Splay stromy

kuko

30.9.2020

Vybrané partie z dátových štruktúr

operácia splay(x)

- nájde x, alebo najbližší väčší, alebo najbližší menší prvok
- „vybuble“ ho nahor až do koreňa

operácia splay(x)

- nájde x, alebo najbližší väčší, alebo najbližší menší prvok
- „vybuble“ ho nahor až do koreňa

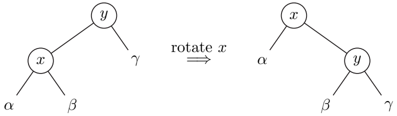
operácia splay(x)

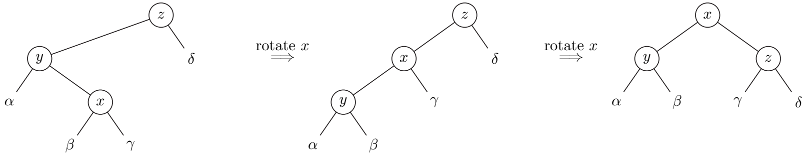
- nájde x, alebo najbližší väčší, alebo najbližší menší prvok
- „vybuble“ ho nahor až do koreňa

- min: splay($-\infty$)
- max: splay(∞)
- *find*(x): splay(x) a pozri koreň

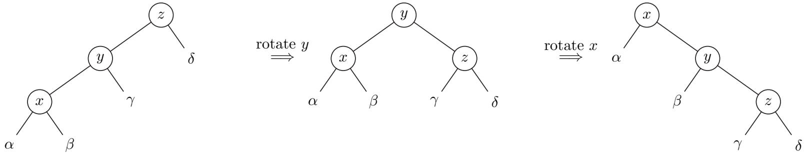
- min: splay($-\infty$)
- max: splay(∞)
- *find*(x): splay(x) a pozri koreň

- min: splay($-\infty$)
- max: splay(∞)
- $\mathit{find}(x)$: splay(x) a pozri koreň


y
x
γ
α
β
rotate x
x
y
α
β
γ



z
y
δ
α
x
β
γ
rotate x
z
x
δ
y
γ
α
β
rotate x
x
y
z
α
β
γ
δ


z
y
x
δ
γ
α
β
rotate y
y
x
z
α
β
γ
δ
rotate x
x
α
y
β
z
γ
δ

$rank(x) = r_x = \lfloor \log_2 size(x) \rfloor$

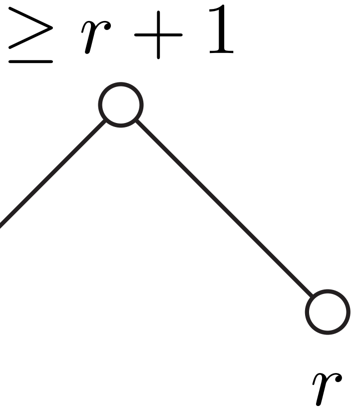
INVARIANT: Každý vrchol stromu má našetrených $rank(x)$\$.

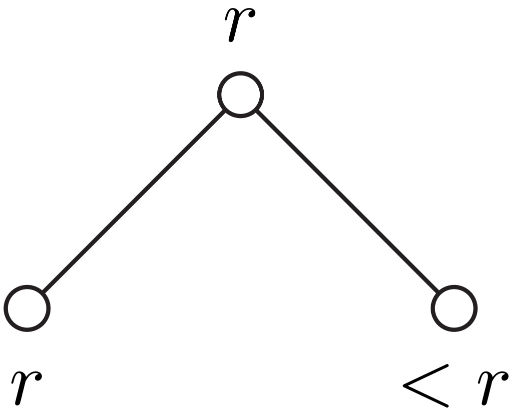
$$\Phi(T) = \sum_x rank(x)$$

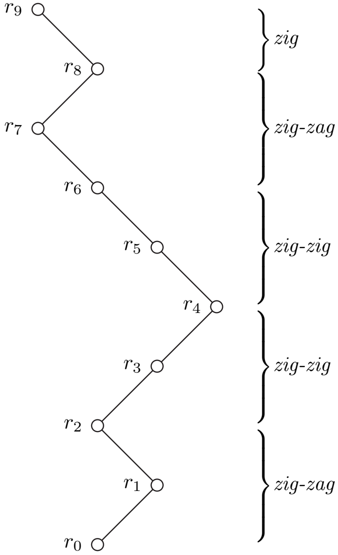
$rank(x) = r_x = \lfloor \log_2 size(x) \rfloor$

INVARIANT: Každý vrchol stromu má našetrených $rank(x)$\$.

