytvoriť trianguláciu jednoduchého mnohouholníka. Tento krok možno zrealizovať v čase $\mathcal{O}(n)$, ako ukázal Bernard Chazelle v *Bernard Chazelle. 1991. Triangulating a simple polygon in linear time.* Discrete Comput. Geom. *6, 5 (August 1991), 485-524.* Tento algoritmus je ale extrémne zložitý. V knihe *Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. 2008.* Computational Geometry: Algorithms and Applications (3rd ed.). *TELOS, Santa Clara, CA, USA.* popisujú autori jednoduchší algoritmus v čase $\mathcal{O}(n \log n)$.

Vytvorenie duálneho grafu je jednoduchý prechod trojuholníkovou štruktúrou, a ak sme vytvorili rozumnú dátovú štruktúru počas prvej časti algoritmu, tak tento krok potrvá $\mathcal{O}(n)$.

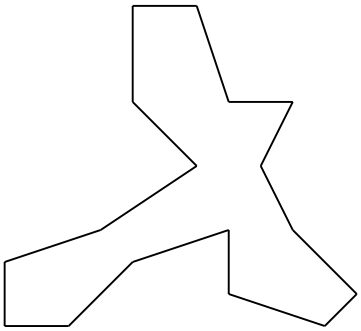
Určenie koeficientov (a následne váh) pre hrany by sa dalo priamo: každú hranu odoberieme a zrátame počet vrcholov na jednej a na druhej strane hrany. Hrán je ale $\mathcal{O}(n)$ a vrcholov tiež, preto takýto algoritmus by trval $\mathcal{O}(n^2)$. Dá sa to spraviť šikovnejšie. Určíme jeden z vrcholov grafu za koreň (na jeho výbere nezáleží, môže to byť hociktorý) a z neho rekurzívne prebehneme, aby sme zistili, koľko prvkov má každý podstrom. Dá sa to zapísať nasledujúcim pseudokódom.

```
PocetVrcholovPodstromu(Vrchol v)
{
    pocetVrcholov = 1
    pre každého syna vrchola v, označme ho w
        pocetVrcholov = pocetVrcholov + PocetVrcholovPodstromu(w)
    ulož pocetVrcholov do vrchola v
    vráť pocetVrcholov
}
```

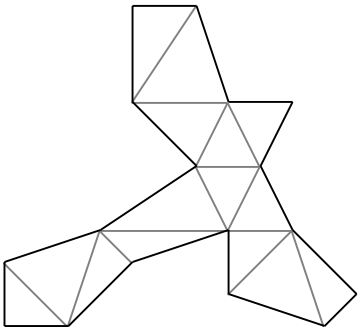
Informáciu o počte vrcholov každého podstromu si uložíme pre každý vrchol. Teda v každom vrchole bude uložená informácia, koľko vrcholov má podstrom určený týmto vrcholom. Následne nám stačí prebehnúť zoznam hrán (tentoraz sú už orientované, všetky v strome ukazujú zhora nadol) a pre každú hranu vziať ako prvý koeficient k počet vrcholov v podstrome uložený pre vrchol, do ktorého hrana ukazuje. Druhý koeficient sa zráta jednoducho, je to $n - 2 - k$, lebo počet všetkých vrcholov v grafe je $n - 2$.

Táto časť algoritmu beží v čase $\mathcal{O}(n)$, lebo rekurzívna časť sa zavolá $\mathcal{O}(n)$ krát a jeden prechod všetkými hranami je takisto $\mathcal{O}(n)$.

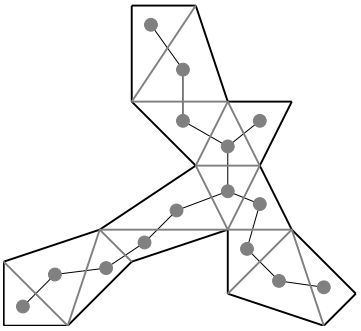
Zvyšok celkového algoritmu je už jednoduchý a beží v čase $\mathcal{O}(n)$. Celkovo by sme teda vedeli zložitým algoritmom na trianguláciu dosiahnuť zložitosť algoritmu $\mathcal{O}(n)$, ale jednoduchším algoritmom dosiahneme $\mathcal{O}(n \log n)$, čo je akceptovateľné. □



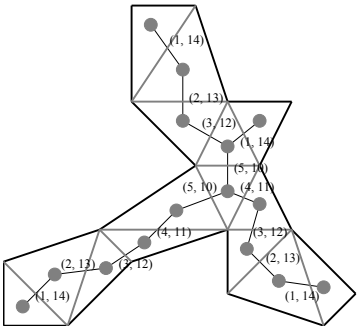
(A) Krok 1 - vstupné dáta



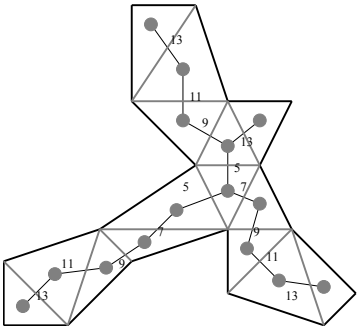
(B) Krok 2 - triangulácia



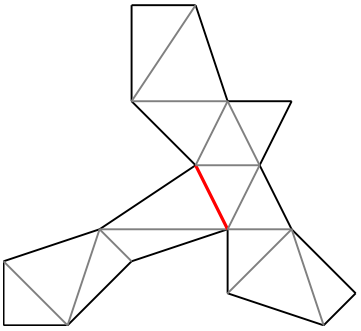
(C) Krok 3 - duálny graf



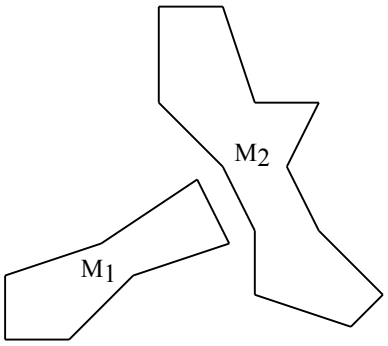
(D) Krok 4 - koeficienty pre hrany



(E) Krok 5 - ováhované hrany



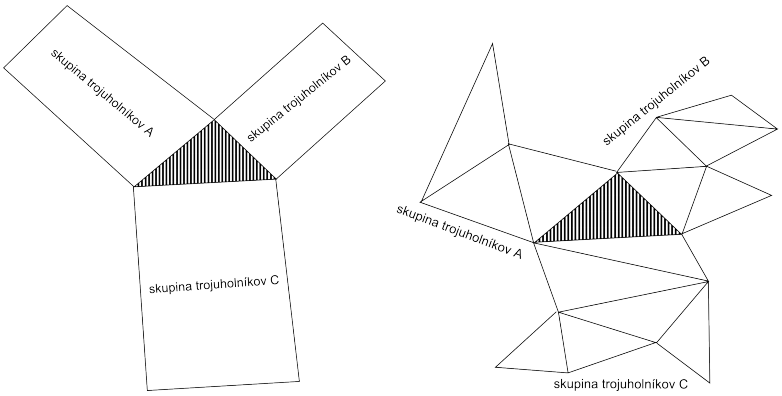
(F) Krok 6 - vybraná hrana s najmenšou váhou



(G) Krok 7 - výstup

OBR. 5.37. Príklad mnohouholníku a aplikácia jednotlivých krokov algoritmu

Riešenie (Jakub Mišún, 01.12.2011): Najprv musíme dokázať, že takáto uhlopriečka vôbec existuje. K tomu prevedieme náš problém na ekvivalentný. Vieme, že každý n-uholník môžeme triangulovať na $n-2$ trojuholníkov. Preto hľadáme takú uhlopriečku mnohouholníka M, ktorá ho rozdelí na mnohouholníky M_1 a M_2 s nanajvýš $\lfloor \frac{2n}{3} \rfloor$ trojuholníkmi, respektíve na mnohouholník s najmenej $\lceil \frac{n}{3} \rceil - 2$ trojuholníkmi. Teraz si vezmime ľubovolný trojuholník triangulácie mnohouholníka M (budeme ho nazývať trojuholník stredový). Triangulácia je tvorená uhlopriečkami mnohouholníka a teda každá hrana nášho stredového trojuholníka je buď uhlopriečkou alebo hranou mnohouholníka M. Preto je jasné, že na každú jeho hranu je napojená skupina trojuholníkov (môže byť aj prázdna), pričom trojuholníky medzi skupinami sa nijako nedotýkajú. Ak je hrana trojuholníka uhlopriečkou M, tak je napojená skupina trojuholníkov neprázdna a ak je hrana trojuholníka hranou M, tak je nutne prázdna. Dá sa to graficky znázorniť nasledujúco.

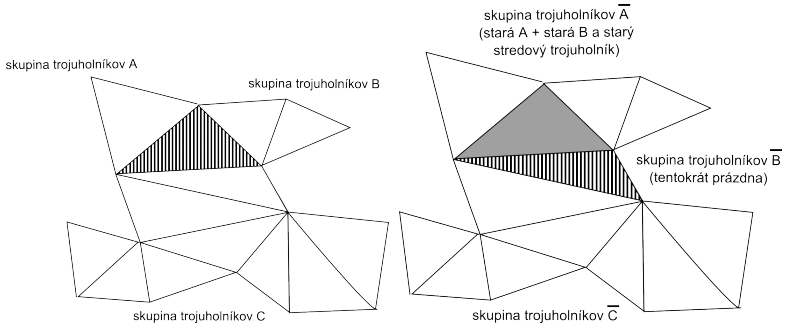


Obr. 5.38. Grafické znázornenie mnohouholníka spolu s reálnym tvarom

Samozrejme, treba si uvedomiť, že graf v ľavej časti obrázka nezodpovedá skutočnému usporiadaniu trojuholníkov v jednotlivých skupinách. To však nie je dôležité, lebo odteraz nás bude zaujímať len počet trojuholníkov v jednotlivých skupinách, pričom pamätajme, že až dve z nich môžu byť prázdne. Teraz sa pozrieme na jednotlivé skupiny. V každej skupine overíme či náhodou nespĺňa požadovaný počet trojuholníkov. Nájdeme si skupinu s najväčším počtom trojuholníkov. Nech je to napríklad C. Ak má potrebný počet trojuholníkov, našli sme riešenie, lebo hľadaná uhlopriečka je hrana stredového trojuholníka, na ktorú sa napája C. Čo ak ale požadovaný počet trojuholníkov nemá? Menej ako $\lceil \frac{n}{3} \rceil - 2$ mať nemôže, lebo C ich má najviac spomedzi všetkých troch skupín a teda v skupine A aj B ich je nutne menej ako $\lceil \frac{n}{3} \rceil - 2$ a teda dohromady by všetkých trojuholníkov triangulácie bolo menej ako $n-2$. Nutne ich teda má viac ako $\lfloor \frac{2n}{3} \rfloor$. Z toho vyplýva, že

aj keby sme spojili skupiny A a B a stredový trojuholník, tak táto novo vytvorená skupina má trojuholníkov menej ako $\lceil \frac{n}{3} \rceil - 2$, pretože skupina C nám ich nechala menej než $n-2-\lfloor \frac{2n}{3} \rfloor - 1 = n-3-\lfloor \frac{2n}{3} \rfloor \leq \lceil \frac{n}{3} \rceil - 2$ a teda tiež nie je dobrá.

Nasledujúcim krokom teda bude zmenšenie skupiny C. Urobíme to tak, že si zvolíme nový stredový trojuholník. Novým stredovým trojuholníkom vyhlásime trojuholník zo skupiny C susediaci so súčasným stredovým trojuholníkom. Dostávame nový graf, v ktorom sa nám do jednej skupiny spojili A a B a starý stredový trojuholník (označme ju $\bar{A}$) a skupina C sa nám rozdelila do dvoch (označme ich $\bar{B}$ a $\bar{C}$), pričom jedna z nich môže byť prázdna. Treba si uvedomiť, že aj $\bar{B}$ aj $\bar{C}$ majú nutne minimálne o jeden trojuholník menej ako C a to aj v prípade, že jedna z nich je prázdna.



Obr. 5.39. Príklad iterácie

Opäť nájdeme tú najväčšiu skupinu. Treba si uvedomiť, že ňou nemôže byť $\bar{A}$, lebo ako sme ukázali má menej ako $\lceil \frac{n}{3} \rceil - 2$ trojuholníkov. Vďaka tomu sa nemôže stať, že sa v iteratívnom hľadaní začneme vracať späť (myslí sa tým, že stredovým trojuholníkom sa stane taký, ktorý ním už raz bol). Najväčšia skupina je teda $\bar{B}$ alebo $\bar{C}$. Nech je to napríklad $\bar{C}$. Ak má požadovaný počet trojuholníkov, skončili sme. Čo ak nemá? Opäť ich nemôže mať menej ako $\lceil \frac{n}{3} \rceil - 2$. Ak ich má viac opakujeme opäť celý postup s posunom stredového trojuholníka. Takto pokračujeme až nenájdeme skupinu s patričným počtom trojuholníkov. Posledná vec, ktorú treba ukázať je, že takáto iterácia vôbec niekedy skončí. Je jasné, že takýmto postupom znižujeme počet trojuholníkov v najväčšej skupine. Teda ich počet raz musí klesnúť pod $\lfloor \frac{2n}{3} \rfloor$. Problém by ale nastal, ak by zároveň hneď klesol aj pod $\lceil \frac{n}{3} \rceil - 2$. Našťastie to nie je možné. A to z toho istého dôvodu, ktorý sme už spomínali. Množina $\bar{A}$ má trojuholníkov menej ako $\lceil \frac{n}{3} \rceil - 2$. Ak by ich mala menej aj väčšia z množín $\bar{B}$ a $\bar{C}$, tak ich má nutne menej aj posledná skupina a dohromady by sme mali v mnohouholníku nedostatok trojuholníkov, lebo $\lceil \frac{n}{3} \rceil - 3 + \lceil \frac{n}{3} \rceil - 3 + \lceil \frac{n}{3} \rceil - 3 + 1 = n - 5$, čo je SPOR.

Vidíme, že dôkaz je konštruktívny, a teda sa dá použiť ako postup pri hľadaní vhodnej uhlopriečky.

Poslednou vecou, ktorú treba ukázať je zložitosť takéhoto algoritmu.

ZLOŽITOSŤ:

(1) Triangulácia mnohouholníka M ($O(n \log n)$)
(2) Voľba stredového trojuholníka ($O(1)$)
(3) Zrátanie trojuholníkov v skupinách ($O(n)$)
(4) Nájdenie najväčšej skupiny ($O(3)$)
(5) Overenie či najväčšia skupina spĺňa podmienku ($O(1)$)
(6) Ak ju nespĺňa vraciame sa ku kroku 2, ten sa opakuje najviac $\lceil \frac{n}{3} \rceil - 2$ krát.
(7) Ak ju spĺňa našli sme uhlopriečku.

Celý proces od zvolenia stredového trojuholníka až po overenie podmienky je lineárny. Tento proces sa môže opakovať až $\lceil \frac{n}{3} \rceil - 2$ krát, a teda celková zložitosť je ($O(n(\lceil \frac{n}{3} \rceil - 2))$).

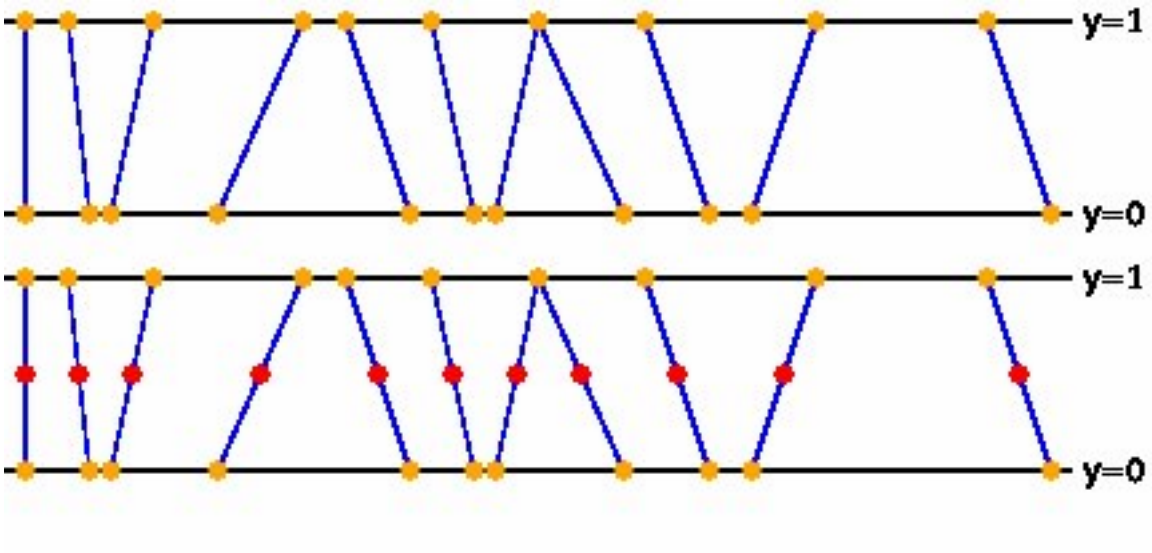
□

Cvičenie 5.11 (30 bodov). *Nech S je množina n úsečiek v euklidovskej rovine s jedným koncovými bodom na priamke $y = 1$ a druhým koncovým bodom na priamke $y = 0$. Tieto úsečky rozdeľujú pás $[-\infty, \infty] \times [0, 1]$ na $n + 1$ oblastí. Opíšte algoritmus vybudovania binárneho vyhľadávacieho stromu na týchto úsečkách z S tak, že oblasť obsahujúca daný bod sa dá určiť v čase $\mathcal{O}(\log n)$.*

Riešenie (Marian Maričák, 13.11.2008):

Keďže úsečky rozdeľujú pás na $n+1$ oblastí, vieme, že ak sa pretínajú, tak je to maximálne v jednom bode a tento bod leží na priamke $y = 0$ alebo $y = 1$. Aby sme sa vyhli nejednoznačnej reprezentácii úsečiek, tak pre účely vybudovania binárneho stromu si každú úsečku reprezentujeme jej stredným bodom (na obrázku sú červenou farbou). Ak sú krajné body úsečky A a B, tak jej stredný bod bude $(A+B)/2$. Následne úsečky utriedim podľa x-ovej súradnice ich stredných bodov. Teraz môžem spustiť algoritmus na tvorbu binárneho stromu:

```
VybudujBinarnyStrom(𝕊)
{
if (𝕊 == 0) return NULL;
v = new BinarnyUzol;
v->usecka = Median(𝕊);
L = UseckyNalavo(v->usecka, 𝕊);
R = UseckyVpravo(v->usecka, 𝕊);
v->lavy = VybudujBinarnyStrom(𝕃);
v->pravy = VybudujBinarnyStrom(ℝ);
}
```



Obr. 5.40. hore sú pôvodné úsečky, dole - úsečky a k nim prislúchajúce stredné body (červenou)

Vo vrchole binárneho stromu sú uložené tri záznamy. Smerníky na ľavý a pravý podpriestor a všeobecná rovnica priamky, ktorú daný vrchol reprezentuje. Úsečku teda popisuje rovnica priamky na ktorej táto úsečky leží. Keďže množinu úsečiek delím vždy pomocou mediánu, výsledná výška stromu bude $\mathcal{O}(\log n)$. Keď príde požiadavka na bod, prechádzam stromom od koreňa k listom, pričom vždy dosadím súradnice bodu do rovnice priamky v danom vrchole a podľa výsledku pokračujem buď vpravo alebo vľavo, prípadne vyhlásim, že bod leží na úsečke. Rozhodnúť sa ako pokračovať ďalej vieme v čase $\mathcal{O}(1)$, výška stromu je $\mathcal{O}(\log n)$, a teda oblasť obsahujúcu daný bod vieme určiť v čase $\mathcal{O}(\log n)$.

□

Cvičenie 5.12 (30 bodov). *Nech S je teselácia roviny zložitosti n a nech P je množina obsahujúca m bodov. Navrhnite zametací algoritmus, ktorý pre každý bod z P nájde región z S, do ktorého tento bod patrí. Pokúste sa nájsť algoritmus s časovou zložitosťou $\mathcal{O}((n + m)\log(n + m))$.*

Cvičenie 5.13 (40 bodov). *Máme n rovnobežných železničných tratí s n vlakmi, ktoré jazdia konštantnými rýchlosťami $v_1, \ldots, v_n$. V čase $t = 0$ sú vlaky v pozíciách $k_1, \ldots, k_n$. Navrhnite dátovú štruktúru na nasledujúcu úlohu: Ktorý vlak je v danom čase $t > 0$ na prvom mieste na trati? Analyzujte čas predspracovania, pamäťovú zložitosť a čas získania výsledku.*

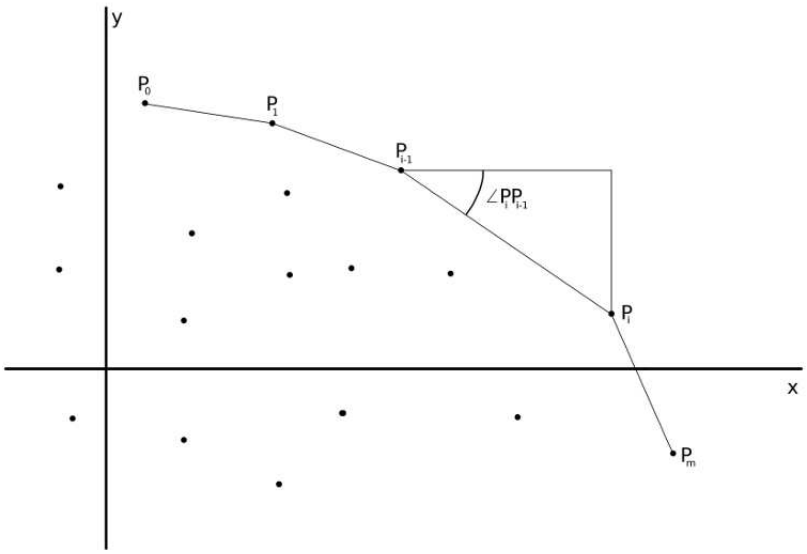
Riešenie (Martin Škorupa, 16.12.2008): Nech každý vlak predstavuje bod $P_i \in \mathbb{E}^2$, pričom x-ovú súradnicu predstavuje rýchlosť vlaku v_i a y-ovú predstavuje začiatočná pozícia k_i. Ak $P_i = P_j$, tak aj vlaky sú totožné. Nech postupnosť $v_0,\ldots, v_m$ je postupnosť vlakov ktoré boli niekedy prvé a postupnosť $t_0,\ldots,t_m$ sú časy kedy boli tieto vlaky prvé. Nech $t(v_i, v_j)$ je čas stretu vlakov v_i a v_j.

$t(v_i, v_j) = (v_j.k$ - $v_i.k)$ / $(v_i.v$ - $v_j.v)$, uvažujeme len vlaky ktoré sa stretnú.
Nech $\angle\ P_iP_j$ je uhol zvierajúci vektor P_iP_j s x-ovou osou (vektorom [0,1])
$\angle\ P_iP_j = \arctan |P_j.y$ - $P_i.y|$ / $|P_i.x$ - $P_j.x|$, uvažujeme len body, ktoré neležia na priamke rovnobežnej s osou x.

- Nájdeme konvexný obal bodov P_i. $\mathcal{O}(n \log n)$
- Nájdeme postupnosť bodov $P_0, \ldots, P_m$ konvexného obalu od bodu s maximálnou y-ovou súradnicou po bod s maximálnou x-ovou súradnicou (v smere chodu hodinových ručičiek). $\mathcal{O}(n)$
- Túto postupnosť vložíme do binárneho stromu, pričom k bodom pridáme ešte jednu súradnicu (čas) podla ktorej triedime v binárnom strome: $P_0.t = 0$, $P_i.t = (P_{i-1}.y$ - $P_i.y)$ / $(P_i.x$ - $P_{i-1}.x)$. $\mathcal{O}(n)$
- V binárnom strome máme uložené vlaky ktoré boli niekedy prvé na trati a to práve od času t. Vyhľadávanie v tomto strome trvá $\mathcal{O}(\log n)$.

Dôkaz: Ukážeme, že postupnosť $P_0, \ldots, P_m$ je totožná s $v_0, \ldots, v_m$. Pre $v_0, \ldots, v_m$ platí:

- v_0 je vlak s najväčším k, teda s vlakom ktorý bol v čase t = 0 najďalej. P_0 je bod s maximálnou y-ovou súradnicou, teda vlak s najväčším k.
- v_m je vlak s najväčším v, teda s najrýchlejší vlak. P_m je bod s maximálnou x-ovou súradnicou (pri rovnosti x-ovej berieme bod s väčšou y-ovou), teda vlak s najväčším v.
- Pre vlaky v_{i-1}, v_i platí:
  - v_{i-1} bol pre t = 0 ďalej ako v_i: $v_{i-1}.k > v_i.k$. Pre našu časť konvexného obalu platí $P_{i-1}.y > P_i.y$.
  - v_i je rýchlejší ako v_{i-1}: $v_{i-1}.v < v_i.v$. Pre našu časť konvexného obalu platí $P_{i-1}.x < P_i.x$.
  - v_i je prvý vlak, ktorý predbehne v_{i-1}. Teda $t(v_{i-1}, v_i)$ je najmenšie s pomedzi všetkých $t(v_{i-1}, v_j)$, takých že $v_{i-1}.v < v_j.v$. Pre našu časť konvexného obalu platí, že $\angle\ P_iP_{i-1}$ je najmenší s pomedzi všetkých $\angle\ P_jP_{i-1}$ takých, že $P_{i-1}.x < P_j.x$. Keďže $P_i.x > P_{i-1}.x$ a $P_i.y < P_{i-1}.y$, tak zo vzorca na vypočítanie $\angle\ P_iP_{i-1}$ môžme odobrať absolutné hodnoty. A teda ak je minimálny $\angle\ P_iP_{i-1}$, tak je minimálne aj $t(P_{i-1}, P_i)$, ktoré má ten istý vzorec.



- Teda $P_0, \ldots, P_m$ je totožná s $v_0, \ldots, v_m$. Keďže postupnosť je už usporiadaná, tak naplnenie binárneho stromu trvá $\mathcal{O}(n)$ a jeho výska je práve $\log n$. Pri hľadaní vlaku, ktorý bol v čase T najďalej postupujeme rovnako ako pri vkladaní uzla do stromu, s tým že si pamätáme uzol, ktorý má k nášmu T najbližšie menšie t. Vkladanie uzla do stromu trvá $\mathcal{O}(\log n)$.

□

□

Riešenie (Gabriel Foltán, 1.12.2011):

Zostrojme si graf závislosti pozície vlaku od času, kde x-ová os predstavuje čas a y-ová os znázorňuje pozíciu vlakov. Potom pozícia vlaku i v čase $t = x$ je daná nasledovne:

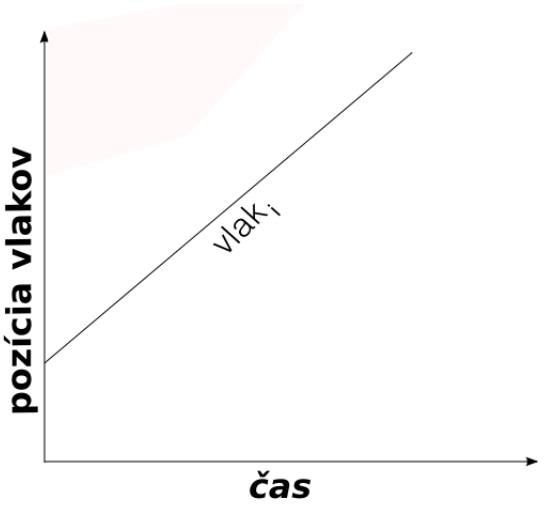
$$y = k_i + v_i x$$

Vidíme, že tento zápis predstavuje priamku(obrázok 5.41.).

Znázornime si teraz pozície n vlakov pomocou takýchto n priamok $y = k_i + v_i x$, pre $i = 1, \ldots, n$. Vedúci vlak v nejakom čase t_0 bude taký vlak, pre ktorý platí, že jeho y-súradnica v čase $t = t_0$ je najväčšia.

