2.2 Prehľadávanie grafov

Majme daný neorientovaný graf G(V, E).
Prehľadávanie grafu je proces prejdenia všetkých vrcholov grafu pozdĺž hrán.

1. Prehľadávanie do hĺbky

Vyšetrovaný vrchol je na vrchu zásobníka a je označený ako starý. Ak tento vrchol má nového suseda *v*, dáme *v* na vrch zásobníka, označíme ho ako starý a vyšetrujeme ho. V opačnom prípade, ak vyšetrovaný vrchol už nemá nových susedov, vyhodíme ho zo zásobníka.
ZAPIS(*v*) – zoznam vrcholov susediacich s vrcholom *v*,
NOVY(*v*) – informácia o tom, či sme už vrchol *v* navštívili, typ boolean.

Algoritmus: Prehľadávanie do hĺbky.

```
Procedure PH(v);
Begin
   NOVY(v) := false;                                  // ozn. v ako nájdený
   Forall u ∈ ZAPIS(v) Do If NOVY(u) Then PH(u);      // postup do hĺbky
End;                                                  // vrchol v je vyšetrený
Begin                                                 // telo programu
   Forall v ∈ V Do NOVY(v) := true;
   Forall v ∈ V Do If NOVY(v) = true Then PH(v);
End.
```

Zložitosť: O(m+n), lebo každú hranu prezrieme 2-krát a pre každý vrchol zbehne procedúra PH práve raz.
Pozn.: Prehľadávanie do hĺbky pre BPS je ekvivalentné s algoritmom PRE_ORDER.

2. Prehľadávanie do šírky

Vyšetrovaný vrchol je označený ako starý a vybrali sme ho z vrchu fronty. Všetkých jeho doteraz neoznačených susedov označíme ako staré a dáme na koniec fronty. Potom prejdeme k ďalšiemu vrcholu, ktorý je na vrchu fronty.

Algoritmus: Prehľadávanie do šírky.

```
Procedure PS(v);
Begin
   fronta := 0; fronta ← v; NOVY(v) := false;
   While fronta ≠ 0 Do Begin
      w ← fronta;
      Forall u ∈ ZAPIS(w) Do If NOVY(u) Then Begin
         fronta ← u; NOVY(u) := false;
      End;
   End;
End;
Begin
   Forall v ∈ V Do NOVY(v) := true;
   Forall v ∈ V Do If NOVY(v) = true Then PS(v);
End.
```

Zložitosť: O(m+n)

Pozn.:
1. Každé volanie procedúry PH resp. PS z tela programu predstavuje vyšetrenie nového komponentu súvislosti. Prostredníctvom týchto algoritmov teda vieme nájsť komponenty súvislosti grafu G.
2. Pomocou uvádzaných algoritmov sa dá zostrojiť kostra prehľadávania do hĺbky, resp. do šírky. Vždy pri nájdení nového vrchola hrana, po ktorej sme do neho prišli, bude hranou kostry. (**Def: Kostra** súvislého grafu je jeho maximálny acyklický podgraf.)

Cvičenie:
1. Navrhnite O(n) algoritmus, ktorý rozhodne, či je daný graf acyklický. *Riešenie:* Prehľadávanie do hĺbky. Ak nájdem hranu do starého vrchola, inú ako spätnú, tak graf má cyklus. Ak nájdem viacej ako n hrán, graf má cyklus. Preto O(n).
2. Ukážte, že nasledovná procedúra PRIESKUM(v) pri vhodnej implementácii prezrie všetky vrcholy grafu v čase O(m+n), pričom prehľadávanie do hĺbky aj do šírky sú jej špeciálnymi prípadmi.

```
Procedure PRIESKUM(v);
Begin
   stav(v) := navštívený; W := {v};
   While existuje vo W nevyšetrený vrchol Do Begin
      v := ľubovoľný navštívený vrchol z W;
      If existuje (u ∈ ZAPIS(v)) and (stav(u) = nový) Then Begin
        W := W∪{u}; stav(u) := navštívený;
      End;
      Else stav(v) := vyšetrený;
   End;
End;
Begin
   Forall v ∈ V Do stav(v) := nový;
   Forall v ∈ V Do If stav(v) = nový Then PRIESKUM(v);
End.
```

2.3 Vyhľadávanie blokov v grafe

Def.: Vrchol grafu sa nazýva **artikulácia**, ak jeho vynechanie z grafu zväčší počet komponent súvislosti. Maximálny podgraf grafu G, ktorý neobsahuje artikuláciu, sa nazýva **blok** (komponent 2-súvislosti).

Na hľadanie komponentov 2-súvislosti (blokov) využijeme algoritmus prehľadávania do hĺbky.

Def.: Nech T je kostra prehľadávania do hĺbky súvislého grafu G s koreňom r. Hrany, ktoré pri prehľadávaní vedú do nových vrcholov, nazývame **priame** (sú to hrany stromu prehľadávania do hĺbky) a ostatné nazývame **obrátené**. Každému vrcholu v priradíme dve čísla:
- **Def(v)** – poradové číslo, v akom bol nájdený algoritmom prehľadávania do hĺbky
- **Low(v)** – minimum z hodnôt Def(u) cez všetky vrcholy u také, že existuje v – u cesta v Grafe G, ktorej všetky hrany s výnimkou poslednej sú priame.

