

Diskrétne geometrické štruktúry

2.

Martin Florek

florek@sccg.sk

www.sccg.sk/~florek

Binárny prehľadávací strom

- Binárny prehľadávací strom – zložitosť na základe výšky stromu
- Potreba zabezpečiť čo najmenšiu výšku stromu
- Pre kompletný binárny strom s výškou h je počet vrcholov $2^h - 1$
- Cestu môžeme skrátiť až na $\lg(n+1)$
- Potreba vyváženosti v každom vrchole

AVL stromy

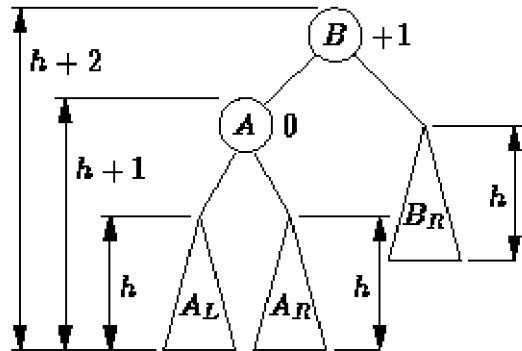
- Adelson-Velskii a Landis
- AVL strom je samovyvažujúci sa BPS
- pre každý jeho vrchol v platí:
 - buď v je list
 - alebo v má jedného syna, ktorý je list
 - alebo v má dvoch synov. Ak ľavý podstrom vrchola v má výšku h_1 a pravý h_2 , potom platí: $|h_1 - h_2| \leq 1$
 - $b(v) = h_1 - h_2$
- $\lg(n+1) \leq h \leq 1,44 * \lg(n+2)$

Vkladanie prvku do AVL stromu

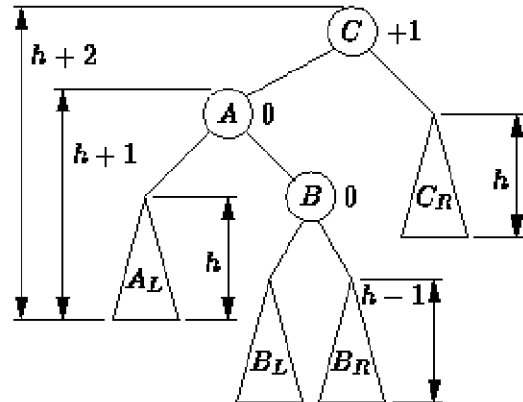
- Najprv Insert pre BPS
- V niektorých vrcholoch môže vzniknúť nevyváženosť
- Vrchol x treba vyvažovať práve vtedy, keď platia nasledujúce dve podmienky:
 - $b(x) = 1$ a nový vrchol je pridaný do ľava od x ,
 - $b(x) = -1$ a nový vrchol je pridaný do prava od x ,
 - pre $\forall y$, ktorý leží na ceste od x ku pridávanému vrcholu platí: $b(y) = 0$

Vyvažovanie

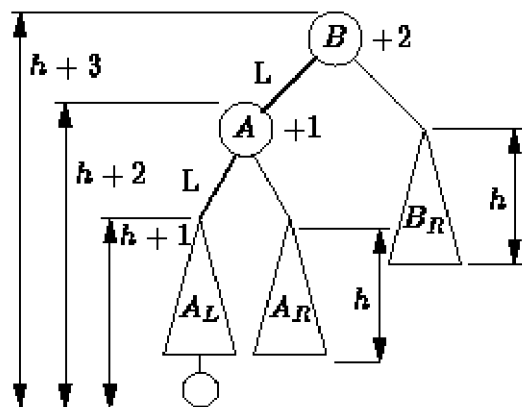
- Podstrom s nevyváženým vrcholom treba preskupiť aby nastala vyváženosť
- Treba zachovať pravidlá BPS
- Treba zachovať výšku podstromu, aby sa nemuseli vyvažovať ostatne vrcholy
- Viaceré konfigurácie - rotácie