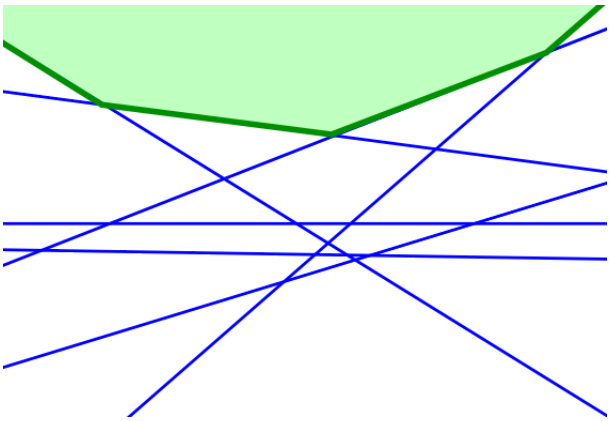
Pretože chceme zistiť množinu všetkých vlakov, ktoré boli vedúcimi v nejakom momente, to čo potrebujeme vypočítať je horná obálka takýchto priamok $y = k_i + v_i x$, pre $i = 1, \ldots, n$. Horná obálka priamok sa definuje nasledovne:

DEFINÍCIA 5.1. *Nech je daná množina n priamok $L = \{l_1, \ldots, l_n\}$, kde l_i má tvar $y = a_i x + b_i$. Tieto priamky definujú n polrovín, $y \geq a_i x + b_i$. Horná obálka L je hranica prienikov týchto polrovín.*



OBR. 5.41. pozícia vlaku i

Horná obálka akýchkoľvek priamok sa dá interpretovať podobne ako na obrazku 5.42.

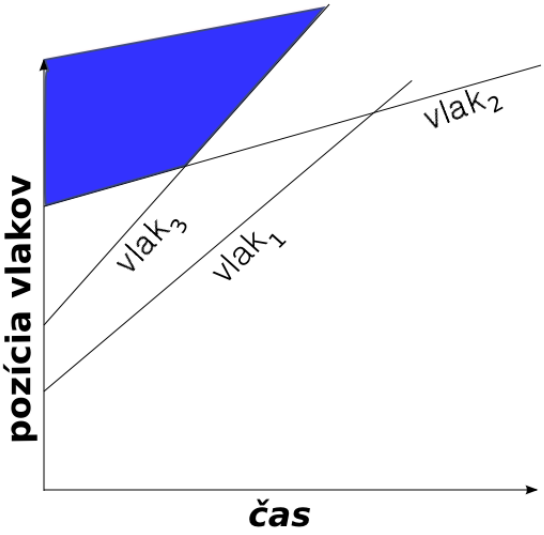


OBR. 5.42. horná obálka priamok

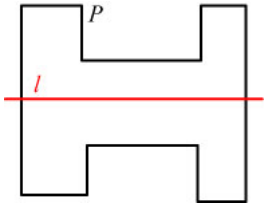
V našom prípade bude situácia vyzerať tak ako na obrázku 5.43.

To znamená, že chceme nájsť prienik n polrovín $y \geq k_i + v_i x$, pre $i = 1, \ldots, n$. To sa dá zistiť (podľa skript *Zložitosť geometrických algoritmov* str. 139) metódou *Rozdeľuj a panuj* v čase $\mathcal{O}(n \log n)$. Táto zložitosť je optimálna, čo sa dá ukázať analogicky ako v prípade konvexných obalov množiny bodov v rovine (podľa spomínaných skript). □

CVIČENIE 5.14 (10 bodov). *Môže byť mnohouholník monotónny len vzhľadom na priamky s jedným smerom? Ilustrujte svoju odpoveď obrázkom.*



OBR. 5.43. horná obálka priamok $y = k_i + v_i x$, pre $i = 1, \ldots, n$



OBR. 5.44. Mnohouholník P monotónny len vzhľadom ku smeru priamky l.

Riešenie (Lukáš Tencer, 29.10.2008):Áno môže. Napríklad pokiaľ bude mnohouholník mať tvar veľkého písmena H. Pozri obr. 5.44.

Pod pojmom monotónny mnohouholník ku priamke l chápeme mnohouholník, ktorého prienik s ľubovoľnou priamkou k ortogonálnou ku l je jeden súvislý komponent. Nejde teda o striktne monotónny mnohouholník, pre ktorý sú podmienky odlišné. □

CVIČENIE 5.15 (15 bodov). *Opíšte algoritmus vkladania a vyberania prvku z kD-stromu.*

Riešenie (Marta Režnáková, 11.11.2008): Majme množinu bodov M. Vytvorme pre ňu kD strom (pozri obr. 5.45) nasledovne (ide o rekurziu):

– urči deliacu súradnicu s (pričom v každej výške majú všetky uzly deliacu súradnicu vzhľadom na rovnakú os, nazvime si tento údaj dimenziou d, dimenzie sa striedajú), ulož ju do stromu t; príslušná časť priestoru (v prípade koreňa ide o celý priestor) je teda delená touto súradnicou na dva podpriestory

– uzol, obsahujúci vrchol v, má dvoch synov, ľavý a pravý podstrom (tieto reprezentujú spomínané dva podpriestory)

– ľavému podstromu priraď tie vrcholy, ktorých príslušná súradnica je menšia ako s (M_l) a proces delenia opakuj pre tieto vrcholy

– pravému podstromu priraď tie vrcholy, ktorých príslušná súradnica je väčšia ako s (M_r) a proces delenia opakuj pre tieto vrcholy

```
    split(){
    if (d < d_max) d++;
    else d = 0;
    s = M.getCoord(d); //vráti deliacu súradnicu dimenzie d
    return s;
}
makeTree(){
    if(|M| == 0) return NULL;
    t = newNode;
    if(|M| == 1){
        t.point = M.point(0);
        t.left = NULL;
        t.right = NULL;
    }else{
        s = split(); //vráti, podľa ktorej súradnice sa delí
        t.coord = s; t.dim = d;
        M_l, M_r;
        t.left = makeTree(M_l);
        t.right = makeTree(M_r);
    }
}
```

VKLADANIE VRCHOLA:

chceme do stromu t vložiť vrchol x (pozri obr. 5.46)

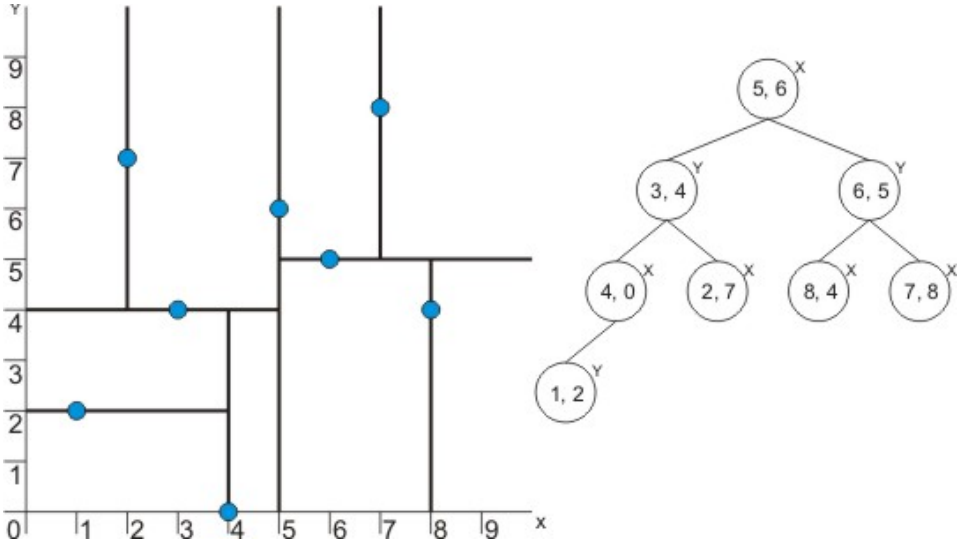
– ak je strom t prázdny, NULL, vytvoríme nový uzol a vložíme doň vkladaný vrchol x

– ak sa nachádzame v liste, ktorého hodnota je NULL, vytvoríme nový uzol a vložíme doň x

– ak sa nachádzame v uzle, v ktorom uložený vrchol v nie je totožný s x, rozhodneme sa podľa deliacej súradnice vrchola v, do ktorého podstromu, ľavého alebo pravého, treba x vložiť tak, aby sme zachovali štruktúru kD stromu

Pointou vkladania je teda nájsť oblasť v už existujúcom strome t (v prípade, že je prázdny, oblasťou je celý priestor), v ktorej sa daný bod x nachádza, tú predeliť príslušnou súradnicou a následne do stromu t vložiť bod x.

Ak by sme vkladali náhodné vrcholy, strom by mal ostať vyvážený. Pri riešení vyváženosti, resp. nevyváženosti, treba kontrolovať výšku,



Obr. 5.45. kD strom z danej množiny bodov

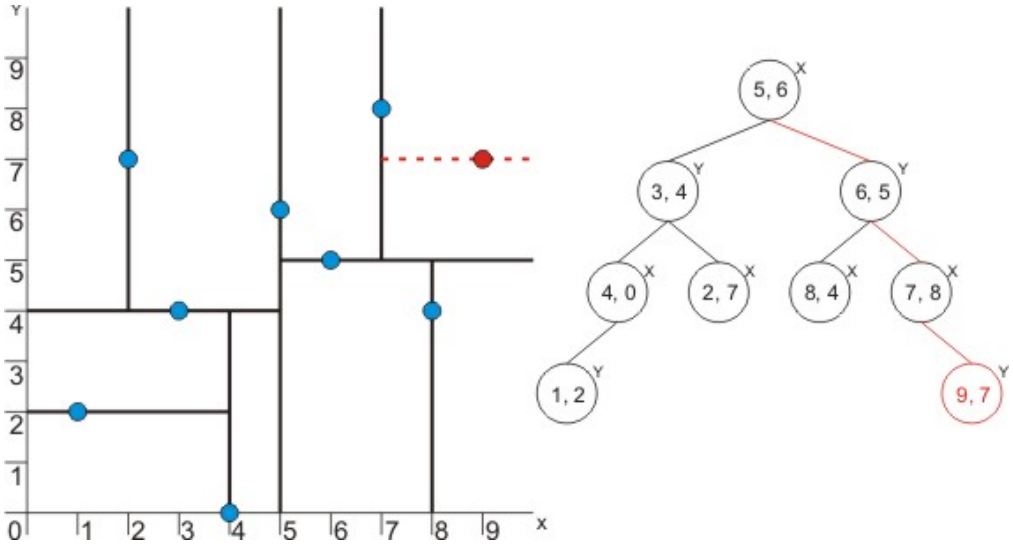
resp. hĺbku, podstromov. Ak rozdiel výšok prekročí stanovený limit (zvyčajne sa strom považuje za vyvážený, ak rozdiel výšok jeho podstromov je najviac 1), máme dva možné prístupy. Buď sa budeme snažiť strom vyvážiť napríklad tým, že presunieme vhodný vrchol z hlbšieho podstromu do koreňa a upravíme, alebo strom vytvoríme odznova.

```
insert(x, t = root){
if (t == NULL){ //sme v liste alebo je strom prázdny
    t = newNode(x);
    s = split();
    t.coord = s;
}else{
    s = t.coord; d = t.dim;
    if (x_d <= s) insert(x, t.l)
    else insert(x, t.p);
}
}
```

ODSTRAŇOVANIE VRCHOLA:

Pri odstraňovaní môžu nastať dva prípady, a síce, že odstraňovaný vrchol x bude listom (čiže nebude mať synov), alebo bude uzlom (bude mať synov). V prvom prípade stačí jednoducho list, v ktorom je vrchol x uložený, zo stromu odstrániť. V druhom prípade potrebujeme prihliadnuť na fakt, že deliace súradnice v jednotlivých výškach stromu sa striedajú, a tak teda nestačí nahradiť mazaný uzol jedným zo synov, treba nájsť taký vrchol, ktorým môžeme odstraňovaný vrchol x nahradiť.

Postupovať budeme nasledovne:



Obr. 5.46. Vkladanie vrcholu $x = (9, 7)$.

– v strome t sa nastavíme na uzol obsahujúci vrchol x - find(t, x)

– zistíme, či ide o list, ak áno, tento list vynulujeme a skončíme, ak nie, pokračujeme ďalej

– v strome t vyhľadáme vrchol x^* (pozri obr. 5.47), ktorým nahradíme odstraňovaný vrchol x - findReplacement(t, d, d^*), kde d je dimenzia mazaného bodu (resp. integer, ktorý určí, podľa ktorej osi sa priestor v danom vrchole delil súradnicou s), a d^* je aktuálna dimenzia bodu

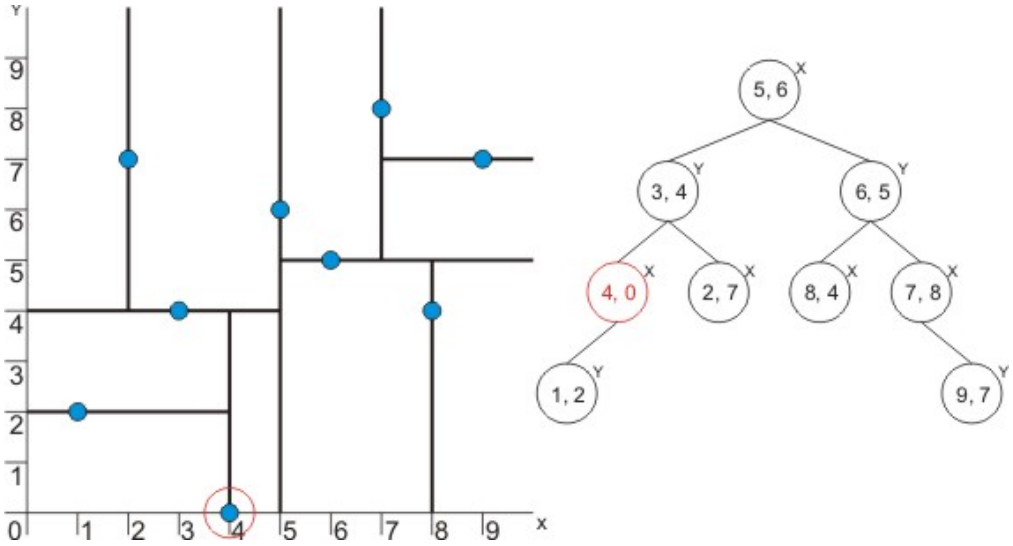
– rekurzívne budeme odstraňovať vrchol x^* (pozri obr. 5.48) z jeho pôvodného uzla (pre zachovanie vlastností kD stromu) - remove(t, x, d)

Metóda findReplacement() hľadá taký vrchol x^*, ktorý má minimálnu hodnotu vzhľadom na deliacu súradnicu, čiže hľadá vhodnú náhradu za odstraňovaný vrchol x.

– ak sme priestor delili súradnicou podľa rovnakej osi ako v odstraňovanom vrchole a ak aktuálny vrchol nemá ľavého syna, je tento minimom v podstrome uzla s vrcholom x, a teda vhodnou náhradou za vrchol x; v opačnom prípade sa vnoríme do ľavého podstromu

– ak sme priestor delili súradnicou podľa inej osi ako v odstraňovanom vrchole, nájdeme najmenší prvok z celého tohoto podstromu vrátane aktuálneho vrchola

```
findReplacement(t, d, d*){
if(d == d*){
    if(t.left == NULL) return t.point;
    else return findReplacement(t.left, d, d* + 1)
}else return min(t.point, findReplacement(t.left, d,
d* + 1), findReplacement(t.right, d, d* + 1));
```

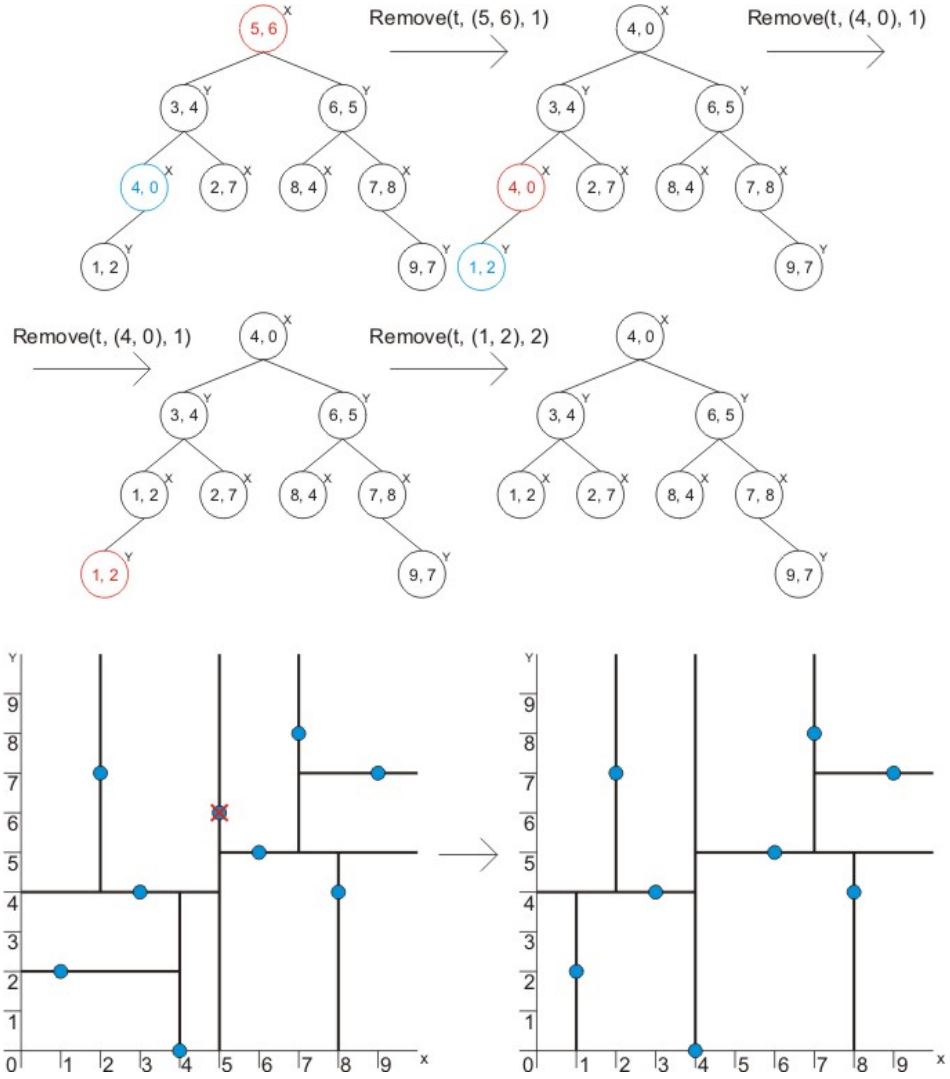


OBR. 5.47. findReplacement(t = root, 2, 1), kde 1 označuje dimenziu súradnice $s_1 = x$ a 2 označuje dimenziu súradnice $s_2 = y$

```
    }


        remove(t, x){
        if (x.coord < t.coord) remove(t.left, x, d+1);
        else if (x.coord > t.coord) remove(v.right, x, d+1);
        else{
            if ((t.left == NULL) && (t.right == NULL)) return
NULL;
            else if (t.right != NULL) t.point = findReplacement(t.right,
d, d+1);
            else{
                t.point = findReplacement(t.left, d, d+1);
                t.left = NULL;
            }
            remove(t.right, t.point, d+1); //mažeme bod, ktorým
sme nahrádzali
        }
    }
```

Daný algoritmus odstraňovania vrchola by nemal vyváženosť stromu veľmi narúšať, no pri mazaní väčšieho počtu bodov sa tomuto faktu, rovnako ako v prípade vkladania, nevyhneme. Ak teda chceme mať kD strom vyvážený (čo je predpoklad pre nízku časovú zložitosť operácií na ňom vykonávaných), treba pristúpiť k metódam vyvažovania.

□



Obr. 5.48. remove(t, root, 1)

Cvičenie 5.16 (10 bodov). *Nájdite spôsob reprezentácie axiálnych obdĺžnikov z* $\mathbb{E}^2$ *vo viacrozmernom priestore. Aký najmenší priestor (v zmysle dimenzie) treba na jednoznačnú reprezentáciu?*

Riešenie (Alojz Kováčik, 30.11.2009): Označme (a, b) a (c, d) súradnice pravého horného a súradnice ľavého dolného vrcholu axiálneho obdĺžnika. Takýto obdĺžnik môžeme reprezentovať ako bod (a, c, b, d) v $\mathbb{E}^4$. Axiálny obdĺžnik môžeme jednoznačne určiť viacerými spôsobmi, každý z nich však potrebuje minimálne 4 údaje, preto najmenší priestor, v ktorom je možné axiálny obdĺžnik jednoznačne reprezentovať je štvorrozmerný. □

CVIČENIE 5.17 (50 bodov). *Nech S je množina n axiálnych obdĺžnikov v rovine. Pre ľubovoľný obdĺžnik $R = [a, b] \times [c, d] \subseteq \mathbb{E}^2$ hľadáme množinu tých obdĺžnikov z S, ktoré sú v R. Nájdite a opíšte dátovú štruktúru, ktorá umožní vyriešiť túto úlohu v zložitosti $M\colon \mathcal{O}(n \log^3 n)$ a $T\colon \mathcal{O}(\log^4 n + k)$, kde k je počet obdĺžnikov v odpovedi.*

Riešenie (Alojz Kováčik, 30.11.2009): Majme axiálny obdĺžnik $R = [a, b] \times [c, d] \subseteq \mathbb{E}^2$ a ľubovoľný ďalší axiálny obdĺžnik $Q = [x_1, y_1] \times [x_2, y_2] \subseteq \mathbb{E}^2$. Potom Q sa nachádza v R práve vtedy, keď bod $(x_1, x_2, y_1, y_2) \in \mathbb{E}^4$ leží v intervale $[a, c] \times [a, c] \times [b, d] \times [b, d]$. Označme $F \subset \mathbb{E}^4$ množinu všetkých bodov reprezentujúcich obdĺžniky z množiny $S \subset \mathbb{E}^2$. Náš problém sa teda premietol na problém hľadania tých bodov z množiny F, ktoré patria intervalu $[a, c] \times [a, c] \times [b, d] \times [b, d] \subset \mathbb{E}^4$.

Najprv si ozrejmíme tento problém pre jednorozmerný prípad a nasledne ho rozšírime do k-rozmerov.

Jednorozmerný prípad:

Spomedzi zadaných bodov patriacich množine $S \subset \mathbb{R}$ hľadáme tie, ktoré sa nachádzajú v danom intervale $[u, v] \subset \mathbb{R}$.

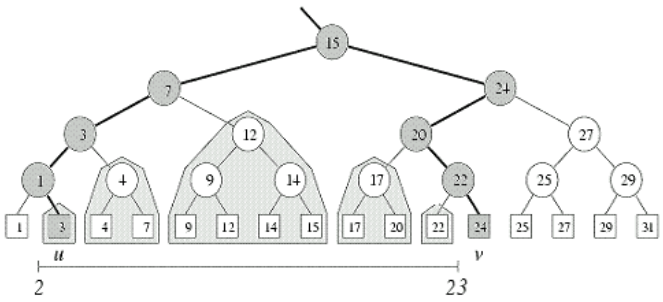
Usporiadajme body vstupnej množiny S vzostupne a uložme do listov vyváženého binárneho stromu vytvoreného z $\|S\|$ prvkov. Pripomeňme si, že vyvážený binárny strom je binárny strom, ktorý sa snaží zachovať si čo najmenšiu výšku. Každému vnútornému vrcholu stromu ďalej priraďme najväčší a najmenší bod spomedzi bodov v ľavom podstrome. Vieme, že počet vrcholov takéhoto stromu je $\mathcal{O}(n)$ a výška $\mathcal{O}(\log n)$, teda vyhľadanie bodu v liste môže trvať nanajvýš $\mathcal{O}(\log n)$.

Chceme teda vyhľadať všetky listy, nachádzajúce sa medzi u a v. Postupne prehľadávame strom od koreňa k listu, kým nenájdeme list obsahujúci bod x_1, pre ktorý platí $x_1 \geq u$, rovnako vyhľadáme list obsahujúci bod $x_2 \geq v$. V jednom z vrcholov stromu, označme ho v, sa cesty vyhľadávania bodov x_1, x_2 rozdelia. Ak napríklad cesta $v \to x_1$ pokračuje vľavo od vrchola v, pravý podstrom každého ďalšieho vrcholu cesty určite obsahuje listy s bodmi nachádzajúcimi sa medzi u, v. Taktiež ak cesta $v \to x_2$ pokračuje pravo od vrchola v ľavý podstrom každého ďalšieho vrcholu cesty určite obsahuje listy s bodmi nachádzajúcimi sa medzi u, v. Z toho vyplýva, že všetky listy s bodmi nachádzajúcimi sa medzi u a v sa nachádzajú v podstromoch ohraničených zľava cestou $u \to x_1$ a zprava $v \to x_2$.

Analýza pamäťovej a časovej zložitosti

Počet vrcholov tohto stromu je $\mathcal{O}(n)$. Teda pamäťová zložitosť je tiež $\mathcal{O}(n)$.

Analyzujme ďalej časovú zložitosť. V našom algoritme najprv nájdeme všetky podstromy, ktoré dostaneme prechodom od spomínaného vrcholu v k vrcholom x_1, x_2. Identifikácia týchto podstromov trvá nanajvýš $\mathcal{O}(\log n)$, keďže $\mathcal{O}(\log n)$ je zároveň dĺžka cesty od koreňa po vrcholy x_1, x_2.



OBR. 5.49. Na obrázku vidíme podstromy, ktoré sme dostali pri prechode od koreňa ku krajným bodom intervalu v jednorozmernom prípade.

Všetky listy daného stromu je možné vyhľadať tzv. prehľadávaním do hĺbky.

Prehľadáme každý z podstromov do hĺbky, a teda dostaneme listy všetkých podstromov, ktoré obsahujú všetky body medzi u, v. Vieme, že časová zložitosť prehľadávania vyváženého binárneho stromu do hĺbky je lineárna vzhľadom na počet listov. Preto suma všetkých časových zložitostí tohto kroku pre všetky podstromy je $\mathcal{O}(k)$, kde k počet bodov patriacich intervalu $[u, v]$.

Celková časová zložitosť algoritmu je preto $\mathcal{O}(\log n + k)$.

Viacrozmerný prípad:

Spomedzi zadaných bodov patriacich množine $M \subset \mathbb{E}^d$ hľadáme tie, ktoré sa nachádzajú v danom intervale $[a_1, b_1] \times \cdots \times [a_d, b_d]$. Na základe predchádzajúcej jednorozmernej štruktúry teraz vytvoríme d-rozmernú štruktúru nasledujúcim spôsobom:

Zo súradnice x_d zadaných bodov vytvorme vyvážený binárny strom ako v jednorozmernom prípade (ozn. **primárny strom**). Ďalej každému vnútornému vrcholu v tohto stromu priraďme $d-1$-rozmernú dátovú štruktúru zo súradníc $(x_1, x_2, \ldots, x_{d-1})$ (ozn. **sekundárna štruktúra**). Každý vrchol stromu obsahuje najmenšiu a najväčšiu d-tu súradnicu ľavého podstromu, ktorá hovorí o tom, že d-ta súradnica všetkých bodov sekundárnej štruktúry tohto vrchola sa nachádza medzi týmito dvoma súradnicami.

Teraz s použitím matematickej indukcie dokážeme, že pamäťová zložitosť takto vytvorenej štruktúry je $\mathcal{O}(n \log n^{d-1})$ a algoritmus vyhľadania podstromov obsahujúcich listy s bodmi v intervale $[a_1, b_1] \times \cdots \times [a_k, b_k]$ má časovú zložitosť $\mathcal{O}(\log n^d)$.

- Ako sme sa presvedčili v jednorozmernom prípade, pre prípad $d = 1$ tvrdenie platí.
- Predpokladajme, že máme $k-1$-rozmernú dátovú štruktúru, kde $k \geq 2$. Jej pamäťová zložitosť je $\mathcal{O}(n \log n^{d-2})$ a algoritmus

vyhľadania podstromov obsahujúcich listy s bodmi v intervale $[a_1, b_1] \times \cdots \times [a_{k-1}, b_{k-1}]$ má časovú zložitosť $\mathcal{O}(\log n^{d-1})$.

