# Splay strom

Splay stromy sú podľa mňa jedna z najzázračnejších dátových štruktúr, aké kedy uvidíte.

Klasické vyvažované stromy si pamätajú dodatočné informácie, nakoľko sú jednotlivé podstromy nevyvážené a vždy keď sa strom príliš odchýli od ideálneho tvaru, snažia sa ho naprávať, vyvažovať. Naproti tomu, splay strom pracuje úplne „naslepo“ – nemá úplne žiadnu dodatočnú informáciu. Nevie, ktoré časti stromu sú perfektne vybalansované a ktoré sú nakrivo. Napriek tomu dosahuje logaritmickú amortizovanú zložitosť.

## Splayovanie

Kľúčovou operáciou splay stromu je – prekvapujúco – operácia splay($x$). Všetky ostatné operácie (find, insert, delete, atď.) budeme implementovať pomocou splay.

Splay($x$) sa začína klasickým vyhľadávaním: začneme v koreni, ak sme našli $x$, končíme, ak je hodnota vo vrchole príliš veľká, ideme vľavo, ak je príliš malá, ideme vpravo, kým sa dá. Na konci buď nájdeme vrchol $x$, alebo skončíme vo vrchole, ktorého hodnota je buď najbližšia menšia alebo najbližšia väčšia ako $x$. Tento vrchol následne „vybubleme“ až do koreňa stromu. Tento proces prebieha pomocou sérií rotácií, ale nie hocijakých: používame špeciálne pravidlá, ktoré zabezpečia dobrú amortizovanú časovú zložitosť.

Nech $y$ je rodič vrcholu $x$ a $z$ jeho starý rodič (ak existuje). Rozlišujeme tri možné situácie:

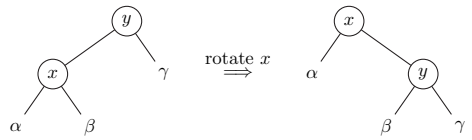
1. Prípad „cik“: Ak vrchol $x$ nemá starého rodiča, teda jeho rodič $y$ je už koreň, vykonáme jedinú rotáciu, ktorá dostane $x$ do koreňa (pozri obr. 1a). Tento prípad nastáva najviac raz, na konci bublania, keďže $x$ sa dostane do koreňa, splayovanie sa končí.

2. Prípad „cik-cik“: Ak $x$ aj $y$ sú obaja ľavými synmi (alebo naopak obaja pravými), hovoríme o prípade „cik-cik“ (pozri obr. 1b). V takomto prípade spravíme rotáciu najprv na vrchole $y$, a až potom na $x$.

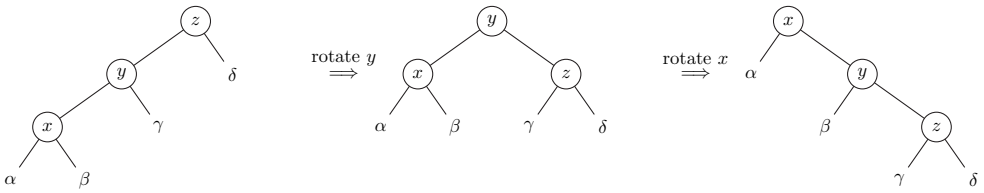
3. Prípad „cik-cak“: Ak $x$ je ľavý syn a $y$ pravý (alebo symetricky, $x$ je pravý a $y$ ľavý syn svojho otca), máme situáciu „cik-cak“ (pozri obr. 1c). Vtedy urobíme dve rotácie $x$. Najskôr sa $x$ posunie na miesto svojho rodiča, a následne ešte vyššie, na miesto starého rodiča. Výsledkom je, že $x$ sa posunie o dva úrovne vyššie a cesta $x$–$y$–$z$ sa otočí.

V prípade „cik-cik“ aj „cik-cak“ spravíme dve rotácie a vrchol $x$ sa dostane o 2 úrovne vyššie. Proces opakujeme, až kým sa $x$ nedostane do koreňa.