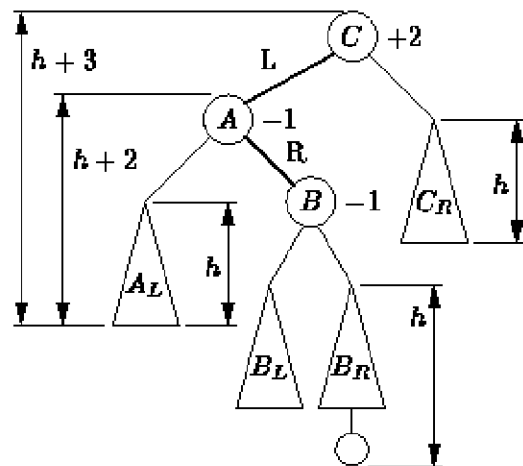
(a)



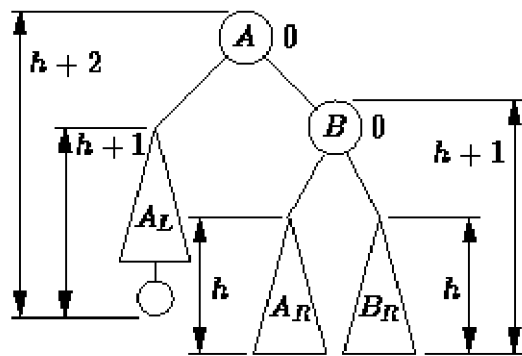
(a)



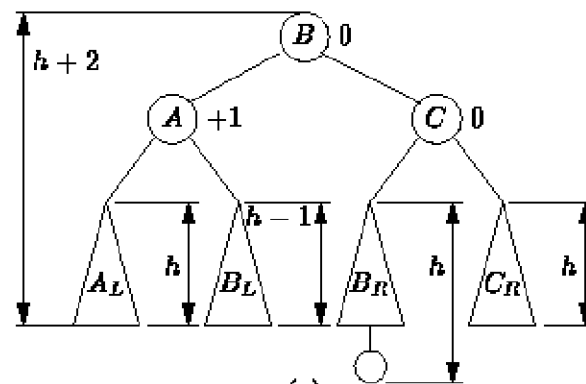
(b)



(b)



(c)



(c)

AVL – algoritmy

```
MODIFY (x, v)
{
    if (v->key < x->key)
        if (x->left != NULL)
            {
                x->b = x->b + 1;
                MODIFY (x->left, v);
            }
    if (v->key > x->key)
        if (x->right != NULL)
            {
                x->b = x->b - 1;
                MODIFY (x->right, v);
            }
}
```

```
struct AVLTree
{
    AVLNode* root;
}
```

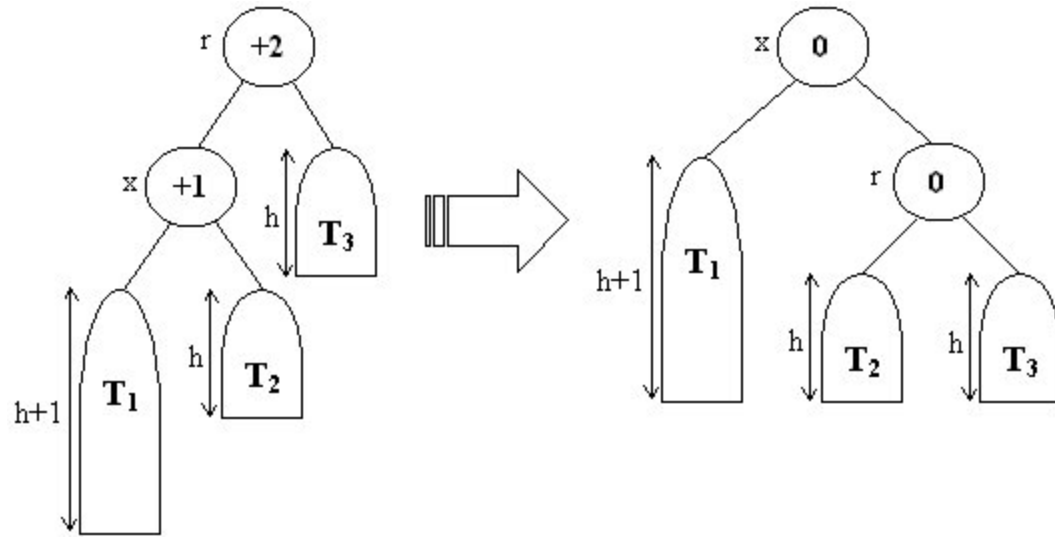
```
struct AVLNode
{
    int key;
    AVLNode* left;
    AVLNode* right;
    AVLNode* parent;
    int b;
    void* data;
}
```

```
AVL_INSERT(T, v)
{
    x = T->root;
    if (T->root == NULL) {T->root = v; v->b = 0;}
    else
        {
            INSERT (T->root, v);
            MODIFY (x, v);
            if (x->b == 2)
                {
                    if (x->left->b == 1) vyvažuj podľa A
                    if (x->left->b == -1) vyvažuj podľa B
                }
            if (x->b == -2)
                {
                    Symetricky ku (b(x) == 2);
                }
        }
}
```