- Majme teda danú d-rozmernú štruktúru. Výška vyváženého binárneho stromu v tejto štruktúre je $\mathcal{O}(\log n)$, preto existuje $\mathcal{O}(\log n)$ úrovní tohto stromu. Zoberme ľubovoľnú úroveň stromu a označme jej vrcholy $v_i, i \leq n$. Listy podstromu, ktorého je daný vrchol v_i koreňom obsahujú pre ľubovoľné dva vrcholy $v_i, v_j, i \neq j$ rozdielne súradnice x_k. Preto celkovú veľkosť sekundárnej štruktúry pre jednu úroveň môžeme ohraničiť s použitím indukčného predpokladu nasledovne

  $$\sum_i \mathcal{O}(p_i \log^{d-2} p_i) \leq \sum_i \mathcal{O}(p_i \log^{d-2} n) \leq \mathcal{O}(n \log^{d-2} n),$$

  kde p_i je počet listov prislúchajúcich i-temu vrcholu danej úrovne a

  $$\sum_i p_i \leq n.$$

  Prenásobením počtom úrovní tak dostávame celkovú pamäťovú zložitosť $\mathcal{O}(n \log n^{d-1})$, čo sme chceli dokázať.

  Pri vyhľadavaní všetkých bodov patriacich intervalu $[a_1, b_1] \times \ldots \times [a_k, b_k]$ nájdeme v čase $\mathcal{O}(\log n)$ najviac $\mathcal{O}(\log n)$ vrcholov s prislúchajúcimi bodmi, ktorých súradnica x_k patrí intervalu $[a_k, b_k]$. Ďalej pokračujeme v prehľadávaní sekundárnych štruktúr týchto vrcholov, ktoré podľa indukčného predpokladu trvá $\mathcal{O}(\log n^{d-1})$.

  Celková časová zložitosť je teda $\mathcal{O}(\log n^d)$, čo sme chceli ukázať.

Dostali sme podstromy obsahujú listy s bodmi v danom intervale. Na základe rovnakej úvahy, ako v jednorozmernom prípade dostávame, že prehľadanie týchto podstromov do hĺbky podľa súradnice x_1, má časovú zložitosť $\mathcal{O}(h)$, kde h je počet bodov patriacich danému intervalu.

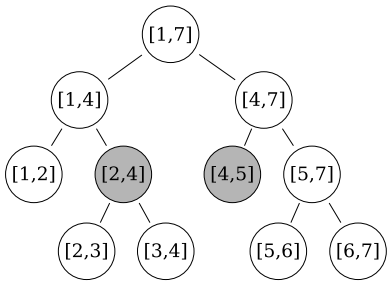
Celková časová zložitosť vyhľadania podmnožiny všetkých bodov patriacich danému intervalu je teda $\mathcal{O}(\log n^d + h) < \mathcal{O}(n \log n^d + h)$.

Na základe predchádzajúcej konštrukcie dostávame pre $d = 4$ štruktúru, ktorá je riešením nášho problému so zodpovedajúcou pamäťovou a časovou zložitosťou. □

Riešenie (Martin Manduch, 6.12.2009): Na riešenie využijeme rozsahový strom. Rozsahový strom je dátová štruktúra, ktorá vznikne rozšírením segmentového stromu.

Segmentový strom je vyvážený binárny strom, ktorý slúži na manipuláciu s intervalmi, ktoré sú uložené vo vrcholoch. Interval vo vrchole v budeme označovať $[B(v), E(v)]$. Ak je vo vrchole v interval $[B(v), E(v)]$, tak v jeho ľavom synovi je $[B(v), \lfloor (B(v) + E(v))/2 \rfloor]$ a v pravom $[\lfloor (B(v) + E(v))/2 \rfloor, E(v)]$. Požiadavka na vyhľadávanie v

segmentovom strome je interval $[a, b]$. Postupujeme od koreňa smerom dolu, ak nájdeme vrchol v taký, že $[B(v), E(v)] \subseteq [a, b]$, tak vypíšeme jeho listy. Na obr. 5.50 je segmentový strom pre pre interval $[1, 7]$, tmavšie vrcholy zobrazujú rozklad intervalu $[2, 5]$ na korene podstromov, ktoré stačí vypísať.



Obr. 5.50. Segmentový strom.

Rozsahový strom pre množinu bodov $\mathcal{P}$ v $\mathbb{E}^d$ konštruujeme induktívne. Najprv vytvoríme primárny segmentový strom podľa prvých súradníc. Teda vytvoríme segmentový strom $\mathcal{T}$, ktorý zodpovedá množine $\{x_1(P); P \in \mathcal{P}\}$. V každom vrchole $v \in \mathcal{T}$ budeme mať smerník na rozsahový strom zodpovedajúci množine $\mathcal{P}_{d-1}(v) = \{(x_2(P)), \ldots, (x_d(P)), P \in \mathcal{P}_d(v)\}$, kde množina $\mathcal{P}_d(v) = \{P; P \in \mathcal{P}, x_1(P) \in [B(v), E(v)]\}$. Inak povedané, smerník vo vrchole primárneho stromu bude na taký rozsahový strom, ktorý obsahuje len body, ktoré majú prvé súradnice v intervale dané tým vrcholom.

Postup vyhľadávania v rozsahovom strome je rovnaký ako segmentovom strome s tým rozdielom, že ak nájdeme vrchol, ktorého interval je z požadovaného intervalu, nevypíšeme ho, ale pokračujeme vyhľadávanie cez smerník na rozsahový strom s nižšou dimenziou. Až keď sa dostaneme na dimenziu 1, môžeme listy vypísať.

Časová zložitosť
Označme čas vyhľadanie $T(n, d)$. Platí rovnosť

$$T(n, d) = \mathcal{O}(\log n) + \sum_{v \ z \ rozkladu \ intervalu} T(n(v), d - 1),$$

kde $\mathcal{O}(\log n)$ reprezentuje rozklad intervalu v prvej súradnici na maximálne $\mathcal{O}(\log n)$ častí v primárnom segmentovom strome a suma vyjadruje zložitosť určenia požadovaných bodov v týchto častiach. $n(v) = E(v) - B(v)$ označuje maximálny počet bodov, ktoré môžu byť z tohoto intervalu požadované. V segmentovom strome je najviac $2\lceil \log_2 n \rceil - 2$ častí ľubovoľného intervalu a triviálne je $n(v) \leq n$. $T(n, d)$ môžeme odhadnúť:

$$T(n, d) = \mathcal{O}(\log n) T(n, d - 1).$$

Vieme, že $T(n, 1) = \mathcal{O}(\log n)$ a tak $T : \mathcal{O}(\log^d n)$. Počet vnútorných uzlov binárneho stromu je menší ako počet listov, preto čas výpisu k

listov je lineárny k počtu listov – $O(k)$ a teda celkový čas behu algoritmu vyhľadávania pre k odpovedí je $\mathcal{O}(k + \log^d n)$.

Pamäťová zložitosť
Označme pamäťovú zložitosť $M(n, d)$. Platí:

$$M(n,d) = \mathcal{O}(n) + \sum_{v \ z \ primarneho \ stromu} M(n(v), d-1),$$

kde $\mathcal{O}(n)$ vyjadruje pamäťovú zložitosť primárneho stromu a v sume sú asociované rozsahové stromy k vrcholom primárneho stromu. Odhadnime druhý člen. V primárnom strome pre koreň je $n(v) \leq 2^{\lceil \log_2 n \rceil}$. Najviac dva vrcholy (synovia koreňa) môžu dosahovať $n(v) \leq 2^{\lceil \log_2 n \rceil - 1}$, maximálne 4 vrcholy majú $n(v) \leq 2^{\lceil \log_2 n \rceil - 2}$ atď. Môžeme to zapísať ako:

$$M(n,d) \leq \mathcal{O}(n) + \sum_{i=0}^{\lceil \log_2 n \rceil} 2^{\lceil \log_2 n \rceil - i} M(2^i, d-1).$$

Dokážeme indukciou, že $M(n,d) = \mathcal{O}(n \log^{d-1} n)$. Keďže $M(n,1) = \mathcal{O}(n)$ pre $d = 1$ tvrdenie platí. Nech platí tvrdenie pre dimenziu d, teda $M(n,d) = \mathcal{O}(n \log^{d-1} n)$. Ukážeme, že platí pre dimenziu $d+1$. Dosadením IP dostaneme:

$$M(n,d+1) \leq \mathcal{O}(n) + \sum_{i=0}^{\lceil \log_2 n \rceil} 2^{\lceil \log_2 n \rceil - i} C 2^i \log^{d-1} 2^i =$$

$$= \mathcal{O}(n) + C \sum_{i=0}^{\lceil \log_2 n \rceil} 2^{\lceil \log_2 n \rceil} \log^{d-1} 2^i \leq$$

$$\leq \mathcal{O}(n) + C(n+1) \sum_{i=0}^{\lceil \log_2 n \rceil} \log^{d-1} 2^i \leq$$

$$\leq \mathcal{O}(n) + C(n+1) \lceil \log_2 n \rceil \log^{d-1} n$$

C je konštanta z $\mathcal{O}$-notácie. Z posledného riadku je zrejmé, že $M(n, d+1) = \mathcal{O}(n \log^d n)$.

Axiálny obdĺžnik v $\mathbb{E}^2$ vieme jednoznačne reprezentovať v $\mathbb{E}^4$ ako štvoricu $(x_{min}, x_{max}, y_{min}, y_{max})$, preto na vyľadávanie použijeme 4-rozmerný rozsahový strom. Keď chceme zistiť, ktoré axiálne obdĺžniky ležia v obdĺžniku $[a,b] \times [c,d]$, musíme v rozsahovom strome hľadať body $(x_{min}, x_{max}, y_{min}, y_{max})$ také, že $x_{min} \in [a,b], x_{max} \in [a,b]$ a $y_{min} \in [c,d], y_{max} \in [c,d]$. Teda požiadavkou na vyhľadanie bude 4-rozmerný interval $[a,b] \times [a,b] \times [c,d] \times [c,d]$. Použili sme dátovú štruktúru rozsahový strom, preto bude T: $\mathcal{O}(k + \log^4 n)$ a M: $\mathcal{O}(n \log^3 n)$. □

CVIČENIE 5.18 (50 bodov). *Nech S je množina n mnohouholníkov v rovine. Pre ľubovoľný obdĺžnik $R = [a, b \times [c, d] \subseteq \mathbb{E}^2$ hľadáme množinu tých mnohouholníkov z S, ktoré sú v R. Nájdite a opíšte dátovú štruktúru, ktorá umožní vyriešiť túto úlohu v zložitosti $M\colon \mathcal{O}(n \log^3 n)$ a $T\colon \mathcal{O}(\log^4 n + k)$, kde k je počet mnohouholníkov v odpovedi.*

Riešenie (Alojz Kováčik, 30.11.2009): Riešenie tejto úlohy môžeme zredukovať na riešenie cvičenia 5.17. Pre každý mnohouholník P z S určíme najmenší axiálny obdĺžnik Q, ktorý ho obsahuje. Tento obdĺžnik určíme nájdením najväčšej a najmenšej x-vej a y-onovej súradnice spomedzi bodov určujúcich polygón.

Ak mnohouholník $P \in R$, potom aj $Q \in R$ a naopak. Z tejto ekvivalencie vyplýva, že na to aby sme našli všetky mnohouholníky z S patriace R, potrebujeme nájsť všetky najmenšie axiálne obdĺžniky prislúchaúce daným polygónom, patriace R. V predchádzajúcom cvičení sme našli štruktúru, ktorá rieši túto úlohu v zložitosti $M : \mathcal{O}(n \log n^3)$ a $T : \mathcal{O}(n \log n^4 + k)$, kde k je počet obdĺžnikov v odpovedi. Táto štruktúra je našim riešením. □

CVIČENIE 5.19 (30 bodov). *Nech $P \subset \mathbb{E}^2$ je množina n bodov. Nájdite takú hodnotu $k > 0$, že transformácia daná predpisom $x' = x + ky, y' = y$ nezmení poradie bodov P podľa súradnice x. Nájdite algoritmus so zložitosťou $\mathcal{O}(n \log n)$.*

Riešenie (Jana Hlinková, 28.11.2008):

Pozrime sa najprv na prípad dvoch bodov z $\mathbb{E}^2$. Všeobecne sú to body $[x_1, y_1]$ a $[x_2, y_2]$. Nech $x_1 < x_2$. Hľadáme také $k > 0$, pre ktoré bude platiť $x_1 + ky_1 < x_2 + ky_2$, a teda poradie bodov vzhľadom na x-ovú súradnicu ostane po transformácii zachované. Riešením nerovnice získame horné ohraničenie pre k:

$$k < \frac{x_2 - x_1}{y_1 - y_2} \qquad \text{ak} \quad y_1 - y_2 > 0.$$

V prípade, keď $y_1 - y_2 < 0$ získame dolné ohraničenie nejakým záporným číslom (z uvedeného usporiadania vyplýva, že rozdiel x-ových hodnôt je kladný, a teda celý zlomok je záporné číslo). Avšak hľadáme $k > 0$, a teda tento prípad vyber k neovplyvní (môžeme zvoliť ľubovoľné kladné číslo a transformácia zachová poradie).

Tento postup hľadania vhodného k sa dá jednoducho rozšíriť aj pre n bodov v rovine. Majme body uložené v poli. Utriedime ich podľa x-ovej súradnice. Pre každé dva susedné body vyriešime nerovnice

$$x_i + ky_i < x_{i+1} + ky_{i+1} \qquad i = 0, \ldots, n-1.$$

V prípade $y_i - y_{i+1} > 0$ do pomocného poľa uložíme hodnotu horného ohraničenia $\frac{x_{i+1} - x_i}{y_i - y_{i+1}}$. V opačnom prípade uložíme hodnotu reprezentujúcu maximálnu hodnotu (napr. DOUBLE_ INFINITY). Ostáva

už iba nájsť minimum z prvkov v pomocnom poli. Potom pre $0 < k < minimum$ budú všetky nerovnosti splnené, a teda usporiadanie každých dvoch susedných bodov ostane zachované. Z vlastnosti tranzitivity usporiadania potom vyplýva, že poradie všetkých n bodov z P taktiež ostane zachované. Hľadaným k je ľubovoľné k z intervalu $(0, minimum)$.

Analýza časovej zložitosti:

- utriedenie bodov podľa x-ovej súradnice: $\mathcal{O}(n \log n)$
- vyriešenie $n - 1$ nerovníc: $\mathcal{O}(n)$
- nájdenie minima: $\mathcal{O}(n)$
- zvolenie k z intervalu $(0, minimum)$: $\mathcal{O}(1)$
- celková zložitosť: $\mathcal{O}(n \log n)$

Analýza pamäťovej zložitosti:

- uloženie vstupu v jednorozmernom poli: $\mathcal{O}(n)$
- pomocné pole s $n - 1$ záznamami: $\mathcal{O}(n)$
- celková zložitosť: $\mathcal{O}(n)$

□

Riešenie (Lukáš Apalovič, 1.12.2008):

Najskor si množinu P usporiadame podľa x-ovej súradnice a v prípade, že sa dva body zhodujú v súradnici x, usporiadame ich podľa y. Do premennej k_{max} vložíme maximálnu hodnotu z definičného oboru. Táto hodnota nám bude určovať maximálnu možnú hodnotu k. Následne prechádzame usporiadanú množinu P' od najmenšieho bodu po najvačší. Pre každú po sebe idúcu dvojicu bodov $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$, kde $p_1 < p_2$ hľadáme maximálne k, ktoré zachová podmienku $x_1 + ky_1 < x_2 + ky_2$. Ak je k menšie ako k_{max}, tak do k_{max} vložíme k. Možu nastať tieto prípady:

(1) $x_1 = x_2$
Potom musí platiť $ky_1 < ky_2$. Z podmienky usporiadania vyplýva, že $y_1 < y_2$, čiže nerovnosť platí pre ľubovoľné k. k_{max} teda ostáva nezmenené.

(2) $x_1 < x_2$
Tu možu nastať tri prípady:

(a) $y_1 = y_2$
Potom $ky_1 = ky_2$ pre ľubovoľné k, čiže musí platiť nerovnosť $x_1 < x_2$ a tá podľa predpokladu platí. k_{max} ostáva nezmenené.

(b) $y_1 < y_2$
Z predpokladov vyplýva: $x_1 - x_2 < 0$ a $y_2 - y_1 > 0$.
$x_1 + ky_1 < x_2 + ky_2$
$x_1 - x_2 < k(y_2 - y_1)$ možeme vydeliť $(y_2 - y_1)$ bez zmeny znamienka

$\frac{x_1-x_2}{y_2-y_1} < k$
Táto nerovnosť platí vždy, pretože $x_1-x_2 < 0$, $y_2-y_1 > 0$ a $k > 0$. k_{max} ostáva nezmenené.

(c) $y_1 > y_2$
Z predpokladov vyplýva: $x_1 - x_2 < 0$ a $y_2 - y_1 < 0$.
$x_1 + ky_1 < x_2 + ky_2$
$x_1 - x_2 < k(y_2 - y_1)$ po vydelení $(y_2 - y_1)$ musíme zmeniť znamienko
$\frac{x_1-x_2}{y_2-y_1} > k$
Hodnote k_{max} priradíme teda hodnotu $\frac{x_1-x_2}{y_2-y_1}$, ak je menšia ako k_{max}.

Po prechode celou množinou sme získali nejaké $k_{max} > 0$. Hodnota k, ktorá pre danú transformáciu nezmení poradie bodov P podľa súradnice x je z otvoreného intervalu $(0, k_{max})$.

Analýza zložitosti

Usporiadanie množiny P má časovú zložitosť $\mathcal{O}(n \log n)$. Množinu prejdeme v lineárnom čase, čiže celková zložitosť algoritmu je $\mathcal{O}(n \log n)$.
□

Cvičenie 5.20 (35 bodov). *Opíšte podrobne procedúru tvorby stromu v metóde reťazcov. Určte jeho zložitosť. Nájdite príklad, kde by bola pamäťová zložitosť bez optimalizácie uloženia hran kvadratická.*

Cvičenie 5.21 (35 bodov). *Opíšte podrobne procedúru tvorby stromu v metóde lichobežníkových oblastí. Zamerajte sa najmä na časť získania oddeľujúcich hrán. Určte jeho zložitosť.*

Cvičenie 5.22 (40 bodov). *Nájdite príklad, v ktorom nadobúda metóda reťazcov svoju asymtotickú zložitosť. Svoje tvrdenie zdôvodnite.*

Cvičenie 5.23 (35 bodov). *Podrobne opíšte algoritmus regularizácie grafu spolu s príkladom typických situácií a pseudokódom.*

Cvičenie 5.24 (35 bodov). *Opíšte detailne procedúru vytvorenia dátovej štruktúry k metóde postupných triangulácií spolu s pseudokódom.*

Cvičenie 5.25 (35 bodov). *Opíšte detailne procedúru vytvorenia dátovej štruktúry k metóde reťazcov spolu s pseudokódom.*

Cvičenie 5.26 (35 bodov). *Ukážte, že pre ľubovoľné $\varepsilon > 0$ možno zostrojiť viacškálové vyhľadávanie prvkov fixnej množiny $P \subset \mathbb{E}^2$ v axiálnom obdĺžniku, ktorého pamäťová zložitosť je $\mathcal{O}(n^{1+\varepsilon})$ a časová zložitosť je $\mathcal{O}(m \log n + k)$, kde m je počet škál a k je mohutnosť výstupu.*

KAPITOLA 6

Prieniky

CVIČENIE 6.1 (30 bodov). *Popíšte algoritmus na vyhľadanie prieniku dvoch konvexných n-uholníkov, nájdite jeho zložitosť a zistite, či je asymptoticky optimálny.*

Riešenie (Miroslava Fekiačová, 5.12.2008): Dané sú 2 konvexné mnohouholníky P, ktorý má m hrán a Q s n hranami.

Algoritmus s použitím hrubej sily:

(1) Vyrátaj body prieniku medzi mnohouholníkmi P a Q a ulož tieto body do množiny R.

(2) Vlož do množiny R všetky body z mnohouholníka P, ktoré sa nachádzajú v mnohouholníku Q a body z Q, ktoré sa nachádzajú v mnohouholníku P.

(3) Vieme, že prienik konvexných mnohouholníkov je konvexný mnohouholník, takže prienik mnohouholníkov P a Q je konvexným obalom množiny bodov R. Tento mnohouholník má maximálne $m + n$ hrán.

Analýza algoritmu

Krok (1): Pre každú hranu z mnohouholníka P hľadáme priesečník s hranami z mnohouholníka Q, čo nám zaberie čas $\mathcal{O}(nm)$

Krok (2): Vyrátanie polohy každého bodu z P a každého bodu z Q nám zaberie čas $\mathcal{O}(nm)$

Krok (3): Konvexný obal vieme vytvoriť v čase $\mathcal{O}((n+m)log(n+m))$

Z toho vyplýva, že zložitosť algoritmu je $\mathcal{O}(nm)$, čiže algoritmus nie je časovo optimálny. □

Riešenie (Pavol Beluško, 15.12.2008):

Úlohu vyriešime napr. pomocou algoritmu M. I. Shamosa (kniha Geometric intersection problems) pre všeobecnejší prípad, keď testujeme konvexný m-uholník a n-uholník, v čase $\mathcal{O}(m + n)$. V prípade zo zadania to znamená časovú naročnosť $\mathcal{O}(n)$. **Algoritmus a analýza časovej náročnosti:**

(1) Prerozdeľme mnohouholníky tak, že každým vrcholom oboch mnohouholníkov vedieme horizontálnu priamku. Vznikne $\mathcal{O}(m+n)$ pásov.

Pri hľadaní prienikov horizontálnej priamky s hranicou mnohouholníkov nie je potrebné testovať na $\mathcal{O}(m+n)$ hranách. Konvexnosť mnohouholníkov indukuje usporiadanie vrcholov (v našom prípade je podstatné usporiadanie podľa y). Pre oba mnohouholníky udržujeme dvojicu „aktuálnych" hrán. Akonáhle prejdeme s y-súradnicou horizontálnej priamky nižšie ako je najnižší koncový bod niektorej z aktuálnych hrán, nahradíme ju jej susednou hranou v zostupnom poradí podľa y. Keďže testujeme na konštantne veľa hranách, nájdenie prieniku pásu s mnohouholníkmi trvá $\mathcal{O}(1)$.

Pozn.: Jednotlivé takto oddelené časti mnohouholníkov sú štvor- alebo trojuholníky.

Prvý krok vyžaduje spolu $\mathcal{O}(m+n)$ operácií.

(2) Pre každý pás počítame prienik dvoch nanajvýš štvoruholníkov. Ich prienik možno vypočítať v čase $\mathcal{O}(1)$, pretože počet ich vrcholov je zhora ohraničený konštantou. Prienik počítame $\mathcal{O}(m+n)$-krát, preto celý tento krok vyžaduje $\mathcal{O}(m+n)$ operácií.

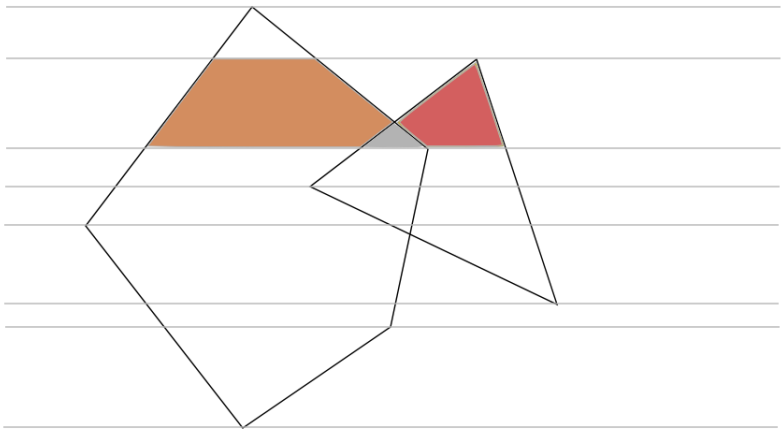
(3) Na záver je potrebné zjednotiť $\mathcal{O}(m+n)$ oblastí prienikov. Vrcholy mnohouholníka tvoriaceho vysledný prienik možno získať traverzovaním po čiastkových prienikoch.. po ľavej strane smerom dole a po pravej smerom hore. Prebytočné vrcholy odstránime tak, že pretraverzujeme vrcholy prieniku a ak sa i-ty vrchol nachádza medzi $(i-1)$. a $(i+1)$. vrcholom, vynecháme ho. Tvorba prieniku a vynechanie prebytočných vrcholov vyžaduje spolu $\mathcal{O}(m+n)$ operácií.

Analýza pamäťovej náročnosti:

V algoritme je použitých $\mathcal{O}(m+n)$ štvor-/troj-uholníkov, z ktorých získame $\mathcal{O}(m+n)$ čiastkových prienikov. Výstup je jeden mnohouholník s $\mathcal{O}(m+n)$ vrcholmi. Okrem týchto štruktúr nie je použitá žiadna iná pomocná štruktúra. Pamäťová náročnosť je teda $\mathcal{O}(m+n)$.

Záver: Pre prípad zo zadania, kedy $m=n$, je časová aj pamäťová náročnosť $\mathcal{O}(n)$. Prienik nie je možné vytvoriť bez toho, aby sme použili všetky vrcholy oboch mnohouholníkov, algoritmus v čase $\mathcal{O}(n)$ je teda asymptoticky optimálny. □

Riešenie (Jana Hlinková, 18.12.2008): Prienikom dvoch konvexných n-uholníkov M_1 a M_2 je konvexný mnohouholník (ozn. $M_1 \cap M_2$), ktorého hranica má maximálne $2n$ hrán, pričom tieto hrany sú hrany alebo časti hrán pôvodných dvoch n-uholníkov. Konvexný mnohouholník je jednoznačne určený svojou hranicou. Na určenie prieniku teda stačí nájsť hranicu mnohouholníka $M_1 \cap M_2$. Hranicu nájdeme pomocou zametacej metódy v optimálnom čase $\mathcal{O}(n)$. (Lineárny čas potrebujeme vo všeobecnosti na vymenovanie hranice mnohouholníka, preto lineárny čas považujeme za optimálny.)



Obr. 6.1. Tvorba čiastkových prienikov.

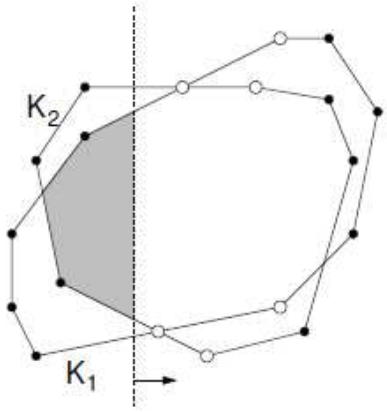
Na určenie hranice mnohouholníka $M_1 \cap M_2$ je potrebné nájsť prieniky hrán mnohouholníkov M_1 a M_2. Ľahko vieme ukázať, že každá hrana z M_1 pretne maximálne dve hrany z M_2 a tak isto aj naopak (pri predpoklade, že daná hrana neprechádza vrcholom druhého mnohouholníka). Ak by totiž existovala hrana h ktorá by preťala aspoň tri hrany M_2, znamenalo by to, že M_2 nie je konvexný.

V lineárnom čase nájdeme v každom mnohouholníku vrchol, ktorý má najmenšiu x-ovú súradnicu (jedným prechodom poľa popisujúceho hranicu M_1 zistíme vrchol s minimálnou x-ovou súradnicou mnohouholníku M_1, obdobne vrchol s minimálnou x-ovou súradnicou mnohouholníku M_2). Z týchto dvoch vrcholov vyberieme vrchol s väčšou x-ovou súradnicou. V tomto vrchole začneme zametanie vertikálnou priamkou. (Je zrejmé, že prienik mnohouholníkov bude ležať napravo od tohto vrcholu.)