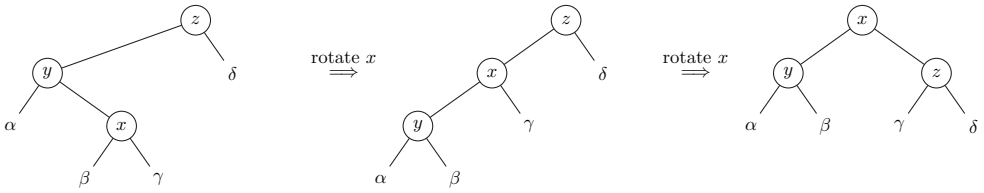
Na obrázkoch 2 a sú dve ukážky, ako prebieha splayovanie vrchola na konkrétnom strome.




(a) Prípad „cik“: Ak je otec $x$ koreň, jednoduchou rotáciou dostaneme $x$ do koreňa a končíme.




(b) Prípad „cik-cik“: Cesta $x$–$y$–$z$ je dvakrát vpravo ako na obrázku (alebo symetricky dvakrát vľavo). Rotujeme najskôr $y$, potom $x$. Výsledkom je, že cesta $x$–$y$–$z$ sa otočí.



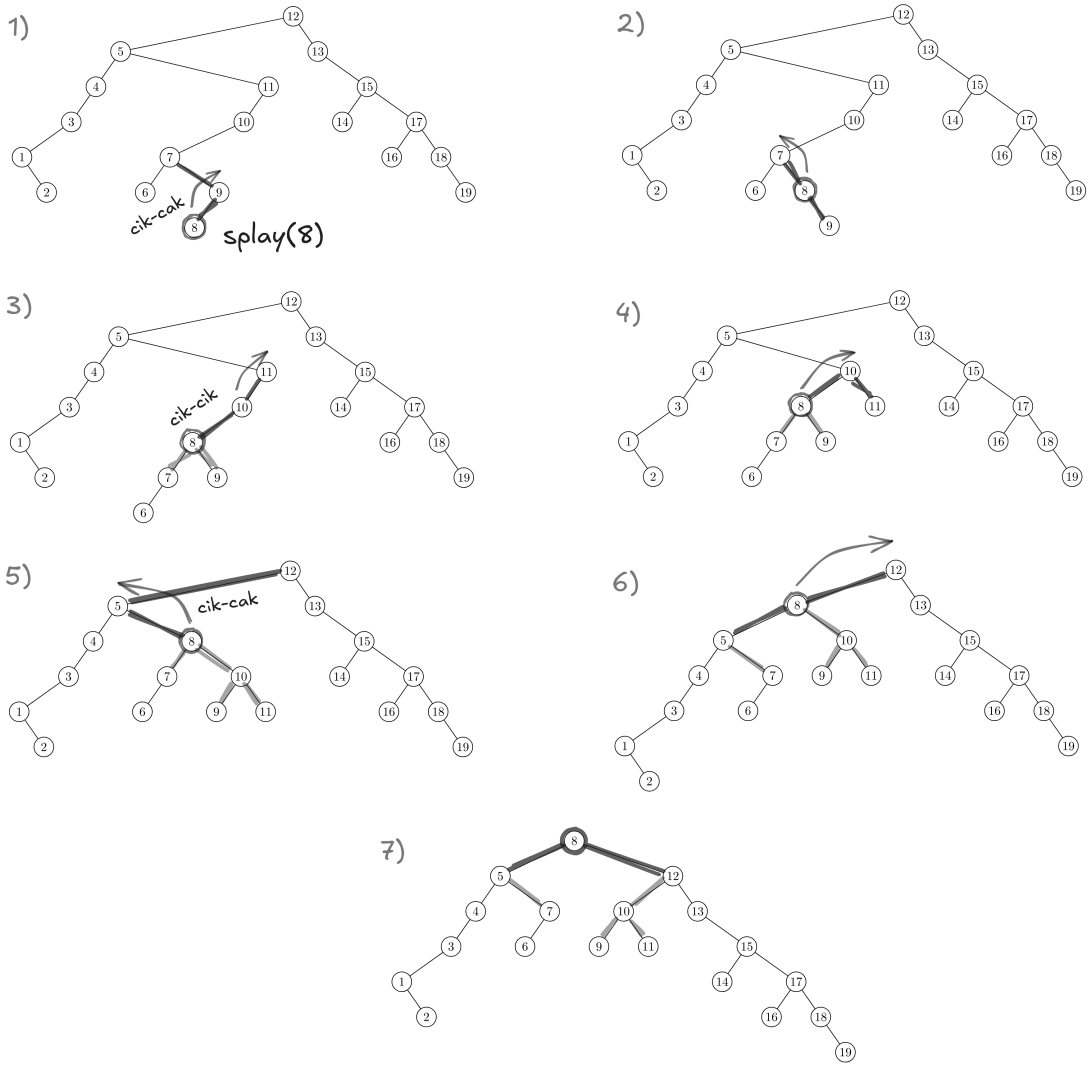