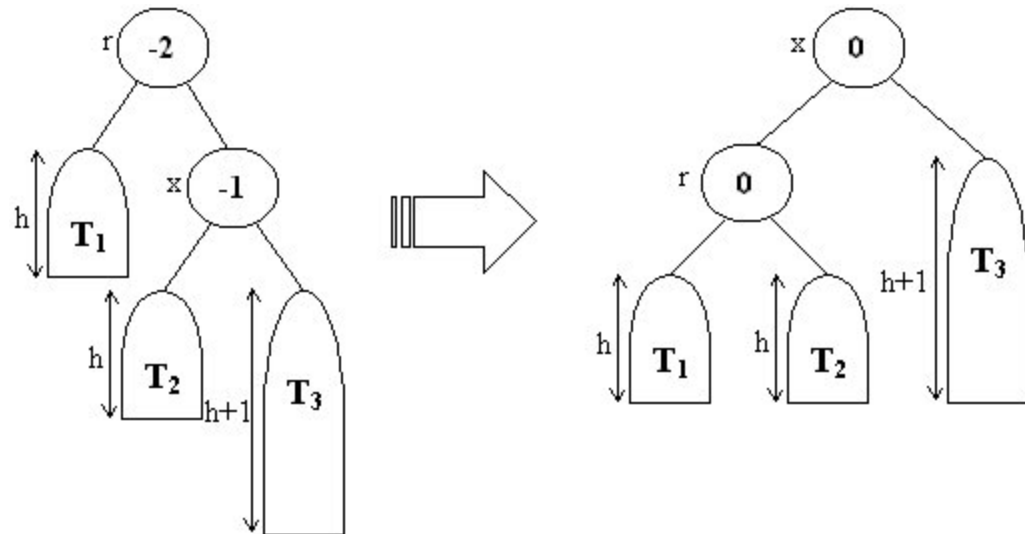
```
INSERT (u, v)
{
    if (v->key < u->key)
        if (u->left == NULL) {u->left = v; if (u->b != 0) x = u;}
        else
            {
                if (u->b != 0) x = u; INSERT (u->left, v);
            }
    if (v->key > u->key)
        if (u->right == NULL) {u->right = v; if (u->b != 0) x = u;}
        else
            {
                if (u->b != 0) x = u; INSERT (u->right, v);
            }
}
```