Pre jednoduchšiu implementáciu rozdelíme hranice oboch mnohouholníkov na hornú a dolnú lomenú čiaru (na to je ešte potrebné nájsť v každom mnohouholníku vrchol s maximálnou x-ovou súradnicou (pozn.: tieto vrcholy využijeme aj pri zametaní – v ľavejšom z týchto vrcholov sa bude končiť zametanie)). Takto budeme vedieť v lineárnom čase utriediť vrcholy oboch mnohouholníkov podľa x-ovej súradnice: horné a dolné lomené čiary vymenovávajú vrcholy v utriedenom poradí, stačí už len tieto štyri reťazce správne spojiť, čo vieme urobiť v lineárnom čase. (A teda dôležité pozorovanie: môžeme uvažovať o algoritme bežiacom v lepšom čase ako $\Omega(n \log n)$.)

Zložky zametacieho algoritmu:

- *udalosti:* vrcholy mnohouholníkov a priesečníky
- *stav zametacej priamky:* zoznam hrán, ktoré zametacia priamka práve pretína, spolu s informáciou o mnohouholníku, ktorému patria. Z konvexnosti mnohouholníkov vyplýva, že zametacia



Obr. 6.2. Hľadanie prieniku zametaním (zdroj: *Design and Analysis of Computer Algorithms, David M. Mount*)

priamka pretína maximálne štyri hrany v jednom okamihu, teda v zozname sú vždy maximálne štyri hrany.

- *horná lomená čiara prieniku (**H**):* pole, v ktorom si ukladáme zatiaľ nájdenú časť hornej lomenej čiary hranice prieniku
- *dolná lomená čiara prieniku (**D**):* pole, v ktorom si ukladáme zatiaľ nájdenú časť dolnej lomenej čiary hranice prieniku

Zametací algoritmus funguje nasledovne:

- Na začiatku vložíme do udalosťovej fronty vrcholy z oboch mnohouholníkov, ktoré sú napravo od vrcholu, v ktorom začíname zametanie a naľavo od vrcholu, v ktorom skončíme zametanie. Taktiež do fronty vložíme vrchol, v ktorom začíname zametanie a v ktorom končíme zametanie. Vrcholy vieme vložiť v utriedenom poradí vzhľadom na x-ovú súradnicu v lineárnom čase (ako som už spomenula vyššie).
- Kým nie je udalosťová fronta prázdna, vyberieme z nej vrchol s najmenšou x-ovou súradnicou. Podľa typu vrcholu vykonáme jednu z nasledovných operácií:
  - *vrchol, v ktorom začíname zametanie:* do poľa, kde si ukladáme status zametacej priamky (ozn. **Z**) vložíme hrany incidentné s týmto vrcholom, ako aj hrany druhého mnohouholníka, ktoré práve pretína zametacia priamka (teda hrany, ktorých začiatočné vrcholy sú naľavo od vrcholu, v ktorom začíname zametanie, a ktorých koncové vrcholy sú napravo.) Overíme, či tento vrchol patrí obom mnohouholníkom, ak hej, zapíšeme si ho ako vrchol s najmenšou x-ovou súradnicou hranice prieniku $M_1 \cap M_2$ do polí **H** a **D**. Pre hrany v **Z** overíme, či existuje prienik medzi nejakou hranou z M_1 a nejakou hranou z M_2. Testujeme štyri

dvojice, čo spotrebuje konštantný čas. Ak nájdeme prienik, spracujeme to ako udalosť *priesečník* (pozri nižšie).
- *pravý koncový vrchol hrany:* overíme, či vrchol patrí obom mnohouholníkom, ak hej, spracujeme ho ako priesečník. Ďalej hranu s ním incidentnú, ktorá bola v **Z** vyberieme a miesto nej do **Z** vložíme druhú incidentnú hranu. Otestujeme prieniky, ak sú, tak ich spracujeme ako udalosť *priesečník*.
- *vrchol, v ktorom končíme zametanie:* (teda ľavejší z najpravejších vrcholov mnohouholníkov) ak patrí obom mnohouholníkom, zapíšeme ho do oboch polí **H** a **D**, a je to teda najpravejší vrchol prieniku. Ak nepatrí obom mnohouholníkom, znamená to, že najpravejší vrchol prieniku leží naľavo od neho, a teda bol už zapísaný do jedného z polí **H** a **D**. Z týchto dvoch polí už teda vieme jednoducho vytvoriť pole popisujúce hranicu prieniku $M_1 \cap M_2$.
- *priesečník:* Ak v poliach **H** a **D** ešte nie je zapísaný žiaden vrchol, zapíšeme tento priesečník do oboch polí (bude to začiatočný vrchol oboch lomených čiar mnohouholníku $M_1 \cap M_2$). Inak zapíšeme tento priesečník na prvú voľnú pozíciu do vhodného z polí **H** alebo **D**.

Analýza časovej zložitosti:

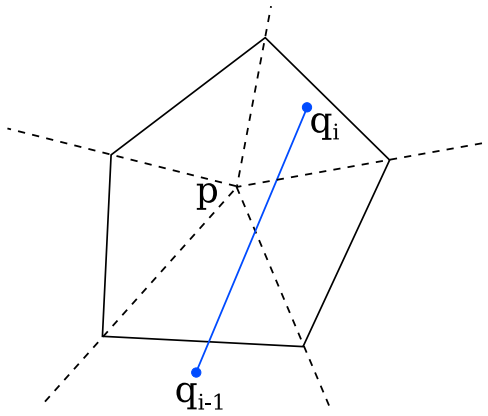
- nájdenie vrcholov s extrémnymi x-ovými súradnicami: $\mathcal{O}(n)$
- vloženie vrcholov do fronty udalostí: $\mathcal{O}(n)$
- spracovanie udalosti: $\mathcal{O}(1)$
- počet udalostí: $\mathcal{O}(n)$
- celková zložitosť: $\mathcal{O}(n)$

□

Riešenie (Martin Manduch, 3.1.2010): Majme dva konvexné mnohouholníky P, Q. Nech P má m hrán a Q má n. Vrcholy P označme $\mathbf{p}_1, \ldots, \mathbf{p}_m$ a vrcholy Q označme $\mathbf{q}_1, \ldots, \mathbf{q}_n$.

Algoritmus:

(1) Nájdeme bod $\mathbf{p}$ vo vnútri P.
(2) Vytvoríme sektory pomocou polpriamok $\overrightarrow{\mathbf{pp}_i}$.
(3) Nájdeme sektor, v ktorom sa nachádza $\mathbf{q}_1$ a zistíme, či je vo vnútri P.
(4) Pre $i = 2, \ldots, n$ hľadáme prienik hrany $\mathbf{q}_{i-1}, \mathbf{q}_i$ s P. Postupne od segmentu, v ktorom je $\mathbf{q}_{i-1}$ hľadáme prieniky s polpriamkami v jednom smere (proti smeru hodinových ručičiek alebo naopak). Prienik s poslednou určuje, v ktorom sektore je $\mathbf{q}_i$ (pozri obr. 6.3). Tiež zistíme, či je $\mathbf{q}_i$ vo vnútri P. Určíme prienik $\mathbf{q}_{i-1}, \mathbf{q}_i$ s P.



Obr. 6.3. Hľadanie sektora, v ktorom leží q_i.

Keďže sme išli postupne po hranách Q tak, ako susedia, prieniky hrán s mnohouholníkom P sú správne usporiadané a nemusíme ich ďalej triediť. Do výslednej postupnosti vrcholov, ktorá reprezentuje $P \cap Q$ ešte potrebujeme pridať niektoré vrcholy z P. To môžeme urobiť počas algoritmu tak, že si budeme pamätať popri prienikoch aj hrany P, ktorým patria a použijeme informáciu, ktoré vrcholy z Q patria P.

Časová zložitosť:

Prvý krok algoritmu trvá $\mathcal{O}(1)$, druhý aj tretí $\mathcal{O}(m)$. Vo štvrtom kroku hľadáme prieniky hrán $\mathbf{q}_{i-1}, \mathbf{q}_i$ s polpriamkami $\overrightarrow{\mathbf{pp}_i}$. Ak má hrana k prienikov s polpriamkami, vieme ich nájsť v čase $\mathcal{O}(k)$. Keďže každá polpriamka pretína najviac dve hrany Q, počet všetkých prienikov s polpriamkami je najviac $2m$. Počas ich hľadania musíme prejsť všetkými n hranami Q, preto je celkový čas nájdenia prienikov s polpriamkami $\mathcal{O}(m+n)$. Keď už poznáme sektor, v ktorom je bod $\mathbf{q}_i$, tak zistenie, či patrí P a nájdenie prieniku hrany $\mathbf{q}_{i-1}, \mathbf{q}_i$ s P trvá $\mathcal{O}(1)$. Spolu pre všetky vrcholy $\mathcal{O}(n)$. Doplnenie vrcholov z P do výslednej postupnosti trvá $\mathcal{O}(m+n)$, lebo musíme prejsť všetky prieniky hrán P a Q. Výsledný čas nájdenia $P \cap Q$ je $\mathcal{O}(m+n)$. □

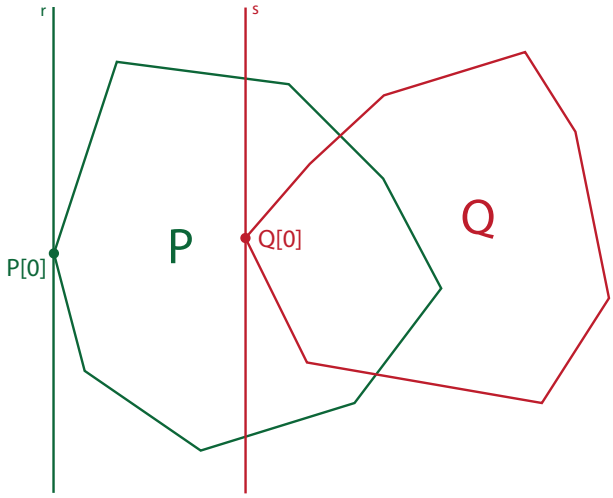
Riešenie (Dušan Pácal, 13.12.2008): Najprv načrtneme algoritmus v hrubých krokoch:

- Nájdeme oporné úsečky,
- určíme kosce a nájdeme ich vrcholy, dostaneme body prieniku hraníc mnohouholníkov,
- z bodov prieniku a hrán mnohouholníkov vybudujeme riešenie.

Nájdenie oporných úsečiek

Na vstupe majme dva konvexné mnohouholníky P a Q, reprezentované postupnosťami vrcholov, napr. proti smeru chodu hodinových ručičiek. Nech prvý vrchol postupnosti je vrchol s najmenšou x-ovou súradnicou.

Majme dve zvislé priamky r, s, ktoré prechádzajú bodmi $P[0]$ a $Q[0]$ (obr. 1).



Obr. 6.4. mnohouholníky s priamkami prechádzajúcimi ich prvým vrcholom

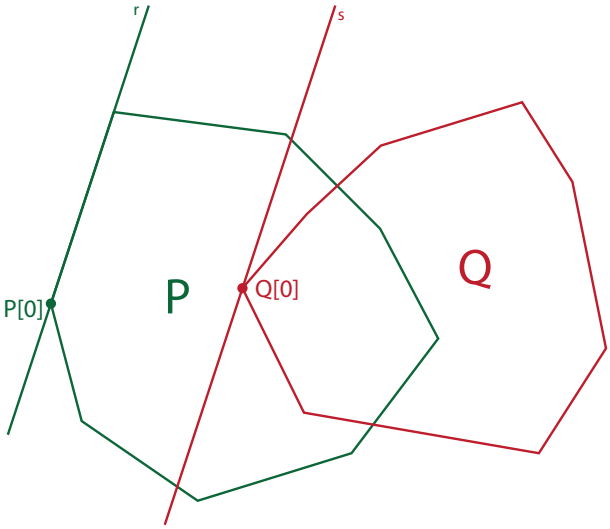
Keďže priamky sú orientované, vieme určiť, že priamka r je naľavo od priamky s. Obe priamky budeme teraz otáčať: V každom kroku sa pozrieme na nasledujúci vrchol a určíme uhol priamky r a úsečky $P[0]P[1]$ (resp. $P[i]P[i+1]$) a uhol priamky r a úsečky $Q[0]Q[1]$ (resp. $Q[j]Q[j+1]$). Obe priamky otočíme o menší uhol okolo začiatočných bodov (obr 2). Ak sme našli menší uhol v mnohouholníku P, posunieme sa do bodu $P[1]$ resp. $P[i+1]$. Okolo tohoto vrchola budeme potom otáčať priamku v nasledujúcom kroku. Rovnako sa posunieme do nasledujúceho bodu, ak bol menší uhol pri mnohouholníku Q.

Pozrieme sa na priamky - či ostalo zachované, že r je naľavo od s. Ak áno, pokračujeme s otáčaním priamok. Ak nie, teda r je napravo od s, našli sme priečku spájajúcu mnohouholníky P a Q. Krajnými bodmi priečky budú body $P[i], Q[j]$ - body, okolo ktorých sme otáčali priamky (obr 3). Takáto hrana nepatrí ani jednému z mnohouholníkov, patrí však konvexnému obalu zjednotenia bodov oboch mnohouholníkov. S otáčaním priamok skončíme, keď sme obišli oba mnohouholníky.

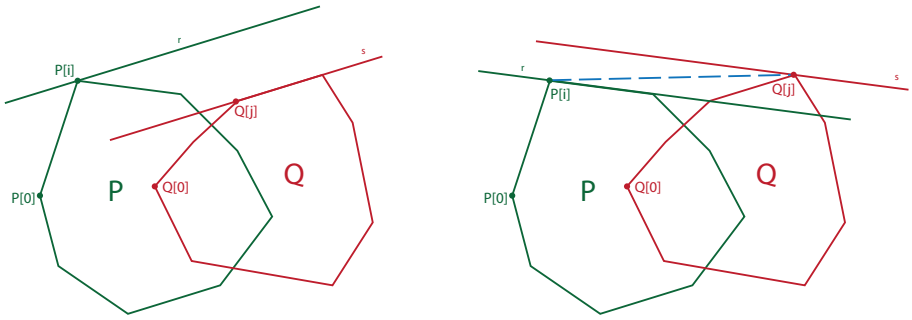
Nájdenie vrcholu v kosáku

Nájdením priečok nájdeme špeciálne mnohouholníky, tzv. kosáky. Takéto mnohouholníky sa skladajú z dvoch konkávnych reťazí, stretávajúcich sa v jednom spoločnom bode a spojené priečkou, ktorú sme našli otáčaním priamok.

My chceme nájsť tento spoločný bod. Budeme hľadať indexy vrcholov v ktorých začínajú pretínajúce sa úsečky. Začneme tak, že budeme posúvať jeden koncový bod priečky po bodoch mnohouholníka,



Obr. 6.5. obe priamky otočíme o menší z dvoch uhlov



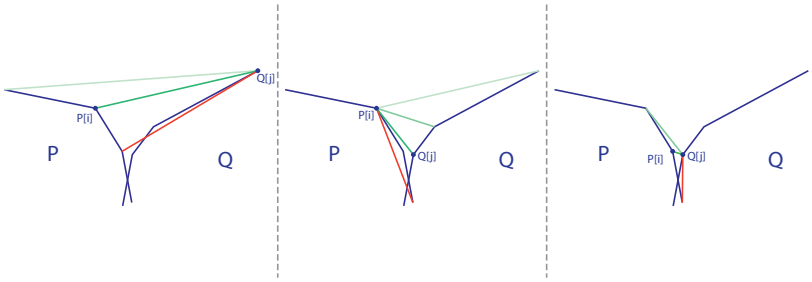
Obr. 6.6. zmenou poradia priamok nájdeme priečku spájajúcu mnohouholníky

kým je úsečka na rovnakej strane oboch reťazí. Takto nájdeme jeden bod mnohouholníka – $P[i]$. Rovnakým postupom, posúvaním druhého konca priečky, pričom jej prvý koncový bod je už bod $P[i]$, nájdeme bod $Q[j]$. Postup opakujeme, kým je spojnica bodov na rovnakých stranách oboch reťazí. Na obrázku č. 4 môžeme vidieť postup nájdenia krajných bodov pretínajúcich sa úsečiek. Potom vieme, že hľadaný prienik hrán mnohouholníkov bude prienik úsečiek $P[i]P[i+1]$ a $Q[j]Q[j-1]$ (obr.5). Nájdeme bod prieniku dvoch úsečiek a uložíme ho do množiny R.: Nech súradnice krajných bodov úsečiek sú: $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$ a bod prieniku nech je P so súradnicami (p_x, p_y). Potom:

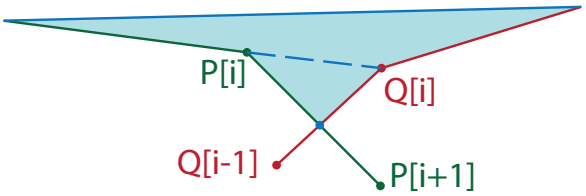
$$p_x = \frac{(x_1y_2-y_1x_2)(x_3-x_4)-(x_1-x_2)(x_3y_4-y_3x_4)}{(x_1-x_2)(y_3-y_4)-(y_1-y_2)(x_3-x_4)}$$

$$p_y = \frac{(x_1y_2-y_1x_2)(y_3-y_4)-(y_1-y_2)(x_3y_4-y_3x_4)}{(x_1-x_2)(y_3-y_4)-(y_1-y_2)(x_3-x_4)}$$

Spolu s bodom si budeme pamätať aj informácie o tom, v ktorom kosci sa bod nachádza a prienikom ktorých dvoch hrán sme ho získali. Takto nájdeme všetky body prieniku hraníc mnohouholníkov P, Q.



Obr. 6.7. hľadanie pretínajúcich sa úsečiek v kosci



Obr. 6.8. kosec s bodmi pretínajúcich sa úsečiek

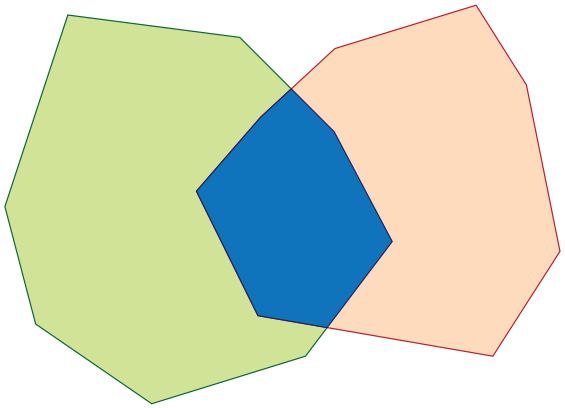
Vybudovanie riešenia

Teraz potrebujeme so získaných výsledkov poskladať mnohouholník, ktorý sa rovná $P \cap Q$.

Začneme v jednom z bodov z R. Nasledujúci bod bude jeden z koncových bodov úsečiek z ktorých sme určili prienik. Budeme pridávať body z jedného mnohouholníka, musíme však dávať pozor na to, že z P pridávame body v smere orientácie, ale z Q proti smeru orientácie. Postupným pridávaním bodov narazíme na bod, ktorý sa nachádza v novom kosci. Vtedy posledný bod zrušíme, pridáme bod z R, ktorý je z príslušného nového kosca a ďalej budeme pridávať body z druhého mnohouholníka.

Skončíme, keď sa vrátime do bodu z R, z ktorého sme začali. Výsledný mnohouholník tvoria alternujúce časti mnohouholníkov P, Q, oddelené bodmi z R (obr 6).

V celom riešení predpokladáme, že existuje prienik mnohouholníkov a nie je to bod ani úsečka, môže však nastať, že jeden mnohouholník



Obr. 6.9. prienik dvoch konvexných mnohouholníkov

leží celý v druhom. Test na takýto špeciálny prípad bude jednoduchý: otáčaním priamok nenájdeme žiadnu priečku. Otázkou je, či chceme o takomto prieniku uvažovať, lebo sa nepretínajú hranice. Záleží od konkrétnej aplikácie.

Odhad zložitosti:
Nájdenie priečok: $\mathcal{O}(m+n)$, kde m,n sú počty vrcholov mnohouholníkov.
Nájdenie bodu prieniku v kosci: $\mathcal{O}(k+l)$, kde k,l počty bodov reťazí v kosci.
Nájdenie prieniku dvoch úsečiek: $\mathcal{O}(1)$.
Vybudovanie výsledného mnohouholníka vyžaduje tiež lineárny počet krokov.

Zložitosť algoritmu je teda $\mathcal{O}(m+n)$. □

Cvičenie 6.2 (30 bodov). *V rovine je daných n úsečiek. Nájdite algoritmus, ktorý zistí, či sú úsečky po dvoch disjunktné a určte jeho zložitosť.*

Riešenie (Martin Škorupa, 02.12.2008):

Algoritmus otestuje každé dve úsečky či sa nepretnú, na to využijeme trochu upravený algoritmus z cvičenia 2.2. Na zistenie či sa úsečky pretínajú, ak ležia na jednej priame využijeme skalárny súčin vektorov. Ak sa úsečky pretínajú musí aspoň jeden vrchol úsečky ležať na druhej úsečke.

Skalárny súčin vektorov $\bar{v}.\bar{u} = \mid \bar{v} \mid.\mid \bar{u} \mid.\cos(\angle\ \bar{v}\ \bar{u})$, teda ak $\bar{v}$, $\bar{u}$ ležia na jednej priamke tak znamienko +/- rozhoduje o tom či majú rovnaký/opačný smer, ak $\bar{v}.\bar{u} = 0$ tak aspoň jeden z vektorov $\bar{v}$, $\bar{u}$ je nulový.

```
skalar(v1, v2, u1, u2) {
    return v1 * u1 + v2 * u2;
}

prienik2useciek(a1, a2, u1, u2, b1, b2, v1, v2) {
    double c1 = u2 * a1 - u1 * a2;
    double c2 = v2 * b1 - v1 * b2;
    double d = u1 * v2 - u2 * v1;
    if (d == 0) { \\ sú rovnobežné
        if (-u2 * b1 + u1 * b2 + c1 == 0) { \\ležia na priamke
            \\ A leží medzi CD (CA, DA majú opačný smer)
            if (skalar( b1 - a1,
                        b2 - a2,
                        (b1 + v1) - a1,
                        (b2 + v2) - a2) <= 0)
                return true;
            \\ B leží medzi CD (CB, DB majú opačný smer)
            if (skalar( b1 - (a1 + u1),
                        b2 - (a2 + u2),
                        (b1 + v1) - (a1 + u1),
                        (b2 + v2) - (a2 + u2)) <= 0)
                return true;
            \\ C leží medzi AB (AC, BC majú opačný smer)
            if (skalar( a1 - b1,
                        a2 - b2,
                        (a1 + u1) - b1,
                        (a2 + u2) - b2) <= 0)
                return true;
            \\ D leží medzi AB (AD, BD majú opačný smer)
            if (skalar( a1 - (b1 + v1),
                        a2 - (b2 + v2),
                        (a1 + u1) - (b1 + v1),
                        (a2 + u2) - (b2 + v2)) <= 0)
                return true;
            return false;
        } else \\ neležia na jednej priamke
            return false;
    } else {
        double r1 = (u2 * b1 - u1 * b2 - c1)/(-d);
        if (r1 > 1 || r1 < 0) \\ prienik je mimo úsečky
            return false;
        double r2 = (v2 * a1 - v1 * a2 - c2)/(-d);
        if (r2 > 1 || r2 < 0) \\ prienik je mimo úsečky
```

```
            return false;
        return true;
    }
}

suUseckyDisjunktne(u) {
    for (int i = 0; i < n; i++)
        for (int j = i+1; j < n; j++)
            if (prienik2useciek(u[i].Ax,
                                u[i].Ay,
                                u[i].Bx - u[i].Ax,
                                u[i].By - u[i].Ay,
                                u[j].Ax,
                                u[j].Ay,
                                u[j].Bx - u[j].Ax,
                                u[j].By - u[j].Ay))
                return false;
    return true;
}
```

Algoritmus porovnáva každé dve úsečky, teda zložitosť bude $\mathcal{O}(\mathbf{n^2})$.

□

Riešenie (Lukáš Tencer, 4.11.2008):

Na riešenie nášho problému použijeme algoritmus zametacej priamky. Ako vstup predpokladáme úsečky ktoré sú určené koncovými bodmi. Tieto úsečky musia byť orientované, to vieme dosiahnuť, bez ujmy na zložitosti, pri prvom kroku algoritmu, keď usporiadavame body úsečiek. Ak by sme chceli jednoduchú odpoveď na otázku či sú úsečky po dvoch disjunktné môžeme skončiť algoritmus po nájdení prvého priesečníku úsečiek, avšak vieme ho jednoducho rozšíriť na nájdenie všetkých priesečníkov úsečiek v rovine.

Povedzme, že naša zametacia priamka bude rovnobežná s osou y. Budeme potrebovať dve pomocné štruktúry a to prioritnú frontu F, ktorá bude obsahovať body zoradené podľa x-ovej súradnice a ku každému bodu budeme mať informáciu o jeho type, a teda či je začiatočný alebo koncový bod úsečky. Druhá pomocná štruktúra bude obsahovať úsečky v poradí v akom pretínajú zametaciu priamky podľa y-ovej súradnice. Na jej realizáciu použijeme vyvážený strom S (scapegoat tree, splay tree), ktorý podporuje následné operácie v čase $\mathcal{O}(\log n)$. Potrebné operácie sú vlož prvok, vyber prvok, nájdi predchodcu/nasledovníka prvku. Pre podrobný popis štruktúr pozri knihu *Jeff Erickson: Jeff Erickson's Algorithms Course Materials.*

Teraz si popíšeme algoritmus hľadania prieniku:

(1) Zober body úsečiek a zoraď ich podľa x-ovej súradnice. Naplň nimi zásobník F.
(2) Zober prvok $x \in F$.

(3) Ak je x začiatočný bod úsečky X tak pridaj X do stromu S. Nájdi úsečku u_2 pod ňou a u_1 nad ňou a zisti či sa pretínajú s úsečkou X. Ak áno tak skonči a vráť zápornú odpoveď. Pozri obr. 6.10.
(4) Ak je x koncový bod úsečky X tak nájdi úsečku pod a nad ňou a zisti či sa pretínajú. Ak áno tak skonči a vráť zápornú odpoveď. Odstráň X zo stromu S. Pozri obr. 6.11.
(5) Ak je F prázdne tak skonči a vráť kladnú odpoveď. Ak F nie je prázdne tak sa vráť na krok 2.

```
boolean suDisjunktne ( priamky P ) {
  zasobnikF.vlozBody( p );
  stromS.inicializuj();
  while ( zasobnikF.nieJePrazndy ){
    x = zasobnikF.hornyPrvok();
    X = priamka( x );
    if ( jeKoncovyBod( x , X ) ){
      u1 = stromS.prvokNad( X );
      u2 = stromS.prvokPod( X );
      if ( pretinajuSa( u1 , u2 ) )
        return false;
      else stromS.odstran( X );
    }
    else if ( jeZaciatocnyBod( x , X ) ){
      stromS.pridaj( X );
      u1 = stromS.prvokNad( X );
      u2 = stromS.prvokPod( X );
      if ( pretinajuSa( u1 , x ) alebo pretinajuSa( u2 , x ) )
        return false;
    }
  }
  return true;
}
```

Algoritmus si teraz ilustrujme na príklade. Pozri obr. 6.12. Možnosť rozšírenia na nájdenie všetkých prienikov tiež možno nájsť v hore spomínanej knihe. Algoritmus sa rozšíri o pridanie priesečníku medzi body ktoré treba spracovať. A spracovanie udalosti pokiaľ je bod, ktorý spracuvávame priesečníkom.

Aby sme ukázali, že algoritmus nevynechá žiadny priesečník analyzujme prípady kedy kontrolujeme či úsečky nemajú priesečník. Pri vkladaní môže mať vkladaná úsečka v priesečník len s úsečkou u_1 nad ňou a u_2 pod ňou keďže bude ležať vo výseku nimi určenom, lebo sú to k nej najbližšie úsečky. Pozri obr. 6.13. Pri odstraňovaní úsečky v sa môžu pretínať len úsečky u_1 nad ňou a u_2 pod ňou vzhľadom na to, že u_1 alebo u_2 mohla začínať až po v a tak v doteraz bránilo nájdeniu ich priesečníka. Navyše koncový bod v je jediná udalosť medzi u_1 a u_2 z ktorej možno vidieť ich priesečník. Pozri obr. 6.14. Analýzou týchto

dvoch prípadov spracovávania udalostí a oprúc sa o koncept zametacej priamky, že časť naľavo je už upratená (nie sú tam žiadne priesečníky) a tú napravo spracovávame vieme vylúčiť existenciu priesečníku, ktorý by sme nenašli.