$$\Phi(T) = \sum_x rank(x)$$


$\geq r+1$
r
r

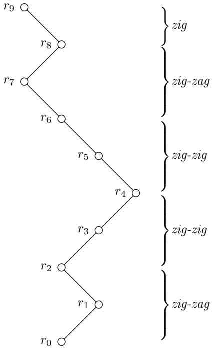

r
r
$< r$



r_9
r_8
r_7
r_6
r_5
r_4
r_3
r_2
r_1
r_0
zig
zig-zag
zig-zig
zig-zig
zig-zag

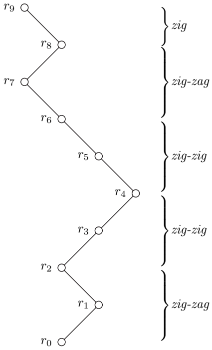
Veta

Na operáciu splay(x) stačí $3(r_{root} - r_x) + 1\$$, *pričom zachováme invariant. Z toho vyplýva, že každý splay trvá* $O(\log n)$ *amortizovane.*



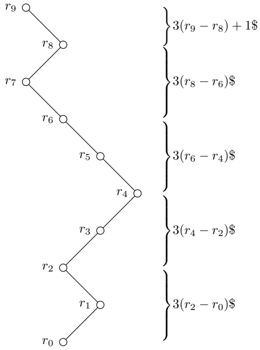
Lema

Na každú dvojrotáciu stačí $3(r_{p(p(x))} - r_x)\$$*, na poslednú rotáciu stačí* $3(r_{p(x)} - r_x) + 1\$$.



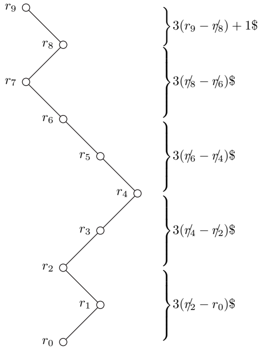
Lema

Na každú dvojrotáciu stačí $3(r_{p(p(x))} - r_x)\$$, *na poslednú rotáciu stačí* $3(r_{p(x)} - r_x) + 1\$$.

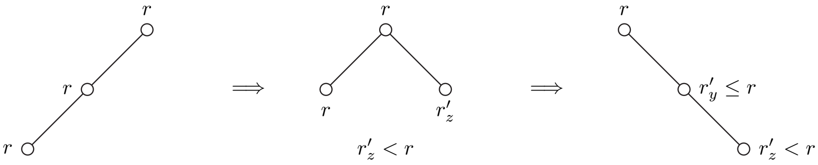


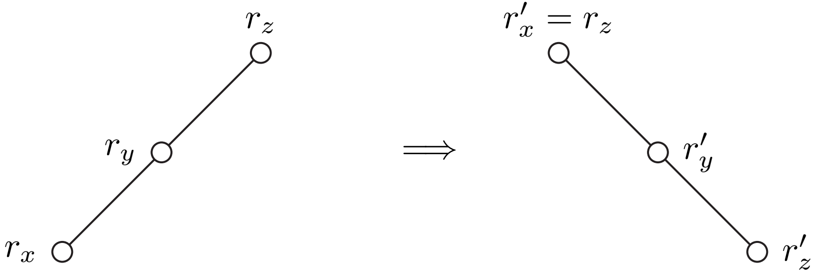
Lema

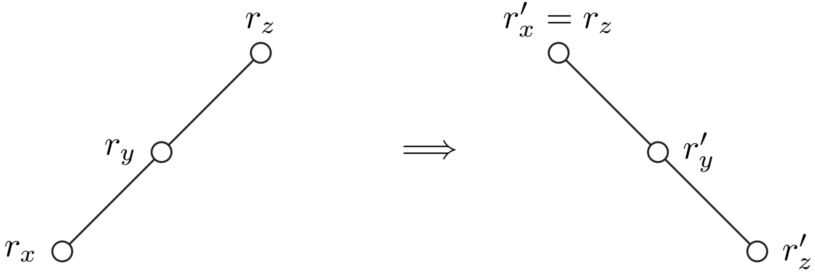
Na každú dvojrotáciu stačí $3(r_{p(p(x))} - r_x)\$$, *na poslednú rotáciu stačí* $3(r_{p(x)} - r_x) + 1\$$.