Rotácie

LL rotation

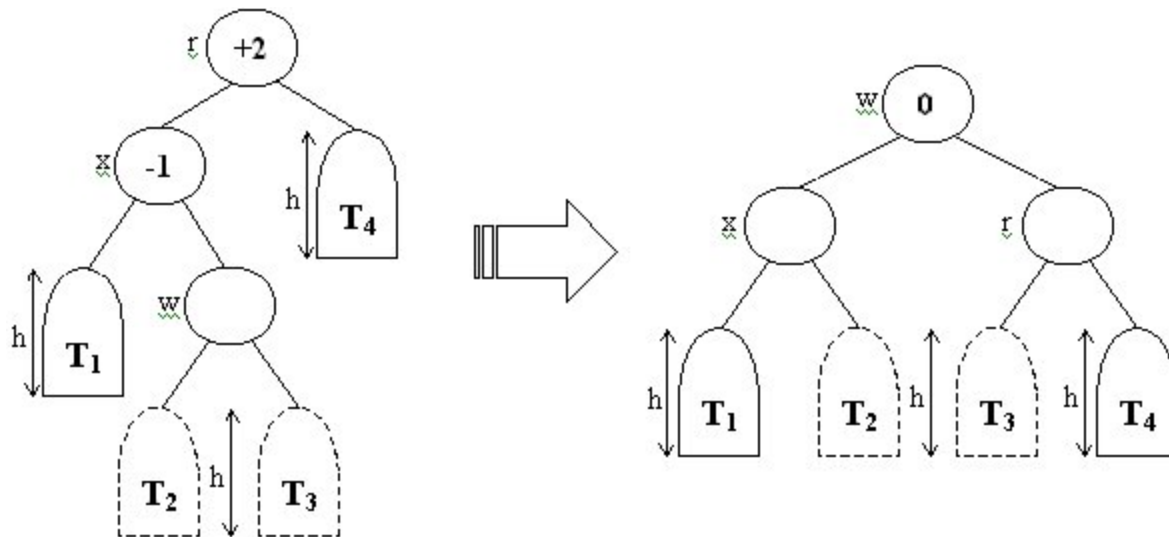


RR rotation

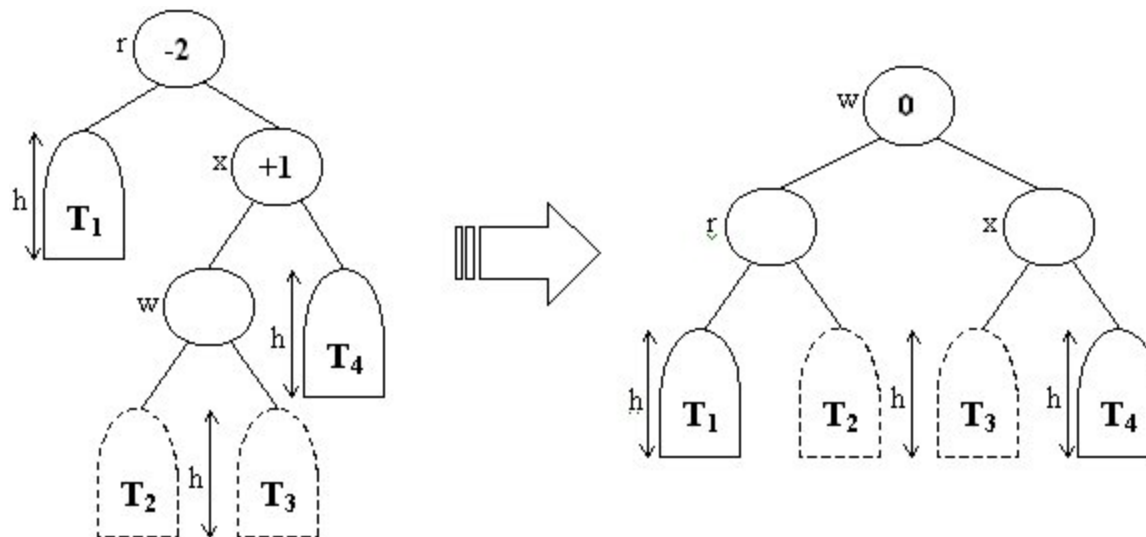


Rotácie 2

LR rotation



RL rotation



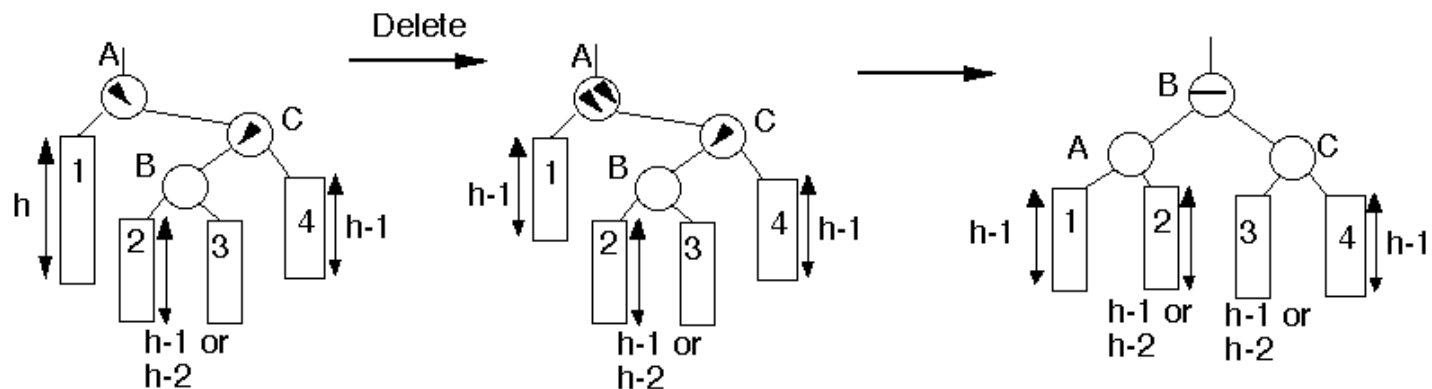
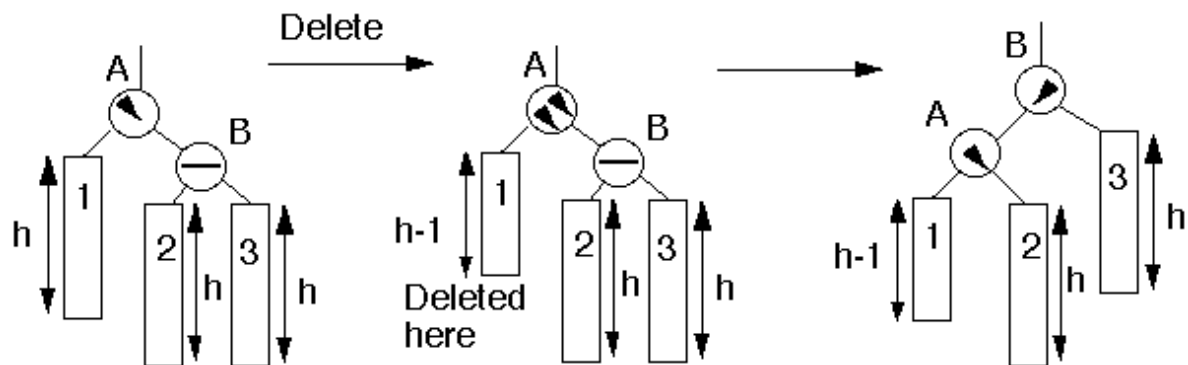
Vymazanie vrcholu z AVL

- Rozšírenie DELETE pre BPS:
- 1) v má najviac 1 syna:
 - vrchol v vynecháme ako pri DELETE pre BPS
 - $x = v \rightarrow \text{parent}$;
 - BALANCE (x, v);
 - delete v ;
- 2) v má dvoch synov:
 - nájdeme maximum v ľavom podstrome vrchola v (MAX)
 - vymeníme v a MAX
 - postupujeme podľa 1)

Balance

```
BALANCE (x, v)
{
    if (x != NULL)
    {
        if ((v->key < x->key) && (x->b ≥ 0))
        {
            x->b = x->b - 1;
            if (x->b == 0) BALANCE (x->parent, v);
        }
        if (v->key > x->key) and (x->b ≤ 0)
        {
            x->b = x->b + 1;
            if (x->b == 0) BALANCE (x->parent, v);
        }
        if (v->key > x->key) and (x->b == 1) situácia A alebo B
        if (v->key < x->key) and (x->b == -1) situácia symetr. ku A alebo B
        if (vyvažovanie A resp. B && x->parent nevyváženost') BALANCE (x->parent, v);
    }
}
```