Počiatočné zoradenie okrajových bodov úsečiek nám zaberie čas $\mathcal{O}(n \log n)$. Spracovávanie každého bodu zahrňuje prácu so stromom S, na čo potrebujeme $\mathcal{O}(\log n)$ a spočítanie priesečníku v čase $\mathcal{O}(1)$. Čiže celková zložitosť bude $\mathcal{O}(n \log n)$. □

Riešenie (Martina Bátorová, 14.12.2008): Označme množinu úsečiek ako $\mathcal{U} = \{u_1, \ldots, u_n\}$, výrazom $u_i = \langle a_i, b_i \rangle$ budeme rozumieť, že úsečka u_i má začiatočné body a_i, b_i. Zadanie potom môžeme sformulovať tak, že hľadáme bod $q \in \mathbb{E}^2$ taký, že existujú (aspoň) dve úsečky $u_i, u_j, i \neq j$, že $q \in u_i \cap u_j$. Na riešenie použijeme zametanie vertikálnou priamkou λ, čo nám umožní problém rozhodnúť v čase $\mathcal{O}(n \log n)$.

Zoznam udalostí:
začiatočné a koncové vrcholy úsečiek, body prieniku úsečiek.

Stav zametacej priamky:
úsečky preťaté zametacou priamkou a usporiadané podľa y-ovej súradnice tohto prieniku. Ako dátovú štruktúru uvažujme binárny strom, ktorý dokáže vkladať, odstraňovať a hľadať susedov v čase $\mathcal{O}(\log n)$.

Spracovanie udalostí:

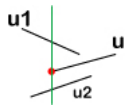
- začiatočný bod: vložíme úsečku u_i do stavu λ a otestujeme na prienik so susednými úsečkami; susednosť máme zadanú vďaka usporiadaniu podľa prieniku s λ. Do úvahy berieme iba body prieniku nachádzajúce sa *napravo* od λ.
- bod prieniku: končíme algoritmus, výsledkom je `false`, tea úsečky *nie sú* po dvoch disjunktné.
- koncový bod: vynecháme zo stavu λ úsečku s týmto koncovým vrcholom. Môže sa stať, že v stave vzniknú nové susediace úsečky. Tieto otestujeme na spoločný prienik.

V prípade, že stav úsečky je prázdny a zároveň sme nenašli prienik, výsledkom je `true`, teda úsečky *sú* po dvoch disjunktné.

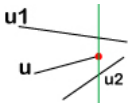
Zložitosť algoritmu:

- začiatočný bod: keďže máme n úsečiek, udalosť nastáva $\mathcal{O}(n)$-krát. Jedno vloženie trvá $\mathcal{O}(\log n)$ operácií. Tento krok spolu teda $\mathcal{O}(n \log n)$.
- bod prieniku: prienik dvoch úsečiek vieme vypočítať v čase $\mathcal{O}(1)$.
- koncový bod: rovnako ako v prípade začiatočného vrchola potrebujeme $\mathcal{O}(n \log n)$, pretože $\mathcal{O}(n)$-krát vymazávame úsečku, resp. jej začiatočný vrchol.

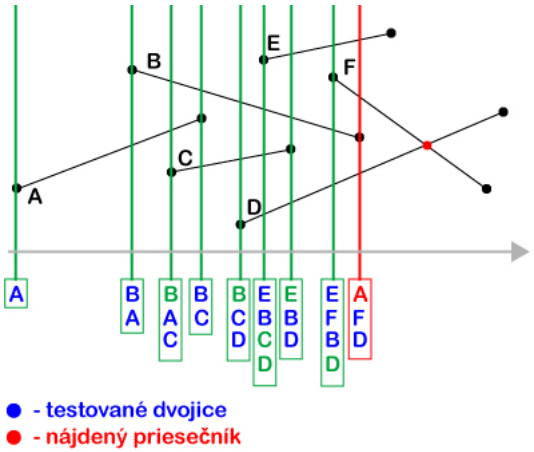
Spolu je časová zložitosť $\mathcal{O}(n \log n)$, pamäťová $\mathcal{O}(n)$ (potrebujeme uložiť n úsečiek, susedov je nanajvýš $(n-1)$). □



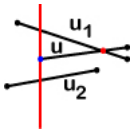
OBR. 6.10. Spracovávanie začiatočného bodu úsečky.



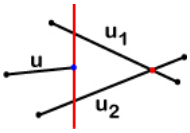
OBR. 6.11. Spracovávanie koncového bodu úsečky.



OBR. 6.12. Príklad zametacieho algoritmu na hľadanie priesečníku.



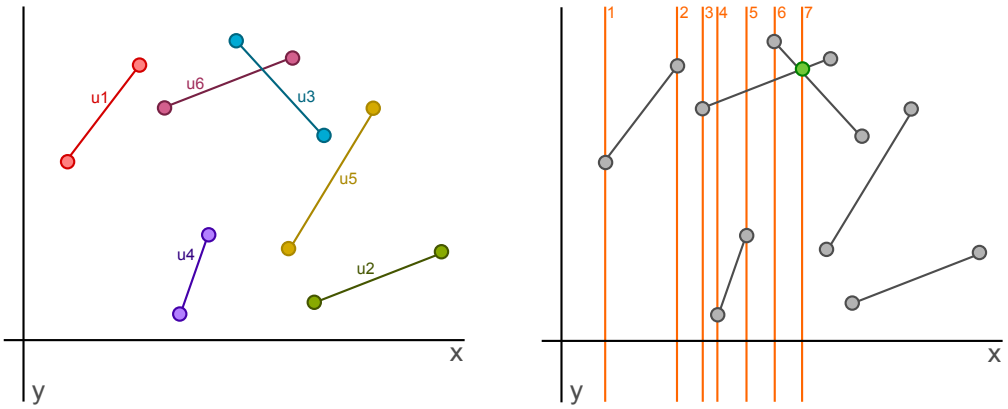
OBR. 6.13. Nájdenie priesečníku pri spracovávaní začiatočného bodu úsečky.



OBR. 6.14. Nájdenie priesečníku pri spracovávaní koncového bodu úsečky.

CVIČENIE 6.3 (40 bodov). *V rovine je daných n úsečiek. Zostavte algoritmus, ktorý zistí, či existuje priamka pretínajúca všetky tieto úsečky a ak áno, nájdite ju.*

Riešenie (Lukáš Apalovič, 23.12.2008): V rovine máme úsečky $u_1, u_2, \ldots, u_n$. Koncové body úsečky u_i si označíme $p_{i,1}$ a $p_{i,2}$. Algoritmus bude testovať pre všetky možné dvojice koncových bodov úsečiek $(p_{i,k}, p_{j,l}), i \neq j; i, j \in \{1, \ldots, n\}; k, l \in \{1, 2\}$, či priamka $p_{i,k}p_{j,l}$ pretína všetky úsečky.



Obr. 6.15. Na obrázku vľavo je vstup algoritmu - konečný počet úsečiek v rovine. Na obrázku vpravo je niekoľko krokov algoritmu: v polohách $1, 3, 4, 6$ zametacej priamky vkladáme úsečky u_1, u_6, u_4, u_3 do stavu. V polohách $2, 5$ odstraňujeme úsečky u_1, u_4. Na prienik testujeme dvakrát: v kroku 4 máme v stave úsečky u_4, u_6, ich prienik je prázdny. V kroku 7 máme v stave úsečky u_6, u_3, ich prienik je neprázdny, takže algoritmus končí s odpoveďou `false`.

Na zistenie, či priamka pretína úsečku použijeme poupravenú metódu z príkladu 6.2. Priamku máme danú bodom $p_{i,k} = a$ a vektorom $p_{j,l} - p_{i,k} = u$ a nejakú úsečku u_r bodom $p_{r,1} = b$ a vektorom $p_{r,2} - p_{r,1} = v$.

```
Prienik(a1, a2, u1, u2, b1, b2, v1, v2) {
    c1 = u2 * a1 - u1 * a2;
    c2 = v2 * b1 - v1 * b2;
    d = u1 * v2 - u2 * v1;
    if (d==0) {     //rovnobežné
        if (-v2 * a1 + v1 * a2 + c2 == 0) //úsečka leží na
priamke
            return true;
        else return false;
    }
    else {
        r = (u2 * b1 - u1 * b2 - c1)/(-d);
        if (r > 1 || r < 0) //prienik je mimo úsečky
            return false;
        else return true;
    }
}
PriamkaPretinajucaUsecky() {
    for(i=1; i<=n; i++)
        for(j=1; j<=n; j++)
```

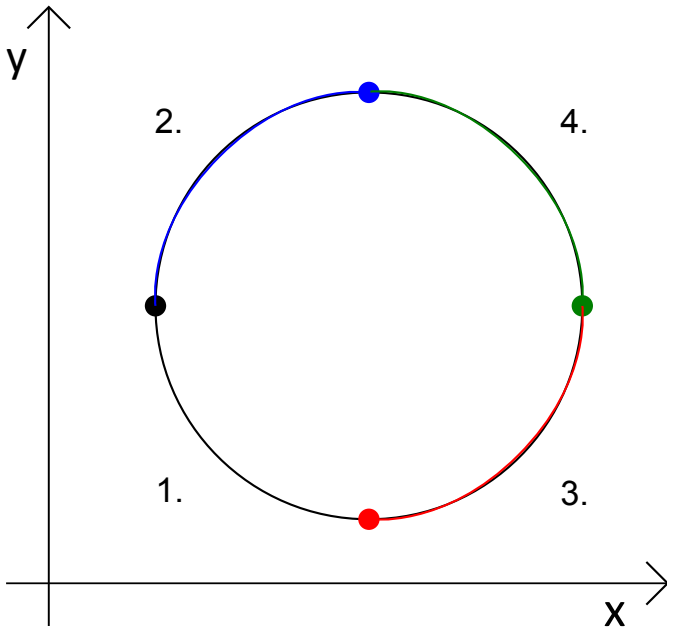
```
            if(i≠j) {
                //testujeme priamku p_{i,1}p_{j,1}
                pretinaVsetky = true;
                a = p_{i,1};
                u = p_{j,1} − p_{i,1};
                for(k=1; k<=n; k++) {
                    b = p_{k,1};
                    v = p_{k,2} − p_{k,1};
                    if(!Prienik(a1,a2,u1,u2,b1,b2,v1,v2))
                        pretinaVsetky = false;
                }
                if(pretinaVsetky)
                    return Line(p_{i,1}, p_{j,1});
                //testujeme priamku p_{i,2}p_{j,1}
                pretinaVsetky = true;
                a = p_{i,2};
                u = p_{j,1} − p_{i,2};
                for(k=1; k<=n; k++) {
                    b = p_{k,1};
                    v = p_{k,2} − p_{k,1};
                    if(!Prienik(a1,a2,u1,u2,b1,b2,v1,v2))
                        pretinaVsetky = false;
                }
                if(pretinaVsetky)
                    return Line(p_{i,2}, p_{j,1});
                //analogicky testujeme aj pre p_{i,1}p_{j,2} a p_{i,2}p_{j,2}
                //...
            }
    return false;
}
```

Časová náročnosť algoritmu je $\mathcal{O}(n^3)$, pretože vyberáme všetky možné kombinácie koncových bodov úsečiek ($\mathcal{O}(n^2)$) a pre každú takúto dvojicu bodov hľadáme prienik priamky nimi určenej so všetkými úsečkami. □

Cvičenie 6.4 (40 bodov). *Nech S je množina obsahujúca n kružníc v rovine. Opíšte zametací algoritmus na vyhľadanie všetkých prienikov medzi kružnicami. Odhadnite zložitosť zostaveného algoritmu. Pokúste sa upraviť algoritmus tak, aby jeho zložitosť bola $\mathcal{O}((n+k)\log n)$.*

Riešenie (Barbora Gallusová, 27.12.2009): Každú kružnicu si rozdelíme na štyri oblúky. Nech má stred kružnice súradnice $[x, y]$ a polomer kružnice je r, potom kružnicu rozdelíme na štyri oblúky (6.16):

(1) oblúk spájajúci body $[x-r, y]$ a $[x, y-r]$, okrem bodu $[x, y-r]$
(2) oblúk spájajúci body $[x-r, y]$ a $[x, y+r]$, okrem bodu $[x-r, y]$
(3) oblúk spájajúci body $[x, y-r]$ a $[x+r, y]$, okrem bodu $[x+r, y]$
(4) oblúk spájajúci body $[x, y+r]$ a $[x+r, y]$, okrem bodu $[x, y+r]$



Obr. 6.16. Každú kružnicu si rozdelíme na štyri oblúky.

Výsledný algoritmus bude využívať techniku zametacej priamky. Zametacia priamka bude rovnobežná s osou y a bude sa pohybovať v kladnom smere osi x. Udalosťami zametacej priamky budú:

(1) začiatok oblúka
(2) koniec oblúka
(3) priesečník dvoch oblúkov (ak sa dva oblúky pretínajú v dvoch priesečníkoch, udalosťami budú oba priesečníky)

Udalosti budeme udržiavať v prioritnej fronte. Vhodná štruktúra je napríklad halda. Potom časová zložitosť operácie pridania nového prvku a tiež aj operácie odobratia najmenšieho prvku bude $O(\log n)$, ak halda obsahuje $O(n)$ prvkov. Pamäťová zložitosť haldy - dátovej štruktúry je lineárna od počtu prvkov.

Kritériom pre porovnanie, ktorá z dvoch udalostí nastane skôr, je x-ová súradnica bodu. Pri rovnosti ďalej rozhoduje polomer a číslo oblúka podľa predošlého očíslovania.

Stav zametacej priamky tvorí postupnosť bodov (ich y-ových súradníc), z ktorých každý patrí inému oblúku. Pritom si pamätáme, ktorý bod patrí ku ktorému oblúku. Body udržiavame usporiadané podľa stúpajúcej y-ovej súradnice vo vyváženom binárnom strome. Takto dosiahneme, že každá z operácií (vloženie prvku, vyhľadanie prvku, vyhľadanie najbližšieho menšieho a najbližšieho väčšieho prvku) bude mať časovú zložitosť $O(\log n)$.

Spracovanie udalostí:

(a) **Začiatok oblúka** Vložíme začiatočný bod oblúka na správne miesto medzi ostatné body na zametacej priamke (vložíme prvok do binárneho stromu). Skontrolujeme, či sa práve pridaný oblúk pretína s niektorým zo susedných dvoch oblúkov. Susedné dva oblúky sú práve tie, ktoré sú na zametacej priamke reprezentované bodmi, ktoré susedia s pridaným bodom. Ak nájdeme prieniky, pridáme ich do zoznamu udalostí zametacej priamky (vložíme prvok/prvky do haldy).

(b) **Koniec oblúka** Zo zametacej priamky odoberieme bod prislúchajúci oblúku, ktorý je aktuálnou udalosťou ukončený. Skontrolujeme, či sa vzájomne pretínajú oblúky, susediace s práve odobratým oblúkom. Ak nájdeme prieniky, pridáme ich do zoznamu udalostí zametacej priamky.

(c) **Priesečník dvoch oblúkov** V rámci zametacej priamky "vymeníme" body prislúchajúce oblúkom, ktoré sa pretli. Operácia výmeny zahŕňa odobratie pôvodných dvoch bodov, reprezentujúcich pretínajúce sa oblúky a pridanie dvoch nových bodov. Ako y-ovú súradnicu nových bodov zvolíme y-ovú súradnicu priesečníka, zväčšenú resp. zmenšenú o vhodné malé číslo.

Takto získa každý z vymenených oblúkov nového suseda. Skontrolujeme nové dvojice oblúkov na vzájomné priesečníky. Ak nájdeme prieniky, pridáme ich do zoznamu udalostí zametacej priamky.

V čom by mohli nastať problémy:

- oblúky prislúchajúce tej istej kružnici nesmú mať priesečník. Preto je vyššie uvedené, ktorému oblúku patrí ktorý hraničný bod.
- každý priesečník pridávame iba raz. V určitých prípadoch sa môže stať, že konkrétnu dvojicu oblúkov testujeme na priesečníky počas behu algoritmu viackrát. Lepšia ako pamätať si, ktoré dva oblúke sme už testovali na vzájomné priesečníky, je pri vkladaní udalosti - priesečníku skontrolovať, či sme tento priesečník už nevložili.

Dôkaz, že algoritmus nájde práve všetky priesečníky kružníc. Vo vhodnom okolí ľubovoľného priesečníka dvoch oblúkov sú priesečníky oblúkov so zametacou priamkou vždy susedné. Na druhej strane, algoritmus skontroluje všetky dvojice oblúkov, ktoré sú aspoň v jednom momente považované za susedné: pri vložení nového oblúka vzniknú dve nové susednosti, pri odobratí oblúka vnikne jedna nová susednosť, pri priesečníku vďaka "výmene" oblúkov vzniknú dve nové susednosti. Algoritmus vždy preverí, či sa dva oblúky v takto vzniknutých susednostiach pretínajú.

Časová zložitosť algoritmu:

(a) Inicializácia haldy obsahujúcej všetky udalosti typu 1) a 2) má časovú zložitosť $O(n \log n)$:

Počet oblúkov je $4n$, každý oblúk má začiatočný a koncový bod. Spolu teda treba do haldy vložiť $O(n)$ prvkov, vloženie jedného prvku má časovú zložitosť $O(\log n)$.

(b) Nech počet priesečníkov kružníc je k. Časová zložitosť pridania všetkých priesečníkov do haldy je $O(k \log n)$:

Počas spracovávania udalostí pridáme do haldy k prvkov, každý v zložitosti $O(\log(n+k))$. Keďže však $k \leq 2\binom{n}{2} = O(n^2)$, zložitosť pridania jedného priesečníka je vlastne $O(\log(n^2 + n)) = O(\log n^2) = O(2 \log n) = O(\log n)$.

(c) Spracovanie udalostí má časovú zložitosť $O((n + k) \log n)$:

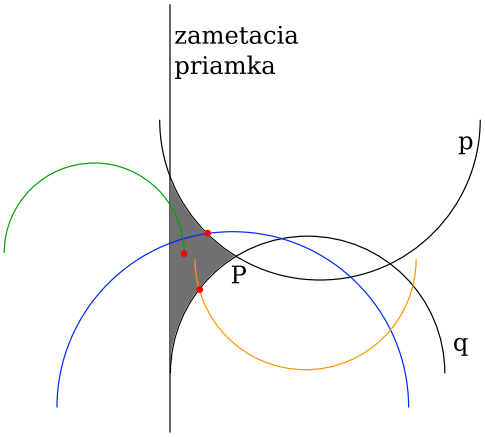
Spracovanie jednej udalosti vyžaduje konštantný počet operácií na vyváženom binárnom strome. Každá z operácií má logaritmickú časovú zložitosť, v závislosti od počtu bodov na zametacej priamke, ktorých je najviac $O(n)$. Teda spracovanie jednej udalosti má časovú zložitosť $O(\log n)$. Počet všetkých udalostí je $8n + k = O(n + k)$.

Celková časová zložitosť algoritmu je teda $O((n + k) \log n)$.

Pamäťová zložitosť haldy aj binárneho stromu je lineárna od počtu prvkov. Teda celková pamäťová zložitosť algoritmu je $O(n+k)+O(n) = O(n + k)$. □

Riešenie (Martin Manduch, 3.1.2010): Pri hľadaní priesečníkov úsečiek pomocou zametacej priamky (cvičenie 6.2) sme vychádzali z predpokladu, že ak sa dve úsečky pretínajú, tak existuje vertikálna – zametacia priamka, na ktorej sú prieniky týchto úsečiek s ňou navzájom susediace. Ukázalo sa, že nemusíme testovať všetky pozície vertikálnej priamky, ale len tie, keď sa mení poradie prienikov úsečiek s ňou. Zametacou priamkou sme postupovali zľava doprava. Poradie prienikov so zametacou priamkou sa menilo len v tých x-ových pozíciách zametacej priamky, keď x-ová súradnica bola x-súradnicou koncového bodu jednej z úsečiek, alebo bola x-súradnicou prieniku nejakých dvoch úsečiek, ktorý sme už predtým našli.

Pre hľadanie prienikov kružníc použijeme analogický spôsob. Kvôli prehľadnosti budeme predpokladať, že žiadne dve kružnice nezačínajú, nekončia a ani sa nepretínajú v rovnakej x-ovej súradnici. Najprv si ale každú kružnicu rozdelíme na dve polkružnice horizontálnou priamkou prechádzajúcou cez stred kružnice. Hľadanie prienikov kružníc nahradíme hľadaním prienikov polkružníc. Použijeme dve štruktúry. Prvou bude x-štruktúra, ktorá obsahuje pozície zametacej priamky, v ktorých sa môže meniť poradie prienikov s polkružnicami. V y-štruktúre budú polkružnice usporiadané podľa y-ovej súradnice prieniku so zametacou priamkou. y-štruktúru budeme reprezentovať vyváženým binárnym vyhľadávacím stromom, x-štruktúra bude prioritná fronta.



Obr. 6.17. Pozícia zametacej priamky. Šedou farbou je znázornená oblasť S.

Ukážme, že ak existuje prienik dvoch polkružníc p, q, nastane taká pozícia zametacej priamky, kde budú tieto polkružnice v y-štruktúre vedľa seba. Nech P je ľavý bod prieniku $p \cap q$. Ak pri vložení druhej z p, q do y-štruktúry, nie sú p, q vedľa seba, pokračujeme v hľadaní. Uvažujme o všetkých polkružniciach, ktoré začínajú, končia alebo prechádzajú oblasťou S vyznačenou na obrázku 6.17. Vytvorme množinu bodov M. Ak nejaká polkružnica má koncový bod v S, zaradíme ho do M. Ak nejaká polkružnica začína v S alebo ňou prechádza, tak do M dáme body prieniku tej polkružnice a častí p, q, ktoré ležia v S (pozri obr. 6.17). Najväčšia x-ová súradnica bodov množiny M je hľadaná pozícia zametacej priamky.

Algoritmus

```
1 HladaniePriesecnikovKruznic()
2 {
3    vlož koncové body polkružníc do x-štruktúry
4    while(x-štruktúra nie je prázdna)
5    {
6        vyber bod P s minimálnou x súradnicou z x-štruktúry
7        odstráň P z x-štruktúry
8        if( P je ľavý koncový bod polkružníc p_j, p_{j+1}) //p_j je nad p_{j+1}
9        {
10           vlož p_j, p_{j+1} do y-štruktúry tak, aby v nej p_j bola nad p_{j+1}
11           nech p_i, p_k sú susedia p_j v y-štruktúre, p_i zhora, p_j zdola
12           vlož p_i ∩ p_j a p_k ∩ p_{j+1} do x-štruktúry
13           zároveň vynechaj p_i ∩ p_k
14       }
15       if( P je pravý koncový bod polkružníc p_j, p_{j+1})
```

```
16      {
17          nech p_i, p_k sú susedia p_j, p_{j+1} v y-štruktúre, p_i zhora, p_j
zdola
18          vynechaj p_j, p_{j+1} z y-štruktúry
19          vlož p_i ∩ p_k do x-štruktúry, ak je napravo od zametacej
priamky
20      }
21      if( P∈ p_i ∩ p_j) //p_j je nad p_i v y-štruktúre
22      {
23          zameň susediace p_i, p_j
24          nech p_h, p_k sú susedia p_i, p_j v y-štruktúre, p_h zhora, p_k
zdola
25          vlož p_h ∩ p_i a p_k ∩ p_j do x-štruktúry, ak sú napravo
26              od zametacej priamky
27          zároveň vynechaj z x-štruktúry p_h ∩ p_j a p_k ∩ p_i
28          zapíš P do výstupného zoznamu
29      }
30  }
31}
```

Časová zložitosť
Operácie na y-štruktúre aj na x-štruktúre trvajú $\mathcal{O}(\log(n))$. Na zabezpečenie $\mathcal{O}(\log(n))$ zložitosti pre x-štruktúru slúžia riadky 13 a 27, v ktorých odstraňujeme prieniky, ktoré budú neskôr pridané. Hlavný cyklus sa opakuje $n+k$ krát. Preto je pamäťová zložitosť $\mathcal{O}((n+k)\log(n))$.

Pamäťová zložitosť pre x-štruktúru aj pre y-štruktúru je $\mathcal{O}(n)$. Aby mala x-štruktúra pamäťovú zložitosť $\mathcal{O}(n)$, zabezpečili už spomínané riadky 13 a 27. □

Riešenie (Alojz Kováčik, 28.12.2009):

Pre každú kružnicu nájdeme bod s najmenšou a najväčšou x-ovou a y-onovou súradnicou. Tieto body nám rozdelia každú kružnicu na štyri štvrťkružnice. Ďalej pracujeme s týmito štvrťkružnicami ako so samostatnými celkami. Využijeme podobný spôsob, ako v prípade zisťovania prienikov úsečiek v rovine.

Budeme potrebovať dve pomocné štruktúry a to prioritnú frontu F, ktorá bude obsahovať body zoradené podla x-ovej súradnice a ku každému bodu budeme mať informáciu o jeho type, a teda či je začiatočný, koncový bod štvrťkružnice alebo je priesečníkom dvoch štvrťkružníc.

Druhá pomocná štruktúra bude obsahovať štvrťkružnice, usporiadané vzhľadom na y-onovú súradnicu ich prieniku so zametacou priamkou. Ďalej bude obsahovať informáciu o tom, ktorej kružnici patrí Na

realizáciu tejto štruktúry použijeme vyvážený strom S, ktorý podporuje následné operácie v čase $\mathcal{O}(\log n)$. Potrebné operácie sú vlož prvok, vyber prvok, nájdi predchodcu/nasledovníka prvku.

Algoritmus

(a) Najprv usporiadame začiatočné a koncové body všetkých štvrťkružníc podľa x-ovej súradnice a vložíme do fronty F. Celkový počet bodov je $8n$.

(b) Kým fronta F nie je prázdna, opakuj:

(c)
- Vyber prvok $x \in F$.
- Ak x je ľavý koncový bod štvrťkružnice X, pridaj X do stromu S. Nech A, B sú susedné štvrťkružnice X v S. Skontroluj, či sa krivka nepretína s týmito susednými krivkami. Ak áno a ak susedné krivky nepatria tej istej kružnici, vlož dané prieniky usporiadané podľa x-ovej súradnice do fronty. Zároveň vynechaj body prieniku $(A \cap B)$ z F. Pokračuj krokom 2.
- Ak x je pravý koncový bod polkružnice X, odstráň X zo stromu S. Nech A, B sú susedné štvrťkružnice, ktoré nepatria tej istej kružnici X v S. Vlož body prieniku $A \cap B$ do fronty F ak ležia napravo od zametacej priamky. (Tento krok je potrebný, keďže po odobratí X môže nastať situácia, kedy sa A a B stanú susedmi.) Pokračuj krokom 2.
- Ak x je prienik dvoch štvrťkružníc A a B, zameň tieto štvrťkružnice. Nech ďalej C a D sú susedia A a B v strome S. Vlož body patriace prienikom $(B \cap C)$ a $(A \cap D)$ do fronty F, ak ležia napravo od zametacej priamky. Zároveň vynechaj body prienikov $(A \cap C)$ a $(B \cap D)$ z F. Zapíš bod x do výstupného zoznamu.

Správnosť algoritmu vyplýva z faktu, že každý bod, ktorý je koncovým bodom štvrťkružnice alebo priesečníkom sa zapíše do fronty práve raz. Bod priesečníku sa potom zapíše do vystupného zoznamu práve raz a už sa vo fronte neobjaví.

Časová zložitosť algoritmu je $\mathcal{O}((n+k)\log n)$, pretože hlavný cyklus sa opakuje (n + k) krát a operácie na aktualizáciu dátovej štruktúry nevyžadujú viac ako $\mathcal{O}(\log(n))$ času. □

Cvičenie 6.5 (50 bodov). *Nech S je množina obsahujúca n disjunktných trojuholníkov. Nájdite $n-1$ úsečiek s nasledujúcimi vlastnosťami:*

- *každá úsečka spája hraničné body dvoch rôznych trojuholníkov z S*
- *vnútra úsečiek sú po dvoch disjunktné a disjunktné aj s trojuholníkmi*

- *spolu spájajú navzájom všetky trojuholníky*

Zostavte zametací algoritmus, ktorý rieši tento problém v čase $\mathcal{O}(n \log n)$. Popíšte použité dátové štruktúry a invariant zametacej priamky(t. j. stav zametacej priamky).