(c) Prípad „cik-cak“: Cesta $x$–$y$–$z$ je cik-cakovito doľava–doprava ako na obrázku (alebo symetricky doprava–doľava). V tomto prípade dvakrát zrotujeme $x$; výsledkom je, že predkovia $y$ a $z$ skončia ako dve deti $x$.

Obr. 1: Pri splayovaní rozlišujeme tri rôzne situácie („cik“, „cik-cik“, „cik-cak“) a podľa situácie volíme, ktoré vrcholy rotujeme.

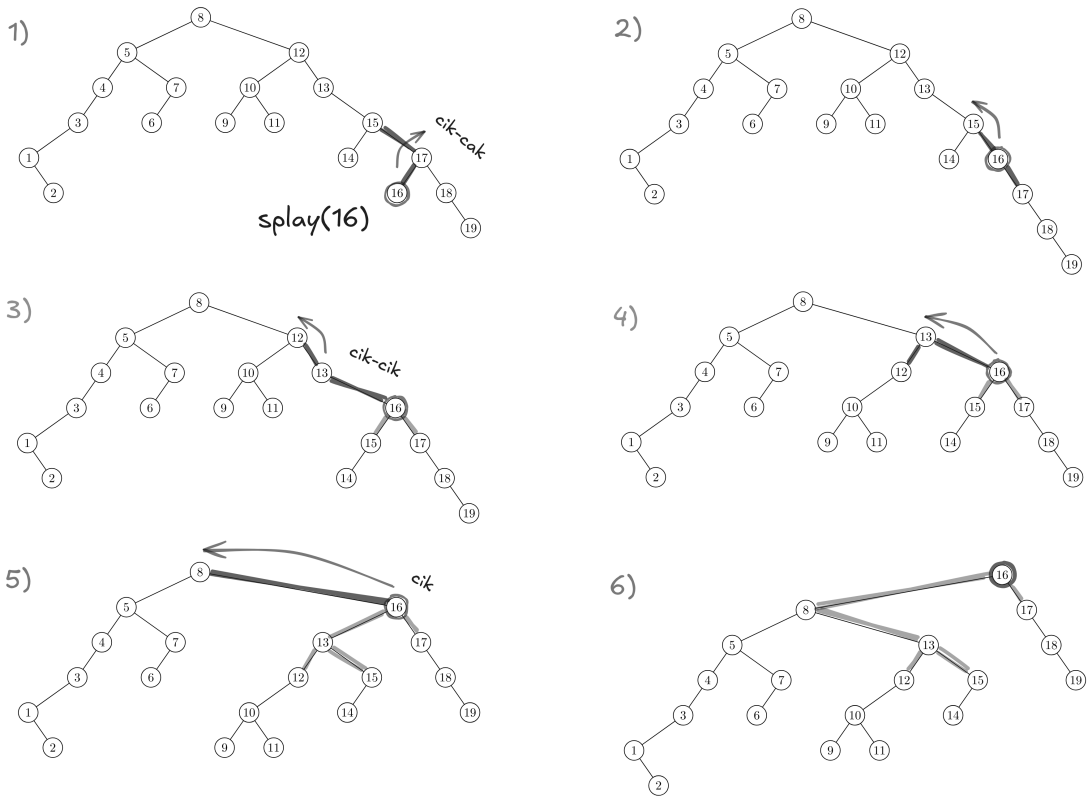
## Ostatné operácie

Všetky operácie v splay strome sa implementujú jednoducho, pomocou operácie splay:

- `find`($x$): Vysplayujeme $x$ a pozrieme sa na koreň – ak je tam $x$, našli sme ho; v opačnom prípade sa v strome nenachádza.
- `min`: Vysplayujeme $-\infty$. Do koreňa sa dostane najbližší väčší prvok, čo je minimum celého stromu.
- `max`: Vysplayujeme $+\infty$. Do koreňa sa dostane najbližší menší prvok, čo je maximum celého stromu.
- `split`($x$): Chceme rozdeliť strom na dve časti – jeden strom s prvkami $\leq x$ a jeden strom s prvkami $> x$. Stačí vysplayovať $x$, čím sa do koreňa

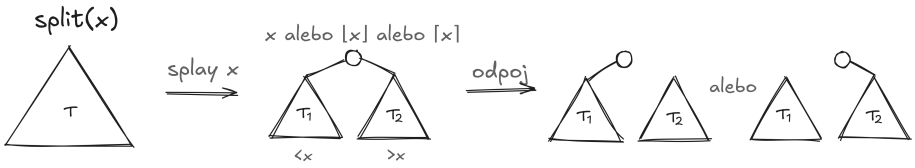



Obr. 2: Splayovanie vrcholu 8 prebehne tromi dvoj-rotáciami: 1) Cesta 8—9—7 je „cik-cak“, takže dvakrát zrotujeme 8. 3) Cesta 8—10—11 je „cik-cik“, takže najskôr zrotujeme 10, potom 8. 5) 8—5—12 je opäť „cik-cak“, takže dvakrát zrotujeme 8. 7) Na konci je vrchol 8 v koreni. Všimnite si na prvom obrázku cestu z 8 do koreňa: 8—9—7—10—11—5—12. Táto cesta sa postupne rotáciami transformovala na podstrom vyznačený šedou na poslednom obrázku.

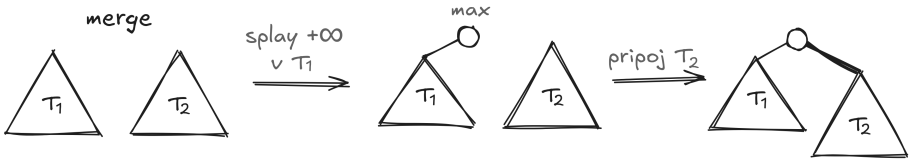



Obr. 3: Splayovanie vrcholu 16: Cesta do koreňa 16—17—15—13—12—8 sa skladá z 16—17—15, čo je „cik-cak“, 15—13—12, čo je „cik-cik“ a posledného kroku 12—8 „cik“. Takto 16 prebubleme až do koreňa.

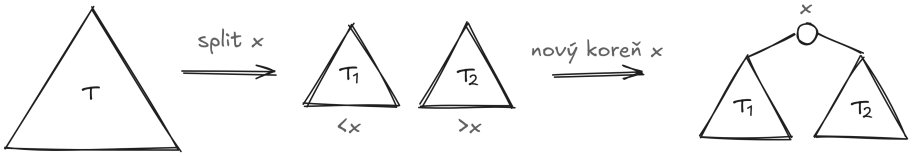
dostane samotné $x$, alebo najbližšia menšia alebo najbližšia väčšia hodnota. Celý ľavý podstrom koreňa bude $< x$, celý pravý podstrom koreňa bude $> x$ a samotný koreň porovnáme s $x$ a pripojíme na správnu stranu. Stačí zmazať jedinú hranu: od koreňa k ľavému alebo pravému synovi.