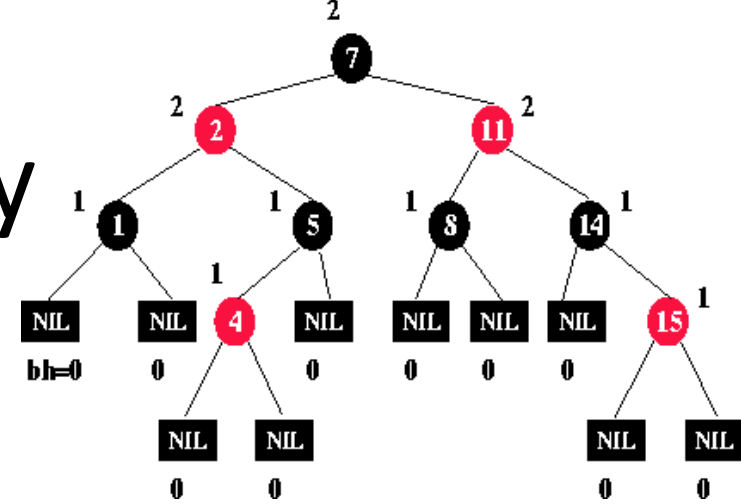
Rotácie 3



Zložitosť pre AVL stromy

- Daná výškou stromu
- Asymptotická zložitosť operácií: $O(\log n)$
- Maximálny počet prechodov: $1.44 * \lg(n)$
- Insert: možná jedna-dve rotácie, priemerne 1 rotácia na 2 vloženia
- Delete: možnosť veľa rotácií pre jedno vymazanie, priemerne 1 rotácia na 5 vymazaní

Červeno-čierne stromy



- Iné pravidlá pre vyváženie:
 1. Každý vrchol je čierny alebo červený
 2. Koreň stromu je čierny
 3. Všetky listy stromu sú čierne
 4. Synovia červeného vrcholu sú čierny
 5. Každá cesta z vrcholu v do nejakého listu má rovnaký počet čiernych vrcholov

Vlastnosti

- $bh(n)$ – počet čiernych vrcholov na ceste z vrchola n do listu
- R-B strom má výšku maximálne $2 \cdot \lg(n+1)$
- Dĺžka najdlhšej cesty z koreňa do listu nie je väčšia ako dvojnásobná dĺžka najkratšej cesty z koreňa do listu
- Na jednej ceste z vrcholu do listu nemôže byť viac červených vrcholov ako čiernych

Vkladanie

- BPS vkladanie
- Vlož vrchol ako červený
- Postupuj od vloženého vrcholu až do koreňa
hľadaj porušenie pravidla 3, 4 a 5
- Ak je porušené pravidlo prefarbuj alebo rotuj v
okolí daného vrchola