Riešenie (Kiss Gábor, 27.12.2010): Zadané trojuholníky považujme za rovinný graf. Vrcholy trojuholníkov budeme považovať za vrcholy grafu, hrany trojuholníkov za hrany grafu. Pre jednoduchosť predpokladajme, že ani jedna z hrán nie je rovnobežná s osou x (ak to neplatí, môžeme zvoliť vhodnú súradnicovú sústavu, kde to platiť bude). Označme si tieto vrcholy ako v_{ki}, kde $k = 1, \ldots, n$ a $i = 0, 1, 2$. Vrcholy budeme označovať tak, že pre k- ty trojuholník bude v_{k0} vrchol s najmenšou y-ovou súradnicou a v_{k2} vrchol s najväčšou y-ovou súradnicou. Ako prvý krok si utriedime v_{ki}, $k = 1, \ldots, n$ podľa ich y-ovej súradnice od najmenšej po najväčšiu. Keďže budeme používať algoritmus zametacej priamky musíme si definovať množinu udalostí a popísať si, ako sa bude meniť stav zametacej priamky pri jednotlivých udalostiach. Množinou udalostí budú vrcholy grafu. Zametacia priamka bude začínať vo vrchole s minimálnou y-ovou súradnicou. Stavom zametacej priamky bude množina intervalov vytvorená hranami grafu pretínajúcimi zametaciu priamku. Pre každý takýto interval si udržiavame aj vrchol, ktorý mal v danom intervale maximálnu y-ovú súradnicu. Predpokladajme, že máme už spracovanú časť nášho grafu. ďalej si popíšeme, aké možnosti môžu nastať: Ďalej môžu nastať 3 prípady:

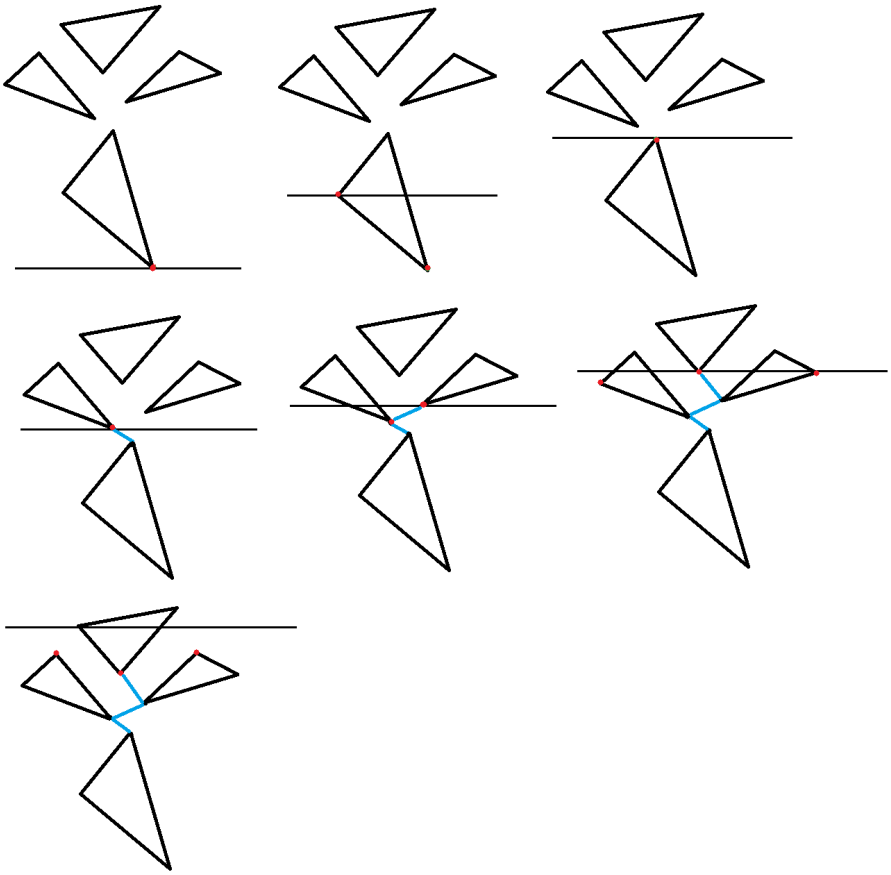
(a) Nasledujúci vrchol, ktorý spracujeme je typu v_{k0} – v tomto prípade postupujeme nasledovne: Tento bod nám rozdelí interval na zametacej priamke na tri časti – interval naľavo od trojuholníka, interval prislúchajúci vnútornej časti trojuholníka a interval napravo od trojuholníka. Ak v_{k0} nie je prvým prvkom utriedenej postupnosti, spojíme ho úsečkou s vrcholom v_{li}, ktorý bol priradený intervalu, do ktorého padol vrchol v_{k0}. Všetkým novovzniknutým intervalom priradíme vrchol v_{k0}.

(b) Nasledujúci vrchol je typu v_{k1}. V tomto prípade nedochádza k vzniku nových intervalov, iba sa budú upravovať vrcholy, ktoré sú jednotlivých intervalom priradené. Presnejšie obom intervalom, ktorým je v_{m1} hraničným bodom ho priradíme.

(c) Poslednou možnosťou je, ak spracúvame vrchol typu v_{k2}. V tomto prípade dochádza k spájaniu troch susedných intervalov do jedného. Novému intervalu priradíme vrchol v_{k2}

Časová analýza: Utriedenie všetkých vrcholov grafu prebehne v čase $\mathcal{O}(n \log n)$. Zametacia priamka bude prechádzať cez všetky vrcholy, ktorých je celkovo $3n$ a bude ich spájať úsečkami – $\mathcal{O}(n)$. Teda celkovo nám bude algoritmus trvať $\mathcal{O}(n \log n)$.

□



OBR. 6.18. Priebeh algoritmu. Červené body označujú vrcholy, ktoré si pamätáme pre jednotlivé intervaly, modré čiary sú pridané úsečky.

CVIČENIE 6.6 (40 bodov). *Nech P_1 je jednoduchý mnohouholník s m vrcholmi s danou triangulóciou. Nech P_2 je konvexný mnohouholník s n vrcholmi. Zistite, v akom čase sa dá vypočítať $P_1 \cap P_2$. V prípade, že ide o čas lepší ako prienik všeobecných mnohouholníkov, svoje tvrdenie aj zdôvodnite. Ak to nejde lepšie, nájdite príklad, kde sa dosahuje zložitosť všeobecného prípadu pre akýkoľvek algoritmus.*

CVIČENIE 6.7 (20 bodov). *Dokážte, že prienik m polrovín je konvexný polygonálny región s najviac m vrcholmi a najviac m hranami.*

Riešenie (Martin Škorupa, 18.11.2008): Nech útvar, ktorý vznikne je $\mathbf{M}$ a priamky $\mathbf{p_1}, \ldots, \mathbf{p_m}$ definujú m polrovín.

- $\mathbf{M}$ je prienik polrovín, teda je polygonálny región.
- dôkaz konvexnosti sporom:
  $\mathbf{A} \in \mathbf{M}$, $\mathbf{B} \in \mathbf{M}$, $\exists\ t \in \langle 0,1\rangle$: $\mathbf{A}.t + \mathbf{B}.(1\text{-}t) \notin \mathbf{M}$
  $\mathbf{a_i}.\mathbf{x_A} + \mathbf{b_i}.\mathbf{y_A} + \mathbf{c_i} \geq 0$
  $\mathbf{a_i}.\mathbf{x_B} + \mathbf{b_i}.\mathbf{y_B} + \mathbf{c_i} \geq 0$
  $\mathbf{a_i}.(t.\mathbf{x_A} + (1\text{-}t).\mathbf{x_B}) + \mathbf{b_i}.(t.\mathbf{y_A} + (1\text{-}t).\mathbf{y_B}) + \mathbf{c_i} < 0$

$t.\mathbf{a_i}.\mathbf{x_A} + (1\text{-}t).\mathbf{a_i}.\mathbf{x_B} + t.\mathbf{b_i}.\mathbf{y_A} + (1\text{-}t).\mathbf{b_i}.\mathbf{y_B} + t.\mathbf{c_i} + (1\text{-}t).\mathbf{c_i} < 0$

$t.(\mathbf{a_i}.\mathbf{x_A} + \mathbf{a_i}.\mathbf{y_A} + \mathbf{c_i}) + (1\text{-}t).(\mathbf{a_i}.\mathbf{x_B} + \mathbf{b_i}.\mathbf{y_B} + \mathbf{c_i}) < 0$ SPOR

- $\mathbf{M}$ je prienikom polrovín, teda $\partial\mathbf{M}$ sa skladá len z častí priamok $\mathbf{p_1}, \ldots, \mathbf{p_m}$. Keďže $\mathbf{M}$ je konvexný, tak z každej priamky $\mathbf{p_i}$ sa v $\partial\mathbf{M}$ bude vyskytovať maximálne jedna časť. Teda hraníc bude maximálne toľko koľko je priamok a teda m.
- Hraníc je maximálne m. Každá hrana môže mať maximálne 2 vrcholy, ale každý vrhol musí patriť 2 hranám, teda vrcholov môže byť maximálne m.

□

Riešenie (Lukáš Tencer, 19.11.2008):

Pracujeme s polrovinami a teda sme v rovine. Polrovina je určená rovnicou priamky a nerovnosťou $\geq$ alebo $\leq$. Teraz si uveďme dôkaz, že prienik takýchto polrovín bude konvexný polygonálny región.

Nech m je počet polrovín. Ak $m = 1$ tak prípad je triviálny a región bude mať 1 hranu a 0 vrcholov. Bude konvexný. Pozri obr. 6.19.

Ak $m = 2$ tak môžu nastať 3 prípady a to: Prienik je prázdna množina. Prienik má 2 hrany a 0 vrcholov. Prienik má 1 vrchol a 2 hrany. Tvarom to bude pás alebo výsek roviny. Oba prípady budú konvexné. Pozri obr. 6.20.

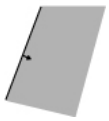
Ak $m = 3$ tak existuje jediný prípad, keď bude mať región m vrcholov a m hrán a to, že prienikom bude trojuholník. V každom inom prípade bude mať región menej hrán alebo vrcholov. Pokiaľ bude oblasť trojuholníkom tak bude konvexná, tak ako i v ďalších prípadoch (pás, otvorený polygonálny región s 3 hranami a 2 vrcholmi). Pozri obr. 6.21.

Teraz predpokladajme, pre m polrovín už máme ich prienik, ktorým je konvexný polygonálny región s m vrcholmi a m hranami. Týmto región bude uzavretý a tak ním bude konvexný mnohouholník.

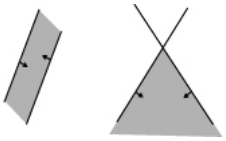
Keď teraz pridáme jednu polrovinu a teda máme $m + 1$ pozrime sa čo sa stane. Najprv vlastnosť konvexnosti. Čiže chceme konvexný útvar predeliť priamkou. Vzniknú nám 2 útvary a oba budú konvexné, takže vlastnosť konvexity sa nám pri pridaní polroviny nezmení. Ďalej chceme ukázať, že sa nám počet vrcholov a hrán zväčší maximálne o 1. Počet hrán a vrcholov sa bude meniť len pokiaľ bude priamka, označme si ju l, polroviny pretínať už vytvorený región.

Povedzme, že ho pretína v 2 po sebe nenasledujúcich hranách. Potom nám pribudnú 2 nové vrcholy a minimálne 2 ubudnú, takže počet nových vrcholov nestúpne nad m. Pribudne nám 1 hrana a aspoň 1 ubudne takže ani počet hrán nestúpne nad m. Pozri obr. 6.22.

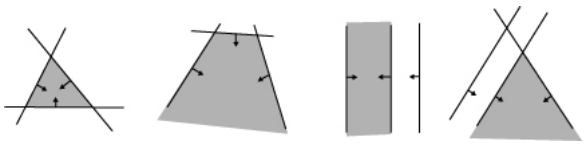
Teraz sa pozrime čo sa stane ak nám priamka pretína 2 po sebe nasledujúce hrany. Pribudnú nám 2 vrcholy a 1 ubudne, takže vrcholov bude $m + 1$. Pribudne nám 1 hrana a žiadna nám neubudne a tak i počet hrán bude $m + 1$. Pozri obr. 6.23.



Obr. 6.19. Polygonálny región určený 1 polrovinou.



Obr. 6.20. Polygonálny región určený 2 polrovinami.



Obr. 6.21. Polygonálny región určený 3 polrovinami.

Tieto pravidlá nám platia, pokiaľ bude región uzavretý, a teda bude mať maximálne m hrán a vrcholov. Rovnaké pravidlá platia, pokiaľ bude región otvorený, avšak v tomto prípade bude mať maximálne $m-1$ hrán a m vrcholov, a teda po pridaní polroviny nám taktiež nestúpne počet hrán a vrcholov nad $m+1$.

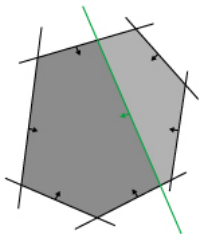
Ešte nám ostáva ošetriť prípad, keď bude l pretínať len v 1 hrane. V tomto prípade bude región otvorený a teda bude mať maximálne $m-1$ vrcholov a m hrán. Pokiaľ l pretína hranicu regiónu v jednej z dvoch polpriamok ohraničujúcich otvorenú časť regiónu, tak celkový počet vrcholov regiónu nám vzrastie maximálne o 1 a počet hrán tiež maximálne 1. Nový počet vrcholov bude maximálne m a hrán $m+1$. Pozri obr. 6.24. Pokiaľ l pretína niektorú z úsečiek ohraničujúcu región tak nám tiež pribudne maximálne 1 vrchol a 1 hrana avšak aspoň 1 vrchol a 1 hrana nám i ubudnú a tak počet vrcholov ani hrán nestúpne nad $m+1$. Pozri obr. 6.25.

Týmto sme ukázali, že pri prechode od m ku $m+1$ vrcholom sa nám zvýši počet hrán i vrcholov maximálne o 1.

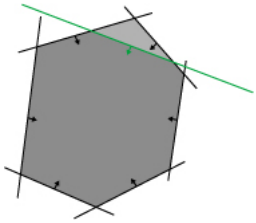
Spolu so zachovaním vlastnosti konvexnosti a aplikovaním prechodu od m ku $m+1$ a ošetrením prípadu bázy sme ukázali, že prienik m polrovín je konvexný polygonálny región s najviac m vrcholmi a m hranami.

□

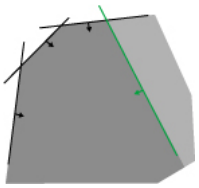
Riešenie (Gabriel Foltán, 18.12.2011): Topológia prieniku sa dá ukázať nasledovne : každá polrovina je konvexná a prienik konvexných množín je opäť konvexná množina. Preto prienik polrovín je konvexná



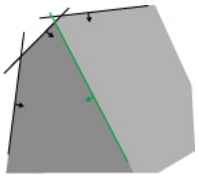
Obr. 6.22. Pridávanie polroviny ktorá pretína 2 po sebe nenasledujúce hrany regiónu.



Obr. 6.23. Pridávanie polroviny ktorá pretína 2 po sebe nasledujúce hrany regiónu.



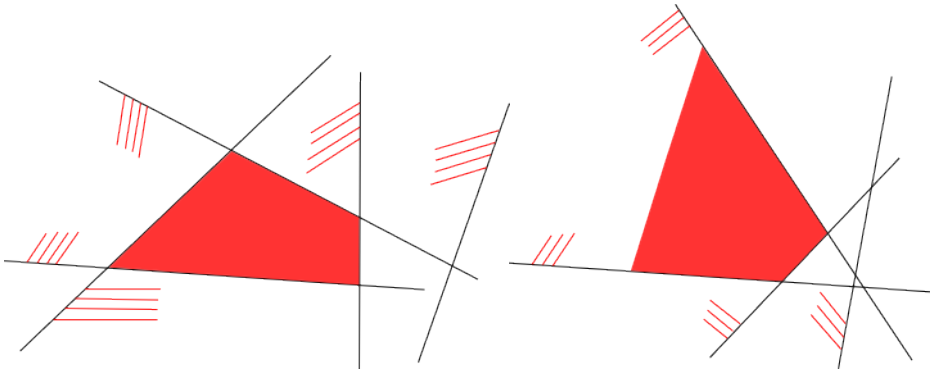
Obr. 6.24. Pridávanie polroviny ktorá pretína hranicu otvoreného regiónu v polpriamke.



Obr. 6.25. Pridávanie polroviny ktorá pretína hranicu otvoreného regiónu v úsečke.

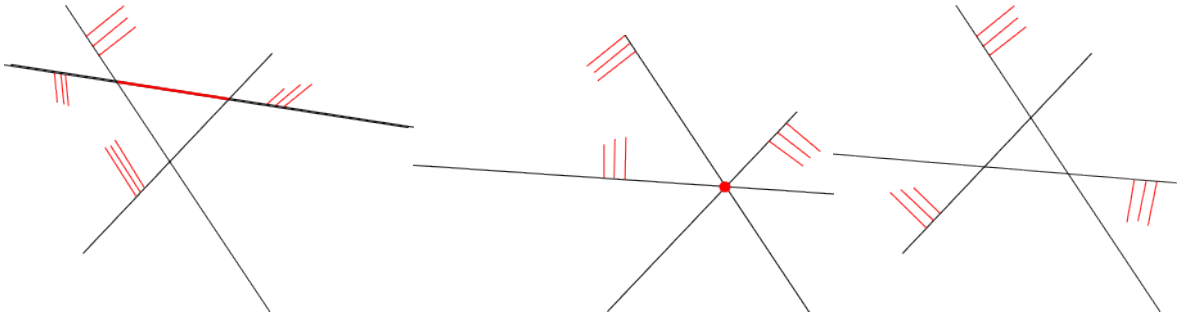
množina. Každý bod hranice prieniku musí ležať aj na nejakej hraničnej priamke definujúcej polrovinu. Preto hranica regiónu pozostáva z hrán nachádzajúcich sa na týchto hraničných priamkach. Kedže prienik je konvexný, každá hraničná priamka može "prispieť"najviac jednou hranou. Z toho vyplýva, že prienik m polrovín je konvexný polygonálny región ohraničený (obr. 6.26a.) najviac m hranami. Takýto región môže byť aj neohraničený (obrázok 6.26b) alebo sa môže stať úsečkou (obrázok 6.27a), bodom (obr. 6.27b.) alebo prázdnou množinou (obr. 6.27c.). Ešte treba určiť počet vrcholov.

Kedže každý vrchol je vrcholom dvoch hrán a súčasne každá hrana obsahuje najviac dva vrcholy, počet vrcholov takéhoto polygonálneho regiónu môže byť najviac m.



(A) príklad ohraničeného prieniku (B) neohraničený prienik polrovín

OBR. 6.26. Možnosti ohraničenosti prieniku



(A) prienikom je úsečka (B) bod (C) prázdna množina

OBR. 6.27. Príklady prieniku polrovín

□

CVIČENIE 6.8 (30 bodov). *Nech S je množina n úsečiek v rovine. Navrhnite dátovú štruktúru na získavanie nasledujúcej informácie: Je daná priamka l, koľko úsečiek pretína? Analyzujte zložitosť navrhnutého algoritmu.*

Riešenie (Marian Maričák, 19.12.2008): Na vstupe máme množinu n úsečiek v rovine. Označme ju $\mathbb{S}$. Pre danú priamku l máme zistiť, koľko úsečiek pretína. Ak by sme úlohu riešili hrubou silou, teda by sme otestovali prienik každej úsečky s priamkou l, vyšla by nám

lineárna časová zložitosť - $\mathcal{O}(n)$ (prienik úsečky a priamky vieme vypočítať v čase $\mathcal{O}(1)$, a to dosadením dvoch koncových bodov úsečky do všeobecnej rovnice priamky).

Potrebujeme štruktúru, ktorá nám pomôže znížiť počet nutných testov na určenie prieniku priamky a úsečky. Náš problém sa veľmi ponáša na ray-tracing, teda môžeme použiť rovnaké štruktúry na obmedzenie počtu rátaných prienikov. Jednou z možností je použitie BVH (Bounding Volume Hierarchy), kde ako ohraničujúce teleso použijeme AABB (Axis Aligned Bounding Box). V našom prípade (2D) bude ohraničujúcim telesom obdĺžnik, ktorého strany sú rovnobežné so súradnicovými osami.

BVH je strom, ktorého uzly sú tvorené ohraničujúcimi telesami. Ohraničujúce teleso v danom uzle obsahuje všetky ohraničujúce telesá svojich potomkov. Listy stromu sú tvorené ohraničujúcimi telesami pre primitíva (v našom prípade úsečky). Čiže ak priamka nepretína ohraničujúce teleso niektorého z uzlov, potom nepretína ani potomkov (úsečky) tohoto uzla a prienik priamky s týmito úsečkami nemusíme vôbec počítať. Pre danú priamku l budeme náš strom prehľadávať do hĺbky, pričom vždy testujeme prienik priamky a ohraničujúceho telesa v uzle. Takto určíme všetky úsečky, ktoré majú prienik s danou priamkou.

```
uzol VybudujBVH(𝕊, os)
{
        if (𝕊.size == 1)
                return 𝕊.front
        uzol rodic
        rodic.aabb = NajdiOhranicujuceTeleso(𝕊)
        utrieď 𝕊 podľa osi os
        𝕃 = 𝕊.lava-polovica
        ℝ = 𝕊.prava-polovica
        os = ďalšia os
        rodic.lavy = VybudujBVH(𝕃, os)
        rodic.pravy = VybudujBVH(ℝ, os)
        return rodic
}
```

Ako je vidno v pseudokóde pre vybudovanie BVH, tvoríme binárny strom. V každom kroku delíme zoznam úsečiek na dve polovice podľa inej súradnicovej osi. Prípadne môžeme úsečky triediť podľa dlhšej z

osí ohraničujúceho obdĺžnika. Pre účely triedenia je každá úsečka reprezentovaná jej stredným bodom.

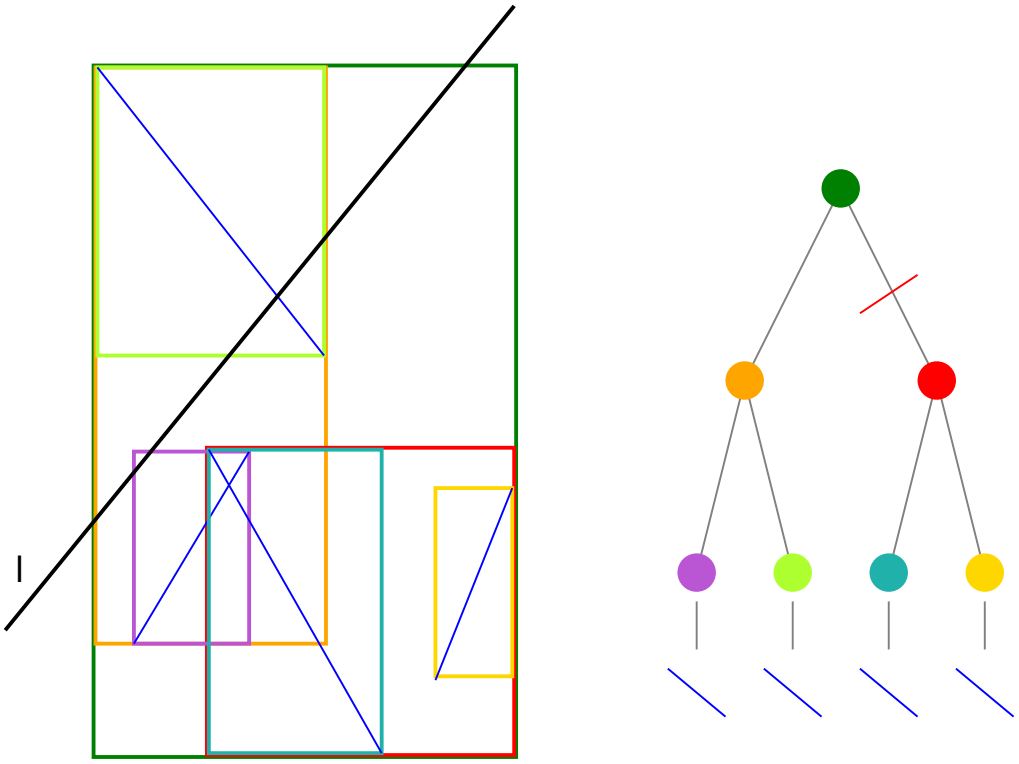
```
int pocetPrienikov(uzolBVH, l)
{
  if (uzolBVH == list)
     if uzolBVH ∩ l == 0 return 0
     else return 1
  int pocL, pocR
  if (uzolBVH.lavy != NULL) pocL = pocetPrienikov(uzolBVH.lavy,
l)
  if (uzolBVH.pravy != NULL) pocR = pocetPrienikov(uzolBVH.pravy,
l)
  return pocL+pocR
}
```

Získali sme binárny strom ohraničujúcich telies (pozri obr. 6.28). V najhoršom prípade budeme musieť prezrieť všetky uzly stromu, z čoho nám vychádza časová zložitosť $\mathcal{O}(n)$. V ostatných prípadoch bude časová zložitosť menšia. Záleží to od výšky uzla v ktorom priamka nepretína aspoň jedného z jeho potomkov. Čím bližšie bude tento uzol ku koreňu, tým menšia zložitosť. V najlepšom prípade nebude priamka pretínať ani koreň stromu a počet prienikov (0) vieme určiť v čase $\mathcal{O}(1)$. Toto je však veľmi špecifický prípad. V ideálnom prípade by sa mala celková zložitosť blížiť k hranici $\mathcal{O}(\log n)$, teda ak bude priamka vždy pretínať len jedného potomka rodičovského uzla.

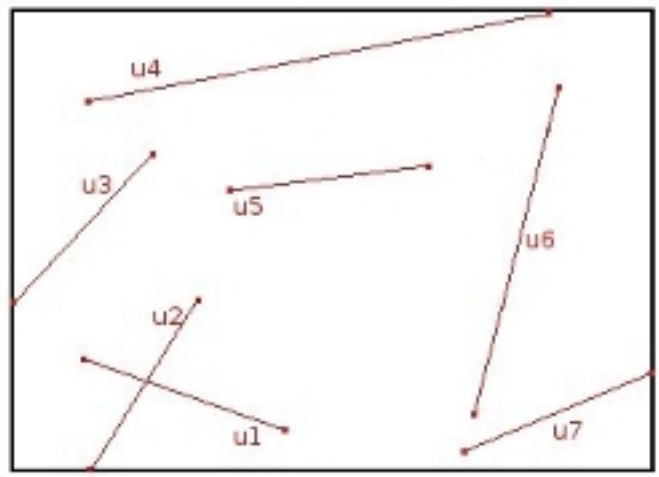
□

Riešenie (Viktória Bakurová, 29.12.2009): Majme množinu úsečiek S s prvkami $u_1, \dots u_n$ a majme danú priamku l. n úsečiek má spolu $2n$ koncových bodov. Nájdime v tejto množine bodov body s maximálnou a minimálnou x-ovou súradnicou a označme ich x_{max} a x_{min} a rovnako nájdime body s maximálnou a minimálnou y-ovou súradnicou a označme ich y_{max} a y_{min}. Zostrojme priamku rovnobežnú s osou x a prechádzajúcou bodom y_{max} a tiež priamku rovnobežnú s osou x a prechádzajúcou bodom y_{min}. Rovnako zostrojme priamky rovnobežné s osou y a prechádzajúce bodmi x_{max} a x_{min}. Takto zostrojené priamky sa pretnú v štyroch bodoch, označme si ich postupne A, B, C, D a vznikne obdĺžnik obsahujúci celý množinu S.Je zrejmé, že bod A má súradnice $[x_{min}, y_{min}]$, bod B má súradnice $[x_{max}, y_{min}]$, bod C má súradnice $[x_{max}, y_{max}]$ a bod D má súradnice $[x_{min}, y_{max}]$.