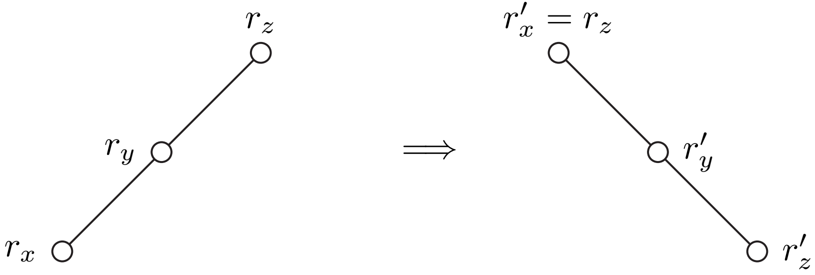
1. cik-cak
2. cik-cik
3. cik


r
r
r
$\Longrightarrow$
r
r
r'_z
$r'_z < r$
$\Longrightarrow$
r
$r'_y \leq r$
$r'_z < r$

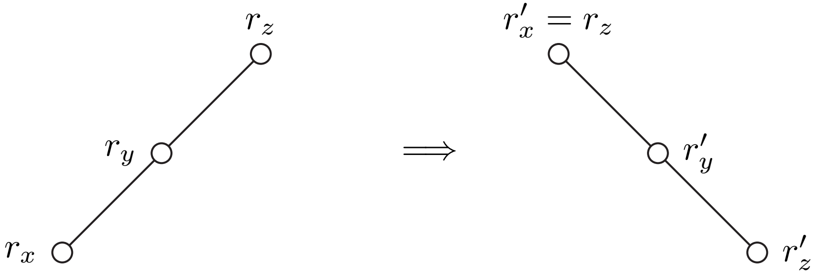

r_z
r_y
r_x
$\Longrightarrow$
$r'_x = r_z$
r'_y
r'_z



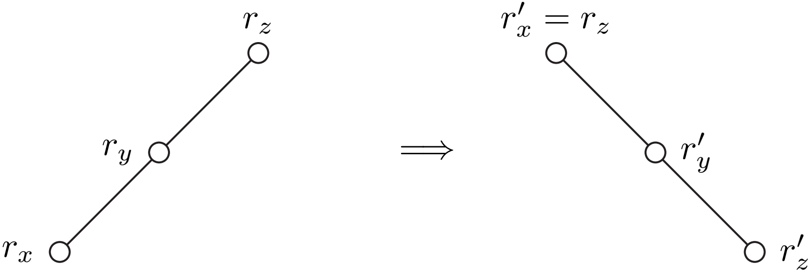
$$(r'_x + r'_y + r'_z) - (r_x + r_y + r_z)$$



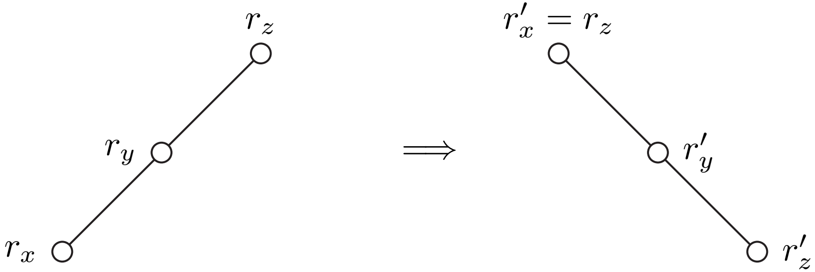
$$(\cancel{r'_x} + r'_y + r'_z) - (r_x + r_y + \cancel{r_z}) = r'_y + r'_z - r_x - r_y$$



$$(\cancel{r'_x} + r'_y + r'_z) - (r_x + r_y + \cancel{r_z}) = \underbrace{r'_y}_{\leq r'_x} + \underbrace{r'_z}_{\leq r'_x} - r_x - r_y$$



$$(\cancel{r'_x} + r'_y + r'_z) - (r_x + r_y + \cancel{r_z}) = \underbrace{r'_y}_{\leq r'_x} + \underbrace{r'_z}_{\leq r'_x} - r_x - \underbrace{r_y}_{\geq r_x}$$



$$(\cancel{r'_x} + r'_y + r'_z) - (r_x + r_y + \cancel{r_z}) = \underbrace{r'_y}_{\le r'_x} + \underbrace{r'_z}_{\le r'_x} - r_x - \underbrace{r_y}_{\ge r_x}$$

$$\le 2(r'_x - r_x) = 2(r_z - r_x)$$

- váha $w_x \in \mathbb{R}^+$
- size s_x je súčet váh v podstrome x
- rank $r_x = \log s_x$
- potenciál $\Phi = \sum r_x$

Veta (O prístupe)

Pre ľubovoľnú voľbu váh má splay(x) *amort. zložitosť*

$$3(r_{root} - r_x) + 1 = O(1 + \log(W/w_x))$$

kde $W = \sum w_x$ *je celková váha stromu.*

Lema

Každý cik-cik/cik-cak prípad má amort. zložitosť $3(r'_x - r_x)$, *posledný prípad cik má zložitosť* $3(r'_x - r_x) + 1$.