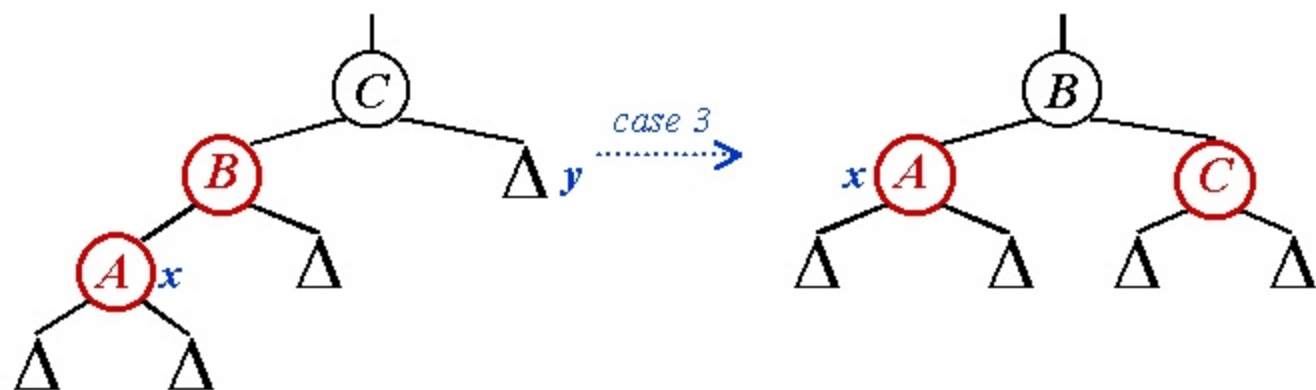
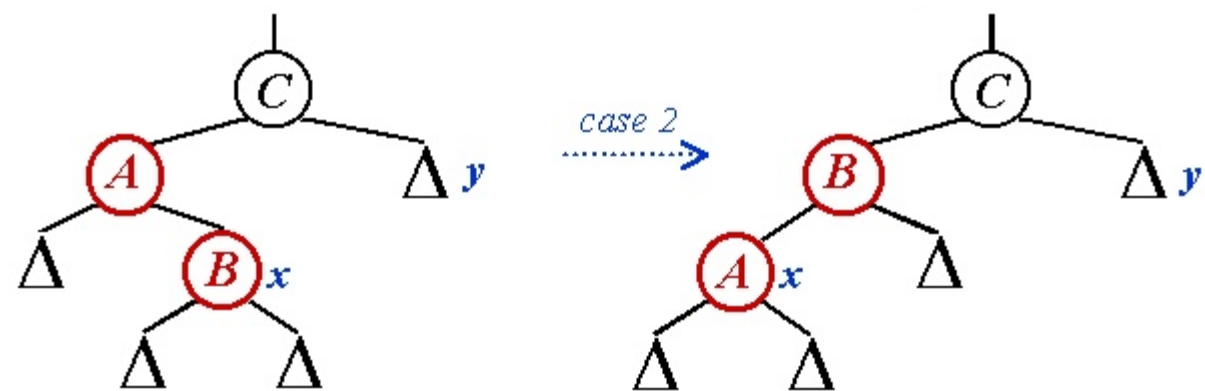
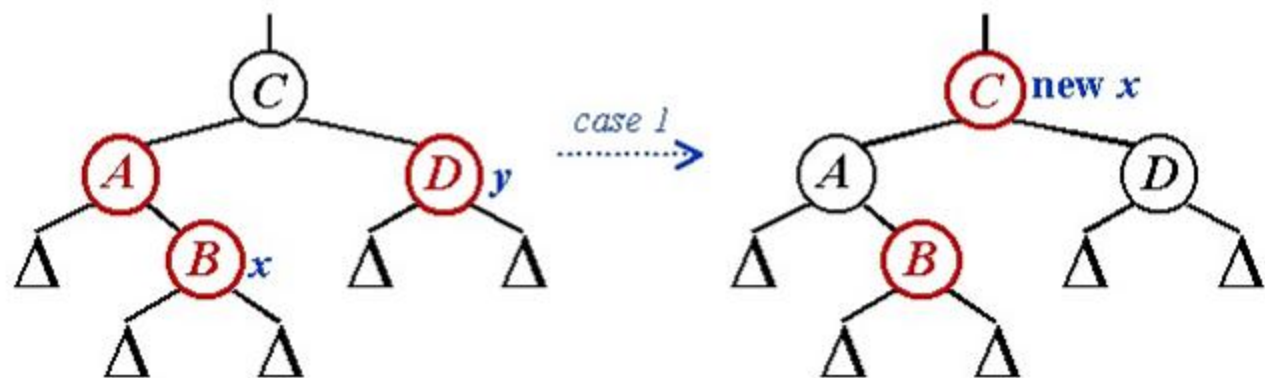
Vkladanie 2

```
RB_INSERT (T, x)
{
    BPS_INSERT(T, x);
    x->color = RED;
    while (x != T->root && x->parent->color == RED)
    {
        if (x->parent == x->parent->parent->left)
        {
            Y = x->parent->parent->right;
            if (y->color == RED)
            {
                x->parent->color = BLACK;
                y->color = BLACK;
                x->parent->parent = RED;
                x = x->parent->parent;
            }
            else
            {
                if (x == x->parent->right)
                {
                    x = x->parent;
                    LeftRotate(x);
                }
                x->parent->color = BLACK;
                x->parent->parent->color = RED;
                RightRotate(x->parent->parent);
            }
        }
        else
        {
            // symetricky, vymenit left a right
        }
    }
}
```