Vytvorme postupne štruktúru, pomocou ktorej budeme hľadať úsečky, ktoré priamka l pretína. Rozdeľme vzniktnutý obdĺžník na 4 časti tak, že nájdeme stredy úsečiek AB a CD (označme ich S_{AB} a S_{CD}) a spojíme ich a rovnako spojíme stredy úsečiek BCa DA(označme ich



Obr. 6.28. Príklad jednoduchej scény so štyrmi úsečkami, priamkou l a zodpovedajúcim BVH.



Obr. 6.29. Množina S a ohraničujúci obdĺžnik

S_{BC} a S_{DA}). Úsečky $S_{AB}S_{CD}$ a $S_{BC}S_{DA}$ sa pretnú v bode, ktorý si označíme S. Pôvodný obdĺžnik $ABCD$ sa tak rozdelil na štyri nové obdĺžniky: obdĺžnik $AS_{AB}SS_{DA}$, obdĺžnik $S_{AB}BS_{BC}S$,obdĺžnik $S_{DA}SS_{CD}D$ a obdĺžnik $SS_{BC}CS_{CD}$. Teraz už ostáva len pre každú úsečku z množiny S rozhodnúť, v ktorej zo vzniknutých 4 oblastí leží buď celá alebo

jej časť. Pre každú úsečku u_i z množiny S nájdeme prieniky s úsečkami $S_{AB}S_{CD}$ a $S_{BC}S_{DA}$. Môžu nastať 4 možnosti:

(a) $u_i \cap S_{AB}S_{CD}=\emptyset$ a $u_i \cap S_{BC}S_{DA}=\emptyset$
(b) $u_i \cap S_{AB}S_{CD}=\emptyset$ a $u_i \cap S_{BC}S_{DA} \neq \emptyset$
(c) $u_i \cap S_{AB}S_{CD} \neq \emptyset$ a $u_i \cap S_{BC}S_{DA}=\emptyset$
(d) $u_i \cap S_{AB}S_{CD} \neq \emptyset$ a $u_i \cap S_{BC}S_{DA} \neq \emptyset$

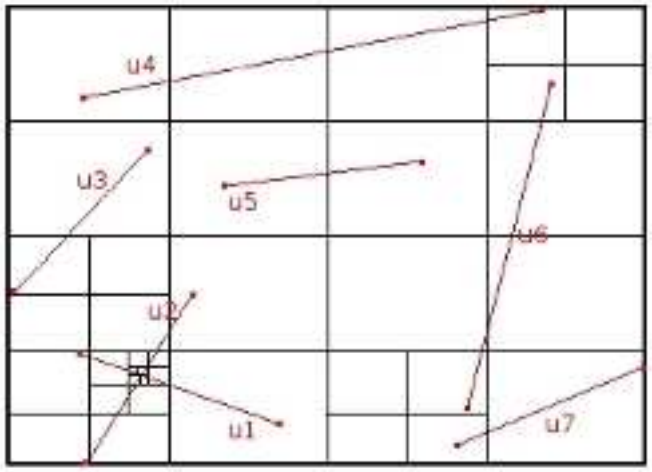
Rozoberme si teraz každú možnosť zvlášť: 1. Ak nastala táto možnosť, znamená to, že daná úsečka leží celá v jednej zo 4 oblastí. O tom, v ktorej, rozhodneme jednoducho tak, že zistíme, v ktorom obdlžníku leží jeden z koncových bodov tejto úsečky. 2. Ak nastala táto možnosť, znamená to, že daná úsečka leží v 2 oblastiach. V jednej oblasti bude ležať začiatočný a v druhej koncový bod tejto úsečky, stačí teda rozhodnúť, v ktorých obdĺžnikoch ležia koncové body danej úsečky. 3. Táto možnosť je analogická možnosti 2., takisto stačí zistiť, v ktorej oblasti ležia začiatočný a koncový bod úsečky. 4.Ak nastala táto možnosť, znamená to, že daná úsečka leží až v troch oblastiach. V tomto prípade sa naša úsečka rozdelí na 3 úsečky v bodoch prieniku a každá z nich leží v jednej z obdĺžnikov. Dve z týchto úsečiek budú obsahovať koncové body pôvodnej úsečky a stačí sa teda rozhodnúť, v ktorej oblasti ležia tieto body. Spojením dvoch nájdených bodov prieniku vznikne nová úsečka, o ktorej ešte musíme rozhodnúť, v ktorej oblasti leží. To možeme urobiť napríklad tak, že spočítame bod ležiaci v strede tejto úsečky a nájdeme obdĺžnik, v ktorom tento bod leží. Takýmto spôsobom teda môžeme jednoducho pre každú úsečku rozhodnúť, ktorú zo vzniknutých oblastí pretína. Rekurzívne delíme každý vzniknutý obdlžník až dovtedy, kým ho nepretína len konečný počet úsečiek z množiny S alebo sa nedosiahla maximálna stanovená výška stromovej štruktúry. Treba si uvedomiť, že v ďalších krokoch delenia treba rozhodovať o polohe len tých úsečiek, ktoré ležali v oblasti, ktorá sa delila na menšie časti. Ak nejaká úsečka neležala v pôvodnom obdĺžniku, nebude ležať ani v žiadnej časti, na ktoré sa tento obdĺžnik rozdelí.

Takto vzniknutá dátová štruktúra má tvar stromu a má nasledovné vlastnosti:

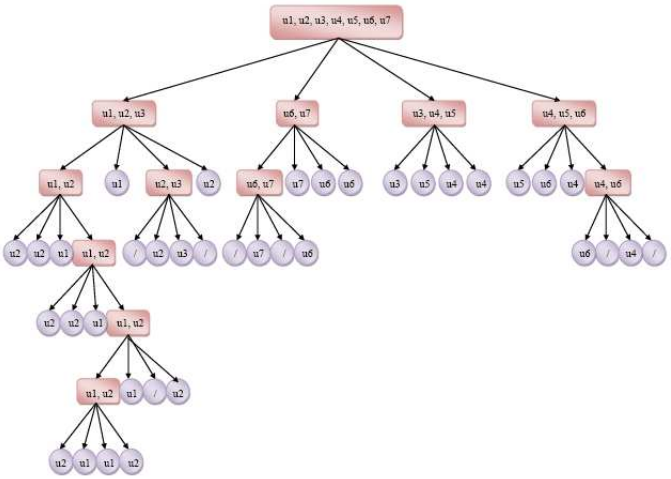
- koreňom stromu je obdĺžnik obsahujúci všetky úsečky množiny S
- každý uzol stromu má 4 synov alebo ani jedného, ak ho pretína len konečný počet priamok množiny S
- každá úsečka môže byť v danej štruktúre zapísaná na jednej úrovni maximálne trikrát

Preľadávanie danej dátovej štruktúry:

Hľadajme teraz, koľko úsečiek pretne priamka l z danej množiny. Budeme hľadať obdĺžniky, cez ktoré daná priamka prechádza, postupujeme od koreňa stromu až k listom. Ak priamka l nepretína obdĺžnik nachádzajúci sa v koreni, nepretína ani jednu úsečku množiny S. Ak



Obr. 6.30. Rekurzívne delenie ohraničujúceho obdĺžnika

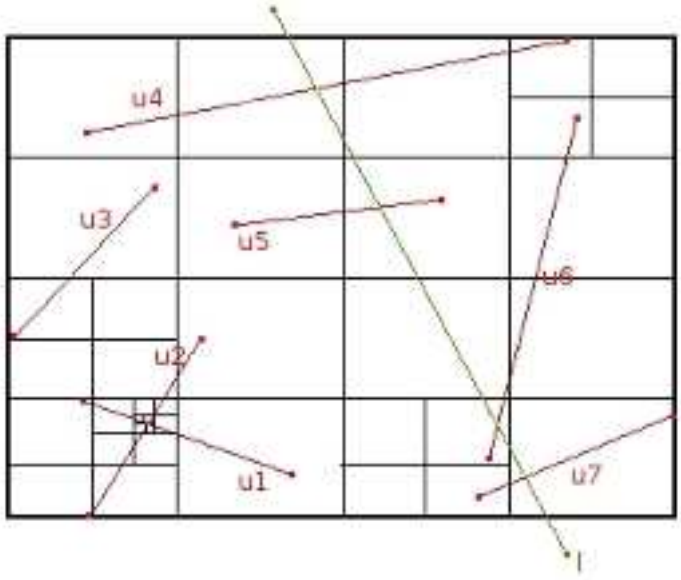


Obr. 6.31. Prislúchajúca dátová štruktúra

tento obdlžník pretína, pozrieme sa, ktorý zo 4 synov pretína. Je treba si uvedomiť, že može preťať maximálne tri zo 4 synov. Ďalej je potrebné si uvedomiť, že ak daná priamka nepretla nejakého otca, netretne už ani žiadneho syna, ktorý patrí tomuto uzlo a netreba ďalej túto časť stromu brať do úvahy. Počet preťatých úsečiek nájdeme tak, že dôjdeme až ku listom stromu a počítame, či existujú prieniky len s tými úsečkami z S, ktoré sa v týchto listoch nachádzajú. Okrem toho je dobré si pamätať, na ktorú úsečku sme už v liste stromu narazili a zisťovali, či sa pretne s priamkou la v prípade, že na takúto úsečku narazíme aj v inom liste, nemusíme znovu počítať prienik.

časová zložitosť:

1) Vytvorenie stromovej dátovej štruktúry
   - Nájdenie ohraničujúceho obdĺžnika: $O(n)$



OBR. 6.32. Množina S a priamka l a ohraničujúci obdĺžnik

- Delenie obdĺžnika na 4 časti: $O(1)$
- Delenie n krát: $O(n)$ (nech n je maximálne stanovená hĺbka stromu)
- Rozhodnutie, do ktorých oblastí úsečka patrí: $O(1)$
- Rozhodnutie pre n úsečiek: $O(n)$

Celková časová zložitosť: $O(n \log n)$

2) Prehľadanie dátovej štruktúry:

- V najlepšom prípade priamka l nepretne obdĺžnik uložený v koreni stromu a v čase $O(1)$ sa zistí, že l nepretína ani jednu úsečku množiny S
- V najhoršom prípade musíme nájsť prienik každej úsečky z množiny Ss priamkou l,čo dáva časovú zložitosť $O(n)$

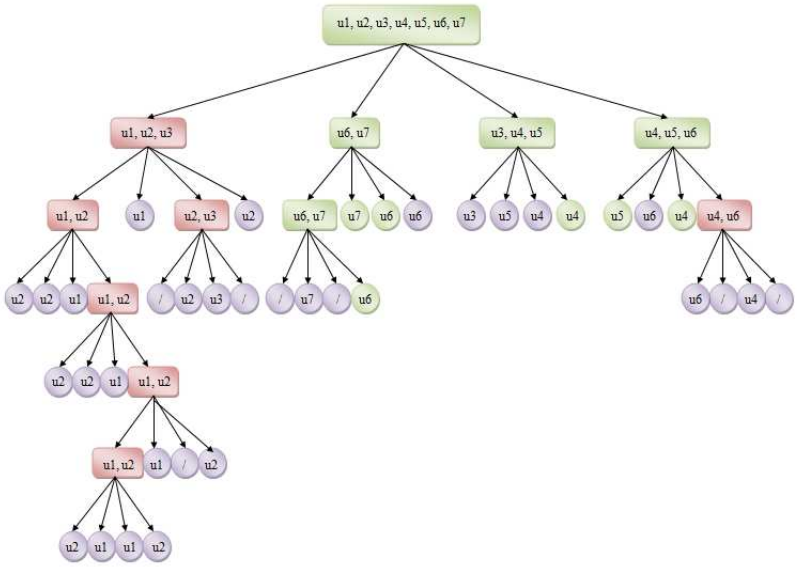
□

CVIČENIE 6.9 (30 bodov). *Zostavte algoritmus na výpočet vzdialenosti medzi mnohouholníkom P s n vrcholmi a ľubovolnou priamkou L. Pokúste sa dosiahnuť hranicu $\mathcal{O}(\log n)$ pri pevnom P.*

Riešenie (Martin Manduch, 3.1.2010): Vzdialenosť dvoch množín $A, B \subset \mathbb{E}^2$ je definovaná ako

$$\inf\{d(\mathbf{x}, \mathbf{y});\ \mathbf{x} \in A, \mathbf{y} \in B\}.$$

LEMA 6.1. *Vzdialenosť priamky p a mnohouholníka M môžeme určiť ako minimálnu vzdialenosť priamky a vrcholov konvexného obalu $CH(M)$.*



Obr. 6.33. Prechod dátovej štruktúry, je znázornený zelenou farbou

Dôkaz: Vzdialenosť priamky a jednoduchého mnohouholníka je určite vzdialenosť priamky a nejakého bodu z jeho hranice. Ak existuje jeden taký bod, tak musí byť vrchol. Ak existuje viac bodov, budú ležať na jednej hrane, tak vzdialenosť zase môžeme určiť ako vzdialenosť koncového bodu tej hrany. Ukážme, že nemôže nastať taký prípad, že by vzdialenosť priamky od M bola vzdialenosť priamky od takého vrcholu $\mathbf{v} \in M$, ktorý nie je vrcholom $CH(M)$. To znamená, že by $\mathbf{v}$ ležal vo vnútri konvexného obalu a kolmica prechádzajúca $\mathbf{v}$ a kolmá na priamku p, by pretla konvexný obal v nejakom vrchole $CH(M)$ (spor) alebo v nejakej hrane $CH(M)$, ktorej aspoň jeden koncový bod musí byť bližšie k p ako vrchol $\mathbf{v}$. Spor. □

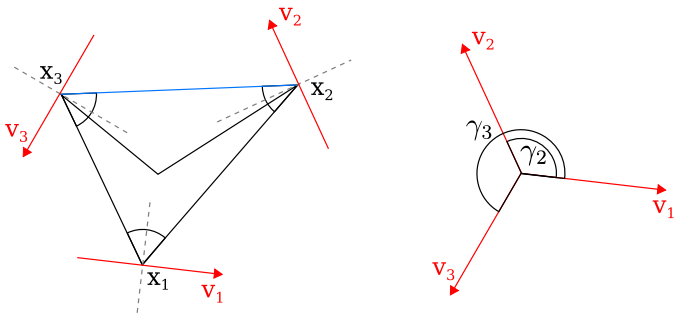
Predpokladajme, že $CH(M)$ je orientovaný proti smeru chodu hodinových ručičiek. Nech vrcholy $CH(M)$ sú $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Označme vnútorné uhly $CH(M)$ ako $\alpha_1, \ldots, \alpha_n$. Vytvoríme vektory $\vec{\mathbf{v}}_1, \ldots, \vec{\mathbf{v}}_n$ tak, že vektor $\vec{\mathbf{v}}_i$ je kolmý na os uhla α_i a sú rovnako orientované ako $CH(M)$ (obr. 6.34 vľavo). Vyberieme jeden vektor, napr. $\vec{\mathbf{v}}_1$ a usporiadame ostatné podľa polárneho uhla s $\vec{\mathbf{v}}_1$ (obr. 6.34 vpravo). Označme polárny uhol, ktorý zviera $\vec{\mathbf{v}}_i$ s $\vec{\mathbf{v}}_1$ ako γ_i, $\gamma_i \in [0, 2\pi)$. Z intervalov $[\gamma_i, \gamma_{i+1}]$ vytvoríme segmentový strom. Ak priamka nepretína M a je orientovaná tak, že M je naľavo a jej smerový vektor je z intervalu $[\gamma_i, \gamma_{i+1}]$, tak najbližšia hrana k nej je $\mathbf{x}_i, \mathbf{x}_{i+1}$.

Určenie vzdialenosti

(a) Zoberieme jeden vrchol $\mathbf{b}$ z $CH(M)$ a podľa neho orietujeme priamku p tak, aby $\mathbf{b}$ ležal od nej naľavo.

(b) Určíme polárny uhol γ_p, ktorý zviera p s $\vec{\mathbf{v}}_1$.
(c) V segmentovom strome nájdeme interval $[\gamma_k, \gamma_{k+1}]$, v ktorom leží γ_p.
(d) Ak je aspoň jeden z bodov $\mathbf{x}_k, \mathbf{x}_{k+1}$ napravo od p, tak vzdialenosť je 0, lebo p pretína M.

Inak vzdialenosť p od $CH(M)$ vypočítame ako vzdialenosť bližšieho z bodov $\mathbf{x}_k, \mathbf{x}_{k+1}$.



Obr. 6.34. Pôvodný n-uholník je čiernou farbou. Modrou je doplnený na konvexný obal. Vpravo sú polárne uhly γ_2 a γ_3, $\gamma_1 = 0$.

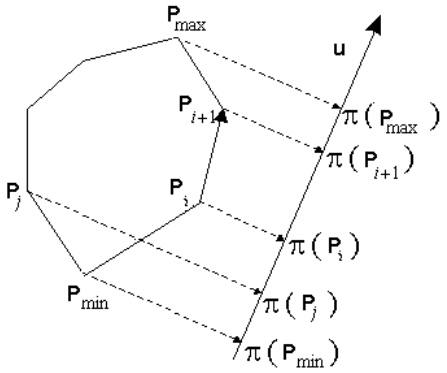
Zložitosť predspracovania:
Vytvorenie konvexného obalu jednoduchého mnohouholníka trvá $\mathcal{O}(n)$, vypočítanie vektorov $\vec{\mathbf{v}}_i$ a uhlov, ktoré zvierajú s $\vec{\mathbf{v}}_1$ trvá $\mathcal{O}(n)$ a vytvorenie segmentového stromu sa dá urobiť v čase $\mathcal{O}(n)$. Teda predspracovanie vieme spraviť v čase $\mathcal{O}(n)$.

Zložitosť nájdenia vzdialenosti
Vo všetkých krokoch okrem tretieho je časová zložitosť $\mathcal{O}(1)$. Vyhľadávanie v segmentovom strome trvá $\mathcal{O}(\log n)$. Celková zložitosť pri pevnom M je $\mathcal{O}(\log n)$. □

Riešenie (Alojz Kováčik, 2.1.2010): Pred tým ako sa pustíme do samotného riešenia problému, ujasníme si najskôr pojem **extrémny bod mnohouholníka v smere vektora**. Extrémny bod mnohouholníka W v smere vektora u je taký bod $x \in W$, pre ktorý všetky ostatné body množiny W ležia v polrovine určenej priamkou r, danou vektorom $\vec{v} =\perp \vec{u}$ a bodom x, pričom vektor $\vec{u}$ smeruje von z tejto polroviny. Najprv si ukážeme, ako nájsť takýto bod, neskôr tento poznatok využijeme pri riešení nášho problému.

Nájdenie extrémneho bodu mnohostena v smere vektora

Majme v euklidovskej rovine priamku L a mnohouholník S určený bodmi $P_0, P_1, \ldots, P_{n-1}, P_n = P_0$ danými proti smeru hodinových ručičiek. Nech sú body P_i kolmo premietnuté na priamku L v smere $\vec{u}$, potom extrémne body mnohostena sú tie, ktorých priemety sú extrémne body na priamke L (obr. 6.35).



Obr. 6.35. Určenie extrémneho bodu na základe projekcie na priamku L.

Nech $\pi : \mathbb{R}^2 \to L$ je kolmé premietanie na priamku L. Hovoríme, že bod P_i je nad bodom P_j vzhľadom na $\vec{u}$, ak má vektor $\pi(P_i) - \pi(P_i)$ zhodný smer s vektorom $\vec{u}$ a tiež platí, že

$$\vec{u} \cdot (\pi(P_i) - \pi(P_i)) > 0. \tag{10}$$

Postupné prehľadávanie

Jedným spôsobom ako nájsť extrémne body mnohostena W v smere $\vec{u}$, je v danom poradí testovať každý bod, pričom ho vždy porovnávame s aktuálnym maximálnym (minimálnym) bodom. Na porovnávanie využijeme vzťah (7). Ak sa nachádza nad (pod), označíme ho ako aktuálny maximálny (minimálny) bod a pokračujeme v testovaní ďalšieho bodu. Po prejdení všetkých bodov máme označené naše extrémy ako aktuálne maximálne (minimálne) body.

Testovanie ľubovoľného bodu prebehne v čase $\mathcal{O}(1)$ a každý bod testujeme práve raz, preto časová zložitosť tohto algoritmu je $\mathcal{O}(n)$.

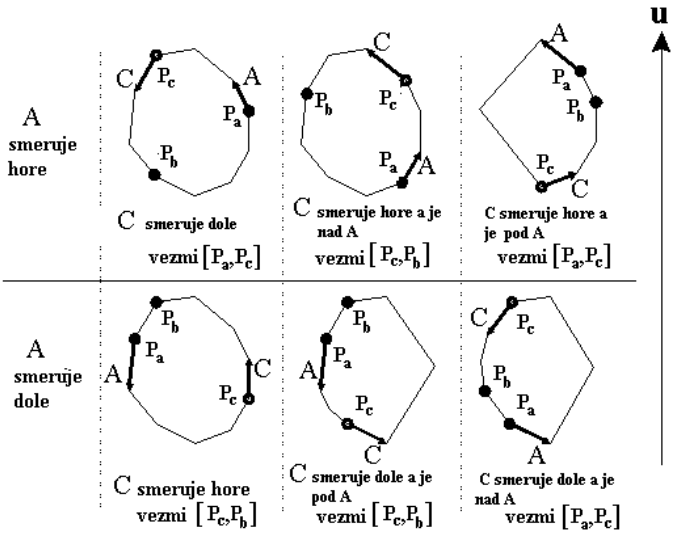
Binárne prehľadávanie

Ak máme pevne dané, že náš mnouholník je konvexný, môžeme na zisťovanie jeho extrémneho bodu v smere $\vec{u}$ využiť nasledujúci algoritmus s časovou zložitosťou $\mathcal{O}(\log n)$.

Predpokladajme, že extrémny bod patrí časti mnohouholníka, začínajúcej v bode P_a a končiacej v P_b. Túto časť budeme ďalej označovať ako reťazec. Zoberme bod P_c patriaci tejto časti a rôzny od P_a, P_b. Potrebujeme nájsť kritérium, ktoré nám pomôže určiť, v ktorom z dvoch podreťazcov $[P_a, P_c]$, $[P_c, P_b]$ sa nachádza hľadaný extrémny bod.

Keďže pracujeme s konvexným mnohouholníkom, môžeme tento podreťazec určiť porovnaním vektora A vychádzajúceho z bodu P_a a vektora C vychádzajúceho z bodu P_c. Môže nastať 6 prípadov znázornených na obrázku 6.36. To že tieto sú všetky možné prípady vyplýva

z faktu, že konvexný mnohouholník W je monotónny vo všetkých smeroch a teda aj v smere vektora $\vec{u}$. Preto pozostáva z práve dvoch monotónnych reťazcov, rastúceho a klesajúceho vzhľadom na vektor $\vec{u}$, ktoré spájajú hľadaný maximálny a minimálny bod.



OBR. 6.36. Obrázok znázorňuje 6 prípadov pre určenie podreťazca. Znaky A, C sú vektory vychádzajúce z bodov mnohouholníka.

Index c môžeme určiť napr. ako $c = \frac{(a+b)}{2}$. Ešte pred zisťovaním, ktorý zo šiestich prípadov nastal, je potrebné zistiť, či bod P_c nie je lokálne extrémny bod. Toto zistíme tak, že skontrolujeme či sa bod P_c nachádza nad (pod) susednými bodmi vzhľadom na vektor $\vec{u}$. V konvexnom polygóne platí, že ak je jeho vrchol lokálne extrémny bod, tak je aj globálne extrémny bod v danom smere.

Algoritmus pre zistenie maximálneho bodu mnohouholníka v smere $\vec{u}$:

(a) Nastav krajné indexy reťazca $a = 0, b = n$. Reťazec začína bodom P_0 a končí bodom P_n a je teda na začiatku celý mnohouholník.

(b) Zisti či bod P_a nie je lokálne maximum v danom smere. Ak áno, bod P_c je hľadaný maximálny bod, ukonči vykonávanie. Ak nie pokračuj.

(c) Kým $b > a + 1$, opakuj:
- Priraď $c = \frac{(a+b)}{2}$.
- Zisti či bod P_c nie je lokálne maximum v danom smere. Ak áno, bod P_c je hľadaný maximálny bod, ukonči vykonávanie. Ak nie pokračuj.
- Zisti ktorý zo šiestich prípadov nastal a podľa toho priraď krajné indexy a, b.

Nájdenie vzdialenosti mnohostena od priamky

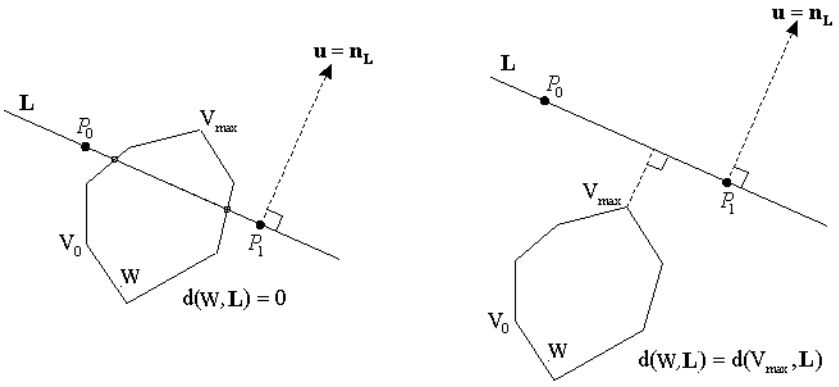
Majme mnohosten W a priamku L. Predchádzajúce dva algoritmy použijeme na zistenie vzdialenosti W od L.

Najprv určime jednotkový vektor $\vec{v}$ tak, aby bol kolmý na priamku L a zároveň tak, aby sa najmenej jeden bod polygónu W nachádzal v opačnej polrovine, ako polrovina do ktorej smeruje vektor $\vec{v}$. Pre ľubovoľný bod P nachádzajúci sa v polrovine, kam smeruje vektor $\vec{v}$ platí, že $\vec{v} \cdot (P - P_0) > 0$, kde $P_0 \in L$ sme si určili. Túto polrovinu budeme označovať ako v-pozitívnu, jej opak v-negatívnu.

Ďalej nájdeme maximálny bod V_{max} mnohouholníka W v smere vektora $\vec{v}$. Tu môžu nastať dva prípady:

(a) V_{max} sa nachádza na vo v-pozitívnej polrovine, vtedy L pretína mnohouholník W, čo znamená, že $d(W, L) = 0$

(b) V_{max} sa nachádza na vo v-negatívnej polrovine, preto sa aj ostatné body mnohouholníka W nachádzajú v tejto polrovine. Z toho vyplýva, že V_{max} je najbližší bod k priamke L a teda $d(W, L) = d(V_{max}, L)$.



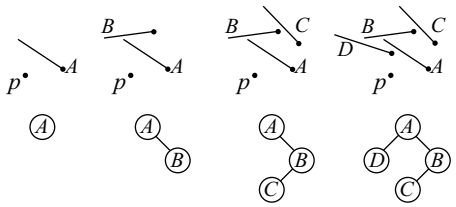
Obr. 6.37. Prípady vzájomnej polohy priamky a mnohouholníka pri určovaní vzdialenosti.

Analýza časovej zložitosti

Určenie jednotkového vektora $\vec{v}$ trvá $\mathcal{O}(1)$. Ako už vieme, v prípade ľubovoľného mnohostena trvá nájdenie extrémneho bodu $\mathcal{O}(n)$. V prípade konvexného mnohostena je to $\mathcal{O}(\log n)$. Preto sa časové zložitosti zhodujú so zložitosťami algoritmu pre nájdenie extrémneho bodu mnohostena v danom smere. □

Cvičenie 6.10 (25 bodov). *Nájdite optimálny algoritmus, ktorý otestuje, či sa nejaké dve kružnice z daných n kružníc pretínajú.*

Riešenie (Alojz Kováčik, 2.1.2010): Algoritmus na vyriešenie tohto problému dostaneme jednoduchou modifikáciou algoritmu z cvičenia



Obr. 6.38. Vkladanie úsečiek do binárneho stromu.