Lema: Nech T je kostra prehľadávania do hĺbky súvislého grafu G. Koreň kostry T je artikuláciou v G práve vtedy, keď má viac ako jedného syna. Vrchol v rôzny od koreňa je artikuláciou práve vtedy, keď pre niektorého z jeho synov neexistuje obrátená hrana spájajúca tohoto syna alebo niektorého jeho potomka s predkom v.
Dôkaz:

1) Nech je koreň r artikuláciou a u je synom r. Keďže prehľadávaním do hĺbky nájdeme kostru v komponente súvislosti G-$\{r\}$ určenom synom u, tak koreň musí mať aspoň dvoch synov.
2) Nech koreň r má aspoň dvoch synov s_1, s_2. Z vrchola s_1, prehľadávaním do hĺbky, sme sa po priamych hranách nedostali do s_2, a teda z s_1 do s_2 existuje len cesta cez vrchol r. To znamená, že r je artikulácia.

Označme T(x) množinu obsahujúcu vrchol x a vrcholy, do ktorých sa dá dostať z x po priamych hranách.

1) Ak vrchol v, rôzny od koreňa, je artikuláciou, tak potom nutne oddeľuje blok, v ktorom sú predkovia v od bloku, v ktorom je niektorý z jeho synov a teda tvrdenie lemy platí. (Keby pre každého zo synov v existovala obrátená hrana spájajúca tohoto syna alebo niektorého jeho potomka s predkom v, potom by všetci potomkovia v z T(v) mali cestu do grafu G-T(v) nevedúcu cez vrchol v, graf G-$\{v\}$ by bol súvislý.)
2) Platí: Ak pre s syna v neexistuje obrátená hrana z T(s) do predka v, potom neexistuje obrátená hrana z T(s) do G-($\{v\} \cup$T(s)).
   Nech existuje obrátená hrana vedúca z vrchola $u \in$T(s) do vrchola $w \in$G-($\{v\} \cup$T(s)), ktorý nie je predkom v. Potom
   - buď Def(w) < Def(s) a teda vrchol u by bol nájdený skôr ako s po hrane (w, u), $\Rightarrow u \notin$T(s), spor
   - alebo Def(w) > Def(y), pre $\forall\ y \in$T(s). Potom by vrchol w bol nájdený po hrane (u, w) a teda $w \in$T(s), spor.

   Dostávame, že v je artikulácia.

Veta: Nech T je kostra prehľadávania do hĺbky súvislého grafu G. Vrchol v rôzny od koreňa je artikuláciou v G práve vtedy, keď pre niektorého jeho syna u platí: Low(u) ≥ Def(v).
Dôkaz: Ide o dôsledok predošlej lemy.

Presná definícia Low(v):

A = min {Low(s); s je synom v}
B = min {Def(w); (v, w) je obrátená hrana v G}
C = Def(v)

Low(v) = min {A, B, C}.

Hodnotu Low(u) získame, keď vyšetríme všetkých potomkov u. Ak po vyšetrení vrchola u zistíme, že Low(u) ≥ Def(v), kde v je rodič u, tak všetky hrany v zásobníku až po hranu (v, u) tvoria blok.

Algoritmus Bloky
Vstup: graf G(V, E), bez izolovaných vrcholov, zadaný zoznamami okolí vrcholov – ZAPIS(v) – pre $\forall\ v \in$V.
Výstup: množiny hrán blokov (2-súvislých komponentov) grafu G.
Pomocné premenné:

- parameter p = rodič vrchola v
- Low, Def, Zásobník, stav (= počet navštívených vrcholov) – globálne premenné

```
Procedure BLOK(v, p);                    // prehľadávanie do hĺbky
begin
   stav := stav + 1;
   Def(v) := stav; Low(v) := stav;
   Forall u ∈ ZAPIS(v) do
      If Def(u) = 0 then begin           // u je nový, (v, u) je priama
         Zásobník.vlož(v, u);
         BLOK(u, v);                     // preskúmame potomkov v.
         Low(v) := min(Low(v), Low(u));
         If Low(u) ≥ Def(v) then         // v je koreň alebo artikulácia
            repeat
               Zásobník.vyber(e); Write(e);
            until (e = (v, u));
      end;
      else                               // u je navštívený, (v, u) je obrátená
      If (u ≠ p) and (Def(u) < Def(v)) then begin
         Zásobník.vlož(v, u);
         Low(v) := min(Low(v), Def(u));
      end;
end;
begin
   Forall v∈V do Def(v) := 0;
   Zásobník := 0; stav := 0;
   Forall v∈V do
      If Def(v) = 0 then BLOK(v, 0);
end.
```

Zložitosť: O(m+n) – algoritmus vyšetruje každý vrchol práve raz a každú hranu práve dvakrát.