Prípad 1

Prípad 2

Prípad 3



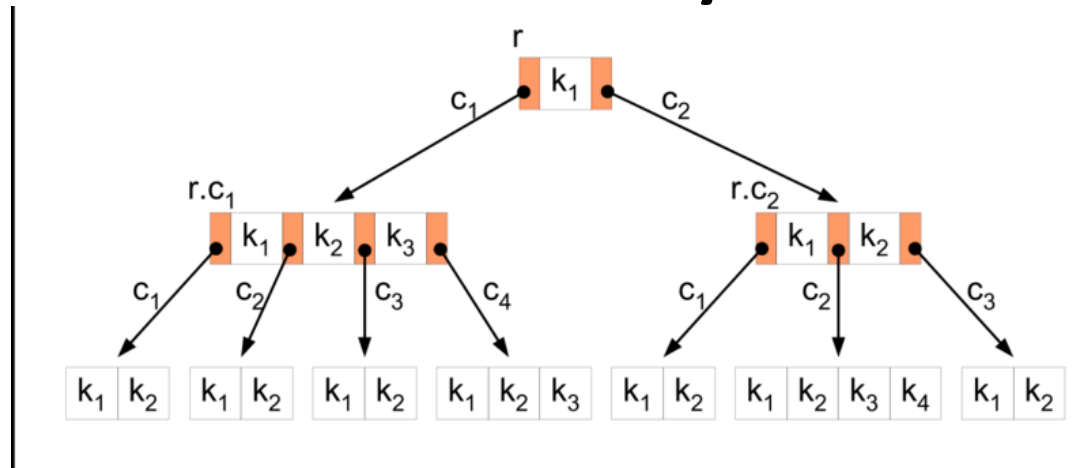
Vymazanie vrcholu z R-B

- Mazanie ako pri BPS
- Ak zmazaný vrchol bol červený, nič sa nedeje
- Ak je vrchol čierny, treba urobiť spätný prechod od mazaného vrcholu do koreňa a opravovať podmienky
- Použitie rotácií a prefarbovania

B-stromy

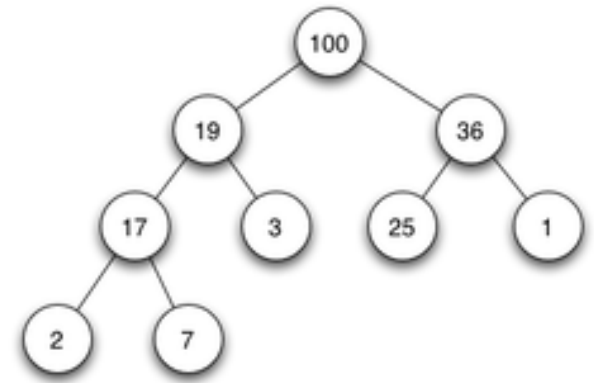
- Každý vrchol obsahuje aspoň n synov
- Každý vrchol obsahuje najviac $2n$ synov
- Všetky listy sú na jednej úrovni
- V každom vrchole je počet kľúčov o 1 menší ako počet potomkov
- n – rád stromu
- Výška stromu – $O(\log_n N)$
- Netreba tak často vyvažovať
- Plytvá miestom – vhodné pre veľké bloky dát

B-stromy 2



- Vkladanie – kontrola na prekročenie limitu $2 \cdot n$
- Mazanie – kontrola na počet potomkov aspoň n
- Náprava – rozdeľovanie a spájanie kľúčov

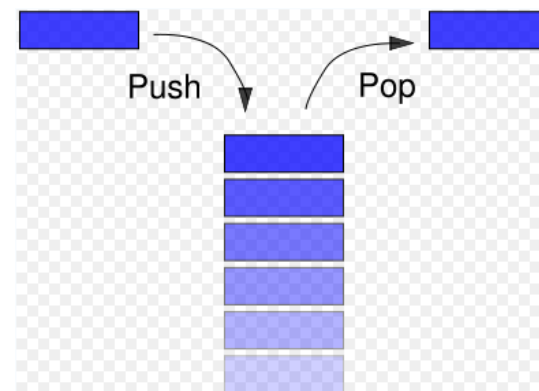
Halda



- Stromová štruktúra
- Ak vrchol B je potomok vrcholu A , tak $\text{key}(A) \geq \text{key}(B)$ ($\text{key}(A) \leq \text{key}(B)$)
- Vytvorenie v $O(n)$
- Použitie:
 - Triedenie
 - Prehľadávanie utriedených množín
 - Grafové algoritmy
 - Prioritná fronta

Zásobník

- Princíp Last In First Out (LIFO), posledný uložený prvok sa vyberie ako prvý
- Operácie (push, pop, peek, size, ...)
- Použitie:
 - spracovanie výrazov
 - manažment pamäte
 - skladanie transformácií



Fronta

- Princíp First In First Out (FIFO)
- Prvky sa vkladajú na jednom konci, vyberajú na druhom
- Vyberá sa vždy „najstarší“ prvok vo fronte
- Prioritná fronta:
 - Pridaj prvok s definovanou prioritou
 - Vyber prvok s najvyššou prioritou

koniec (-:

florek@sccg.sk