Veta (O prístupe)

Pre ľubovoľnú voľbu váh má splay(x) amort. zložitosť

$$3(r_{root} - r_x) + 1 = O(1 + \log(W/w_x))$$

kde $W = \sum w_x$ *je celková váha stromu.*

Lema

Každý cik-cik/cik-cak prípad má amort. zložitosť $3(r'_x - r_x)$*,*
posledný prípad cik má zložitosť $3(r'_x - r_x) + 1$*.*

- prípad „cik-cik“:
  - $a = t + \Delta\Phi = 2 + (r'_x + r'_y + r'_z) - (r_x + r_y + r_z) = 2 + r'_y + r'_z - r_x - r_y \leq 2 + r'_x + r'_z - 2r_x$
  - $r_x + r'_z = \log s_x + \log s'_z \leq 2\log[(s_x + s'_z)/2] \leq 2\log(s'_x/2) = 2r'_x - 2$
  - teda $r'_z \leq 2r'_x - r_x - 2$, dosadíme vyššie:
  - $a \leq 2 + r'_x + (2r'_x - r_x - 2) - 2r_x = 3(r'_x - r_x)$
- prípad „cik-cak“:
  - $a = 2 + r'_y + r'_z - r_x - r_y \leq 2 + r'_y + r'_z - 2r_x$
  - $r'_y + r'_z = \log s'_y + \log s'_z \leq 2r'_x - 2$
  - $a \leq 2 + 2r'_x - 2 - 2r_x = 2(r'_x - r_x)$

Dôsledky

Veta (Vyváženosť)

Amortizovaná zložitosť splay(x) *je* $O(\log n)$.

Dôkaz. $w_x = 1$ □

Veta (Vyváženosť)

Amortizovaná zložitosť splay(x) je $O(\log n)$.

■ **Dôkaz.** $w_x = 1$ □

Veta (Statická optimálnosť)

$f_x \geq 1$ – frekvencia splay(x); m – celkový počet operácií; $p_x = f_x/m$
Zložitosť m splay operácií je

$$O(m + \sum f_x \log \frac{m}{f_x}) = O(m + m \sum p_x \log \frac{1}{p_x}).$$

splay(x) má amortizovanú zložitosť $O(\log(1/p_x))$.

Dôkaz. $w_x = f_x$ □

Veta (Statická optimálnosť)

$f_x \geq 1$ – frekvencia splay(x); m – celkový počet operácií; $p_x = f_x/m$
Zložitosť m splay operácií je

$$O(m + \sum f_x \log \frac{m}{f_x}) = O(m + m \sum p_x \log \frac{1}{p_x}).$$

splay(x) má amortizovanú zložitosť $O(\log(1/p_x))$.

■ **Dôkaz.** $w_x = f_x$ □

Veta (O statickom prste)

Nech p je fixný vrchol (prst).
splay(x) trvá amortizovane

$$O(\log |x - p|)$$

Dôkaz. $w_x = 1/(x - p + 1)^2$ □

Veta (O statickom prste)

Nech p je fixný vrchol (prst).
splay(x) trvá amortizovane

$$O(\log |x - p|)$$

■ **Dôkaz.** $w_x = 1/(x - p + 1)^2$ □

Veta (O pracovnej množine)

- $t_i(x)$ – *#rôznych prvkov (vrátane* x*), ktoré sme splayovali odkedy sme naposledy vysplayovali* x *pred časom* i

splay(x) *trvá amortizovane*

$$O(\log t_i(x_i))$$

Dôkaz. $w_x = 1/t_i(x)^2$ □

Veta (O pracovnej množine)

- $t_i(x)$ – #rôznych prvkov (vrátane x), ktoré sme splayovali odkedy sme naposledy vysplayovali x pred časom i

splay(x) trvá amortizovane

$$O(\log t_i(x_i))$$

■ **Dôkaz.** $w_x = 1/t_i(x)^2$ □

Ďalšie vlastnosti splay stromov

Veta (O skenovaní)

Splay $1, 2, 3, \ldots, n$ *trvá* $O(n)$.

Veta (O dynamickom prste)

splay(x) *trvá amortizovane*

$$O(\log|x_i - x_{i-1}|)$$

Hypotéza (O obojsmernej fronte)

Deque-ové operácie trvajú $O(1)$ amortizovane v splay strome.
Najlepšie dokázané: $\alpha(m)$.

Hypotéza (O split strome)

Čas $n\times$ split je $O(n)$.
Najlepšie dokázané: $n\alpha(n)$.

Hypotéza (Zjednotená hypotéza)

splay(x) *trvá amortizovane*

$$O(\log \min_{y}[t_i(y) + |x_i - y| + 2])$$

Hypotéza (Dynamická optimálnosť)

Splay strom je len konštantný násobok od najlepšieho možného BST algoritmu, ktorý pozná celú postupnosť prístupov dopredu.