6.4. Stačí zastaviť prácu algoritmu vtedy, keď pri jeho vykonávaní narazíme na prvý priesečník štvrťkružníc.

Analýza časovej zložitosti

Hlavný cyklus sa opakuje $\mathcal{O}(n)$ krát a operácie na aktualizáciu použitých dátových štruktúr nevyžadujú viac ako $\mathcal{O}(\log n)$ času. Preto časová zložitosť algoritmu je $\mathcal{O}(n \log n)$. □

Cvičenie 6.11 (30 bodov). *Nech S je množina n disjunktných úsečiek v rovine a* $\mathbf{p}$ *je bod neležiaci na žiadnej z nich. Nájdite všetky úsečky viditeľné z* $\mathbf{p}$ *(t.j. všetky obsahujúce taký bod* $\mathbf{q}$*, že vnútro úsečky* $\mathbf{pq}$ *nepretína žiadnu úsečku z S). Nájdite algoritmus zložitosti $\mathcal{O}(n \log n)$.*

Riešenie (Lukáš Tencer, 1.11.2008):

Majme n úsečiek $\{u_1, \ldots, u_n\}$ a bod p pre ktorý chceme určiť z množiny úsečiek tie, ktoré z tohto bodu vidíme. Nech koncové body priamok sú $(U_1, \ldots, U_m)$, $\quad m = 2n$.

Na určenie viditeľných úsečiek použijeme zametací algoritmus, založený na udalostiach. Zametať budeme z bodu p v uhle 360 stupňov. Pomocou zametacej polpriamky sa priestor rozdelí na časti, v ktorých určíme ktorá priamka je k nášmu bodu najbližšia a teda viditeľná. V našom algoritme budeme potrebovať binárny strom, v ktorom budú uložené práve viditeľné časti priamok. Najľavejší prvok v strome nám bude hovoriť o najbližšej priamke k bodu p.

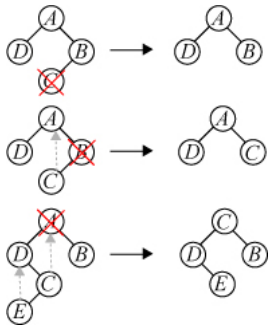
Teraz si bližšie opíšme dátovú štruktúru binárneho stromu, ktorý budeme používať.

Vo vrcholoch nášho stromu budú uložené úsečky. Strom bude podporovať tri operácie a to **vlož()**, **vyber()** a **najľavejšíPrvok()**.

Každá úsečka v rovine vytína dve polroviny. Polrovinu v ktorej leží bod pohľadu p označme ako kladnú (+), druhú polrovinu ako zápornú (-).

Operácia **vlož()** bude mať dva vstupné parametre a to bod úsečky u a samotnú úsečku A. Nový pridávaný vrchol bude vždy listom. Pri vkladaní sa potrebujeme rozhodnúť či pôjdeme z aktuálne testovaného vrcholu stromu V doľava alebo doprava. Doprava pôjdeme pokiaľ bod u leží v kladnej polrovine, doľava pokiaľ leží v zápornej polrovine, určenej úsečkou X uloženou vo vrchole V. Pre ilustráciu pozri obr. 6.38.

```
vloz ( bod u, priamka A ) {
```



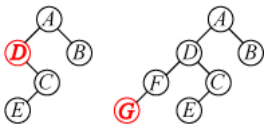
OBR. 6.39. Odstraňovanie prvkov z binárneho stromu.

```
  k = koren;
  while (true){
    if( jevKladnejPolrovine( u , k.usecka )
      if( k.lavySyn != nill )
        k=k.lavySyn;
      else{
        k.pridajLavehoSyna( A );
        skonci();
      }
    else
      if( k.pravySyn != nill );
        k=k.pravySyn;
      else{
        k.pridajPravehoSyna( A );
        skonci();
      }
  }
}
```

Operácia **vyber()** bude mať dva vstupné parametre a to bod úsečky u a samotnú úsečku A. Najprv prehľadávame pomocou vrcholu u strom, pokiaľ nenájdeme vrchol V v ktorom je uložená úsečka A. Teraz chceme vrchol V odstrániť. Môžu nastať 3 prípady:

- V nemá žiadneho syna t.j. je listom a tak ho jednoducho vymažeme.
- V má jedného syna, tak vrchol V odstránime a jeho syna pripojíme ako syna jeho otca na pozíciu vrcholu V.
- V má dvoch synov, tak ako jeho náhradu použijeme vrchol $\widetilde{v}$, ktorý je maximom ľavého podstromu. Ten jelikož je maximom ľavého podstromu je buď listom a tak nepotrebujeme nič doladiť, alebo má jedného syna, v takom prípade tohto syna nastavíme ako syna otcovi vrcholu $\widetilde{v}$ (postup ako keď vymazávame vrchol s jedným synom).

Ešte ošetríme prípad pokiaľ sa úsečka A v strome nenachádza. V tomto prípade nevykonáme žiadnu akciu. Pre ilustráciu pozri obr. 6.39.



OBR. 6.40. Hľadanie najľavejšieho prvku v binárnom strome.

```
vyber ( bod u, priamka A ) {
  v = najdiVrchol( u , A );
  if( jeList( v ) )
    vymaz(v);
    skonci();
  if( maJednohoSyna( v ) )
    v.otec=v.syn;
    vymaz(v);
    skonci();
  if( maDvochSynov( v ) )
    x=maximum( v.lavySyn )
    v.priamka=x.priamka;
    x.otec=x.syn;
    vymaz(x);
    skonci();
}
```

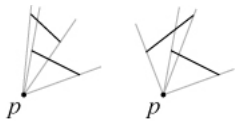
Operácia **najľavejšíPrvok()** bude jednoducho prechádzať strom od koreňa smerom k ľavému synovi až kým nenájdeme taký, ktorý nemá ľavého syna. Pre ilustráciu pozri obr. 6.40.

```
najlavejsiPrvok ( ) {
  k = koren;
  while ( k.lavySyn != nill )
    k=k.lavySyn;
  return k;
}
```

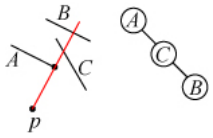
Všetky hore popísané operácie sa vykonávajú na štandardnom binárnom vyhľadávacom strome a vieme ich vykonávať v čase $\mathcal{O}(n)$. Namiesto BVS by sme mohli použiť bez ujmy na všeobecnosti AVL strom na ktorom vieme všetky predchádzajúce operácie vedeli vykonávať v čase $\mathcal{O}(\log n)$.

Aby sme mohli použiť hore uvedenú štruktúru na ukladanie úsečiek musíme zaručiť, že sa situácia v jednotlivých výsekoch nebude meniť. Zoberme si príklad dvoch úsečiek A a B. V strome máme uloženú pozíciu A voči B. pokiaľ celá úsečka A leží v niektorej polrovine určenej úsečkou B tak je prípad triviálny a situácia sa meniť nebude. Pozri obr. 6.41. Pokiaľ nebude celá úsečka A ležať v polrovine určenej úsečkou B tak sa pozícia zmení na priesečníku úsečiek. Toto však vyvracia predpoklad, že úsečky sú disjunktné.

Vychádzajú z predchádzajúcich predpokladov teraz môžeme popísať algoritmus viditeľnosti ktorý bude vyzerať nasledovne:



Obr. 6.41. Vzájomné polohy úsečiek.



Obr. 6.42. Počiatočné vkladanie úsečiek do binárneho stromu.

(a) Zober bod p a priraď hraničným bodom úsečiek polárne súradnice podľa neho. Ulož ich do poľa P=(U_1..U_m).

(b) Zober bod U_i, ak je to začiatočný bod úsečky u_i ,tak pridaj u_i do stromu S. Ak je to koncový bod odstráň úsečku u_i zo stromu S. To ktorý bod je koncový alebo začiatočný vieme určiť vzhľadom na orientáciu polárnej sústavy súradníc.

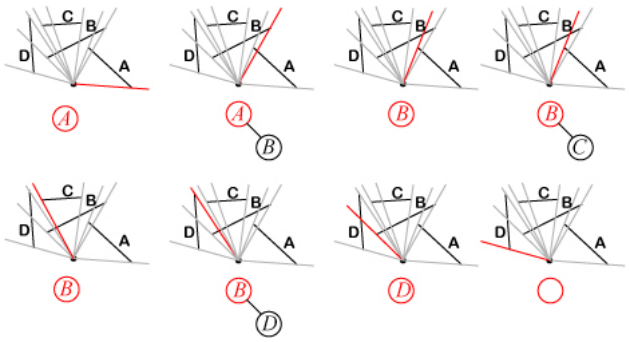
(c) Pozri sa do stromu S, ak strom nie je prázdny tak priraď najľavejší prvok v F do zoznamu L.

(d) Opakuj krok 2. a 3. postupne pre i=1..m.

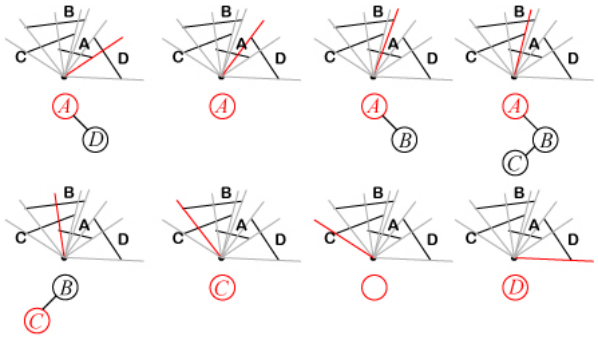
(e) Všetky viditeľné priamky sú teraz uložené v zozname L. Ak chceme, môžeme odstrániť rovnaké prvky zoznamu.

```
suViditelne ( body, priamky, bod p ) {
  priradPolarSur( body , p );
  zoradPodlaPolar( body );
  for ( i=0 , i < pocet(body) , i++ ){
    if ( jeZaciatocny( body[i] ) ) // bod je začiatočný bod
priamky
      strom.pridaj( body[i], priamkaSoZaciatocnym(body[i]) );
// priraď túto priamku do stromu
    else strom.vyber( body[i], priamkaSKoncovym(body[i]) ); //
je koncový, odstráň priamku zo stromu
      vysledok.pridaj( strom.najlavejsiPrvok() ); // do
výsledku pridaj viditeľnú priamku v danom výseku
  }
  return vysledok;
}
```

Po skončení algoritmu dostaneme všetky viditeľné priamky z bodu p. Priradenie polárnych súradníc a zoradenie bodov podľa nich v prvom



OBR. 6.43. Jednoduchý príklad použitia algoritmu viditeľnosti.



OBR. 6.44. Zložitejší príklad použitia algoritmu viditeľnosti.

kroku nám zaberie $\mathcal{O}(n \log n)$ času. Akákoľvek operácia so stromom za použitia AVL stromu trvá $\mathcal{O}(\log n)$ času. Čiže pokiaľ prejdeme všetky vrcholy tak celková zložitosť algoritmu bude $\mathcal{O}(n \log n)$.

Aby algoritmus korektne fungoval je dôležité usporiadanie bodov podľa polárnych súradníc. A preto ako prvý bod v pri usporiadavaní zvolíme taký, ktorý určite vidíme z bodu p, môžme zvoliť napríklad najbližší bod k bodu p. Problém by mohol nastať pokiaľ by sme dostali koncový bod úsečky pred začiatočným. Preto ešte spolu s prvým bodom musíme do stromu vložiť všetky úsečky, ktoré pretína polpriamka l začínajúca v bode p so smerom $\overrightarrow{Y} = v - p$. Pridávame ich v takom poradí v akom ich polpriamka l pretína. Pozri obr. 6.42.

Celý algoritmus si teraz ilustrujme na pár príkladoch ako obr. 6.43 a obr. 6.44. □

CVIČENIE 6.12 (50 bodov). *Nech $P, Q \subset \mathbb{E}^2$ sú jednoduché mnohouholníky také, že $\operatorname{int} P \cap \operatorname{int} Q = \emptyset$ a $Q \subset \operatorname{conv}(P)$. Nájdite kritérium, či existuje taký spojitý jednoparametrický systém $\phi : [0,1] \to \mathbb{E}^2$ zhodností mnohouholníka Q (pohyb mnohouholníka Q), že*

- $\phi(0) = \mathrm{id}_{\mathbb{E}^2}$,
- $\operatorname{int} \phi(t)(Q) \cap \operatorname{int} P = \emptyset$, $t \in [0,1]$,

- $\operatorname{int}\phi(1)(Q) \cap \operatorname{int}\operatorname{conv}(P) = \emptyset$.

Chceme zistiť, či sa dá mnohouholník Q „vyvliecť" z mnohouholníka P.

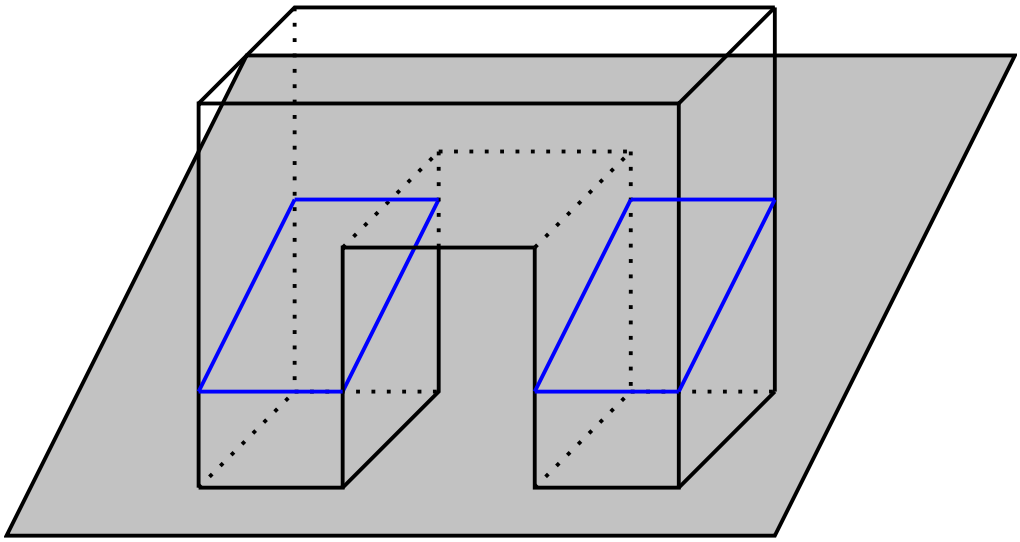
Cvičenie 6.13 (25 bodov). *Opíšte algoritmus prieniku roviny a mnohostena v $\mathbb{E}^3$. Odhadnite zložitosť tohoto algoritmu.*

Riešenie (Michal Ferko, 7.12.2011):Výsledkom takéhoto algoritmu môže byť viacero nesúvislých množín, ktoré sú buď mnohouholníky, úsečky alebo iba body. Najprv sa spraví prechod cez všetky steny F_i mnohostena M a zistí sa, ktoré steny (mnohouholníky) rovina α pretala. Vo všeobecnom prípade je prienik roviny a mnohouholníka v $\mathbb{E}^3$ množina úsečiek. Ak ale bude mať viacero stien prienik s rovinou, nestačí ako výstup oznámiť zoznam úsečiek. Treba z nich následne v rovine vytvoriť mnohouholníky a zistiť, ktoré časti sú iba diery a ktoré sú skutočné prieniky. Ďalej budeme uvažovať, že v danej rovine α neleží žiaden vrchol mnohostena M ani žiadna jeho hrana.

Test, či mnohouholník F_i má prienik s rovinou α je pomerne jednoduchý. Stačí zistiť, či všetky vrcholy mnohouholníka ležia len v jednom z polpriestorov určených rovinou. Ak áno, prienik neexistuje. Ak teda niektoré vrcholy sú na jednej strane roviny a niektoré na druhej, existuje množina hrán mnohouholníka F_i, ktorá spája vrcholy na jednej a na druhej strane roviny. Tieto úsečky získame jednoduchým prechodom po hranici mnohouholníka, pričom pre každú hranu testujeme, na ktorej strane roviny sú oba jej koncové body (dosadzovaním súradníc bodov do všeobecnej rovnice roviny $\alpha : ax + by + cz + d = 0$).

Pre všetky tieto hrany treba zistiť ich prienik s rovinou, čo je jednoduchá operácia v konštantnom čase. Výsledkom je množina bodov $\{b_j\}$ (jeden pre každú hranu) v rovine α, ktoré ležia na jednej priamke (lebo mnohouholník leží v nejakej rovine β a úsečky ležia na priamke $\alpha \cap \beta$. Musíme zistiť, medzi ktorými bodmi sa nachádza vnútro mnohouholníka a medzi ktorými vonkajšok. Priamku vieme vyjadriť v parametrickom tvare a podľa hodnoty parametra pre jednotlivé body vieme tieto body na priamke usporiadať. Následne vieme, že vnútro a vonkajšok mnohouholníka sa budú "striedať" pri prechode cez utriedený zoznam bodov (vnútro bude medzi 1. a 2. bodom, potom medzi 3. a 4., 5. a 6. atď).

Takto vytvoríme množinu úsečiek s_k ležiacu v našej rovine a zároveň na povrchu mnohostenu. Treba si ale pri prechode zapamätať vzťahy medzi jednotlivými úsečkami, a to takto: pre hranu ležiacu v rovine si zapamätáme, do ktorého mnohouholníka (steny F_i) patrila a na ktorých hranách pôvodného mnohostena M ležia jej koncové body. Hranu orientujeme tak, aby spolu s normálou roviny α a normálou steny F_i (vonkajšou alebo vnútornou, pričom musí tento výber byť pre všetky steny konzistentný), v ktorej leží, tvorili pravotočivú sústavu súradníc v $\mathbb{E}^3$. Vďaka takejto informácii vieme "cestovať" po vzniknutých hranách a vďaka orientácii vieme v rovine určiť, ktorá hrana je nasledujúca pre



Obr. 6.45. Príklad prieniku mnohostena a roviny

momentálne spracovávanú hranu (a taktiež z ktorej strany je vnútro mnohostena a z ktorej vonkajšok).

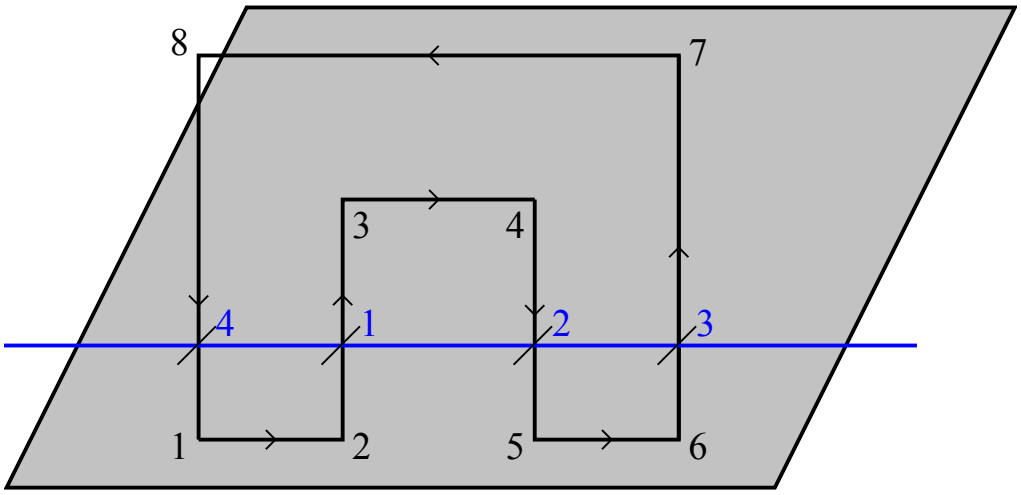
Na vytvorenie sledov úsečiek v rovine α budeme pokračovať nasledovne: Sled si budeme pamätať ako dvojsmerne spájaný zoznam. V každom kroku algoritmu budeme mať niekoľko sledov a pre každý sled S si budeme pamätať informáciu, v ktorom bode S_{start} začína a v ktorom bode S_{end} končí. Na začiatku teda inicializujeme výstupný zoznam sledov hrán na prázdny. Prejdeme cez všetky hrany nachádzajúce sa v rovine a pre každú zistíme, na ktorý sled by sa mohla napojiť. Pre momentálnu hranu H nastávajú 4 možnosti:

(a) V zozname sledov neexistuje sled spĺňajúci $H_{start} = S_{end} \vee H_{end} = S_{start}$. V takomto prípade nemáme žiaden sled, na ktorý by sme hranu napojili, a preto vytvorím nový sled, ktorý bude obsahovať iba túto hranu.

(b) V zozname sledov existuje sled spĺňajúci $H_{start} = S_{end}$, ale neexistuje sled spĺňajúci $H_{end} = S_{start}$. Hranu napojíme na začiatok sledu a upravíme jeho začiatočný bod na začiatočný bod hrany.

(c) V zozname sledov existuje sled spĺňajúci $H_{end} = S_{start}$, ale neexistuje sled spĺňajúci $H_{start} = S_{end}$. Hranu napojíme na koniec sledu a upravíme jeho koncový bod na koncový bod hrany.

(d) V zozname sledov existuje sled spĺňajúci $H_{start} = S_{end} \wedge H_{end} = S_{start}$. Hrana uzatvára tento sled, teda ho vyradíme zo zoznamu momentálnych sledov, lebo je ukončený a môže byť oznámený na výstupe.

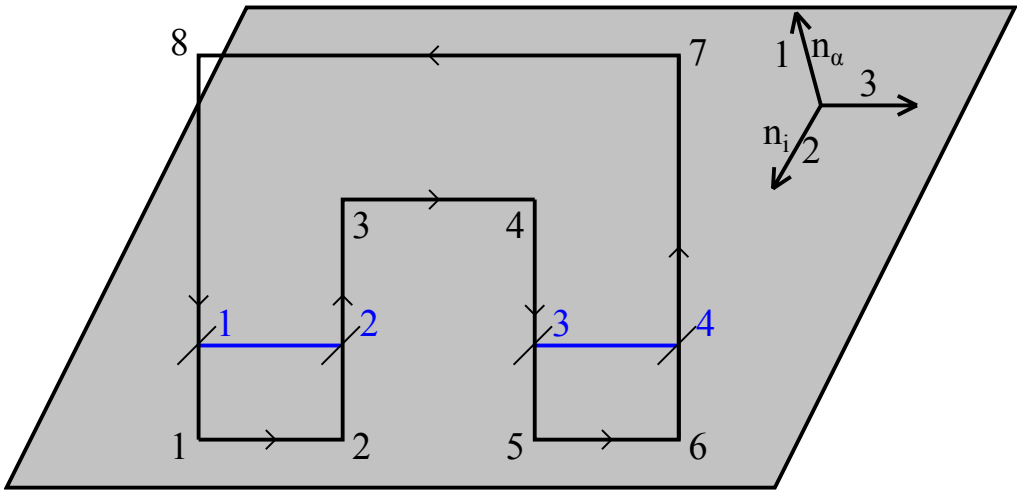


Obr. 6.46. Prienik hranice jednej steny a roviny

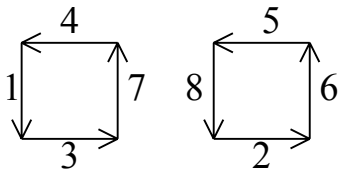
Zložitosť algoritmu bude závisieť od počtu hrán mnohostena M, označme tento počet E. Test a určenie prieniku hrany s rovinou prebehne $\mathcal{O}(E)$ krát. Bodov prieniku vznikne $\mathcal{O}(E)$ - každý leží na jednej z hrán M, môžeme ich podľa toho indexovať. Body pre každú stenu M triedime na priamke, čo celkovo trvá $\mathcal{O}(E \log E)$. Následne pri konštrukcií sledov prechádzame zoznam $\mathcal{O}(E)$ hrán a pre každú hľadáme, na ktorý z existujúcich sledov by sa dala napojiť. Vytvorením dvoch polí o veľkosti E obsahujúcimi informáciu, ktorý sled v bode danej hrany začína resp. končí vieme v čase $\mathcal{O}(1)$ identifikovať, na ktorý sled sa má hrana napojiť. Následné operácie upravujúce sledy budú taktiež v $\mathcal{O}(1)$ ak sledy reprezentujeme ako dvojsmerne spájané zoznamy. Vytváranie sledov preto trvá $\mathcal{O}(E)$. Celková zložitosť algoritmu je teda $\mathcal{O}(E \log E)$.

Teraz si postup ukážeme na jednoduchom príklade. Na obrázku 6.45 vidíme zadaný mnohosten a rovinu. Prienikom sú dva vyznačené obdĺžniky. Aplikujeme teda náš algoritmus. Prvý krok je zistiť, ktoré steny pretínajú rovinu. Prejdeme cez všetky steny a okrem troch z nich (2 spodné a 1 vrchná) majú všetky steny prienik.

Postup identifikovania prienikov hrán si ukážeme na prednej stene. Na obrázku 6.46 vidíme stenu, spolu s očíslovanými vrcholmi (ako sme ich dostali zadané). Podľa toho, na ktorej strane roviny sa nachádzajú body sme identifikovali hrany $(2, 3), (4, 5), (6, 7), (8, 1)$, ktoré pretínajú rovinu v modrých bodoch. Vidno, že použiť poradie, v akom boli modré body vytvorené nebude šikovné, lebo nedostaneme správne úsečky. Na obrázku 6.47 vidno preusporiadané body a z nich skonštruované úsečky. Taktiež je tam zobrazená pravotočivá súradnicová sústava pre tieto úsečky (1. vektor je normála roviny, 2. je normála steny, 3. je smerový vektor úsečky), aby boli vhodne orientované.



OBR. 6.47. Rekonštrukcia úsečiek ležiacich v rovine α a v stene F_i



OBR. 6.48. Orientované úsečky v rovine α ohraničujúce prienik $\alpha \cap M$

Analogický postup zopakujeme pre všetky steny a dostaneme sa k situácii na obrázku 6.48. Množina orientovaných úsečiek v rovine α. Sledy zrekonštruujeme postupným pridávaním hrán. Hrany na obrázku majú úmyselne poprehadzované poradie kvôli lepšej ilustrácii algoritmu.

Postup:

(a) Zoznam sledov je prázdny. Pridáme hranu 1 ako samostatný sled.
(b) Hrana 2 nenadväzuje nijako na jediný sled (hranu 1). Pridáme hranu 2 ako samostatný sled.
(c) Hrana 3 je pokračovaním hrany 1. Napojíme na koniec tohto sledu a upravíme koncový bod.
(d) Hrana 1 je pokračovaním hrany 4. Napojíme hranu 4 na začiatok sledu 1-3.
(e) Hrana 5 nenadväzuje nijako na sledy 2 ani 4-1-3. Pridáme hranu 5 ako samostatný sled.
(f) Hrana 6 je pokračovaním hrany 2 a hrana 5 je pokračovaním hrany 6. Spojíme hranu 6 a sledy 2 a 5 do nového sledu 2-6-5.

(g) Hrana 7 uzatvára sled 4-1-3. Prvý výstupný sled je 4-1-3-7. Zostal nám len sled 2-6-5.
(h) Hrana 8 uzatvára sled 2-6-5. 8-2-6-5 je druhý výstupný sled.

Vidno, že sledy sú plne zrekonštruované v rovine a taktiež majú správnu orientáciu. □

Cvičenie 6.14 (20 bodov). *Napíšte algoritmus prieniku dvoch lichobežníkov, ktorých základne ležia na rovnakých (rovnobežných) priamkach. Ide o čiastkovú úlohu v jednom z algoritmov na počítanie prieniku dvoch konvexných mnohouholníkov.*

Cvičenie 6.15 (15 bodov). *Podrobne opíšte spájanie čiastkových riešení – prienikov dvojíc lichobežníkov – do prieniku dvoch konvexných mnohouholníkov v zametacom algoritme prieniku konvexných mnohouholníkov.*

Literatúra

[BY98] Jean-Daniel Boissonnat and Mariette Yvinec. *Algorithmic geometry*. Cambridge University Press, Cambridge, 1998. Translated from the 1995 French original by Hervé Brönnimann.

[dBvKOS00] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational geometry*. Springer-Verlag, Berlin, revised edition, 2000. Algorithms and applications.

[Eri] Jeff Erickson. Algorithms course materials. `http://compgeom.cs.uiuc.edu/~jeffe/`.

[PS85] Franco P. Preparata and Michael Ian Shamos. *Computational geometry*. Texts and Monographs in Computer Science. Springer-Verlag, New York, 1985. An introduction.