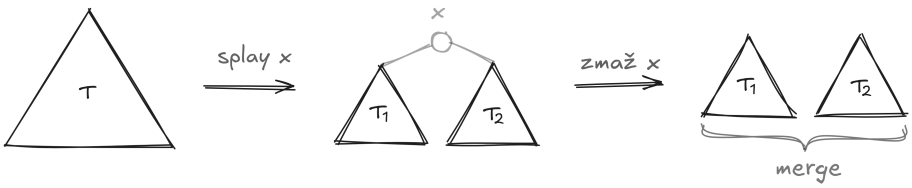

- `merge`$(T_1, T_2)$: Máme dané dva stromy spolu s garanciou, že všetky prvky v $T_1$ sú menšie ako všetky prvky v $T_2$ a chceme ich spojiť do jedného stromu. Riešenie je elegantné: v $T_1$ vysplayujeme $+\infty$. Tým sa do koreňa dostanem maximum a koreň *nebude* mať pravého syna. Tým pádom $T_2$ môžeme jednoducho pripojiť ako pravého syna koreňa.




- `insert`$(x)$: Stačí spraviť `splay`$(x)$. Ak je $x$ v koreni, strom už hodnotu obsahuje a môžeme skončiť. V opačnom prípade rozdelíme strom ako pri operácii `split` na strom $T_1$ s prvkami $< x$ a $T_2$ s prvkami $> x$. Ako výsledok vrátime strom, kde $x$ je nový koreň a jeho ľavý a pravý syn sú $T_1$ a $T_2$.




- `delete`$(x)$: Stačí vysplayovať $x$, čím sa $x$ dostane do koreňa, ktorý vymažeme. Ostanú nám jeho dva podstromy, $T_1$ a $T_2$, na ktoré zavoláme `merge`.



## Jednoduchá analýza

V tejto časti si dokážeme, že splay stromy majú logaritmickú amortizovanú zložitosť. Budeme sa sústrediť iba na analýzu samotnej operácie splay, keďže všetky ostatné operácie vieme implementovať pomocou zopár volaní splay a zvyšná práca je konštantná.

Pre zjednodušenie predpokladajme, že v strome sa nachádzajú kľúče $1, 2, \ldots, n$ a že vždy splayujeme kľúč, ktorý v strome naozaj je.

Pre každý vrchol $x$ si zavedme nasledovné označenia:
$size(x) = s_x$ je počet vrcholov v podstrome zakorenenom vo vrchole $x$,
$rank(x) = r_x = \lfloor \lg s_x \rfloor$ je hodnota odvodená od veľkosti podstromu.
Budeme udržiavať nasledovný invariant:

> Každý vrchol má na svojom „účte“ nasporených práve $r_x$\$.

Spomeňme najskôr niekoľko postrehov o rankoch.

- Pri rotácii vrcholu $x$ s jeho otcom $y$ sa menia iba ich ranky – $r_x$ a $r_y$. Všetky ostatné ranky zostávajú nezmenené.
- Zároveň platí, že po rotácii má vrchol $x$ rovnaký rank, aký mal pred rotáciou jeho otec: $r'_x = r_y$.
- Listy majú veľkosť podstromu $s_x = 1$, a teda rank $r_x = 0$. Všeobecne platí, že rank otca je vždy aspoň taký veľký ako rank jeho syna. Najväčší rank dosahuje koreň, kde $r_{\text{koreň}} = \lfloor \lg s_{\text{koreň}} \rfloor = \lfloor \lg n \rfloor$.
- Vrchol s rankom $r$ má pod sebou aspoň $2^r$ vrcholov.
- Ak majú dvaja bratia rovnaký rank $r$, ich otec musí mať rank aspoň $r+1$, pretože pod ním sa nachádza minimálne $2^{r+1}$ vrcholov.
- Ak má otec a jeho syn rovnaký rank $r$, potom druhý syn musí mať rank prísne menší než $r$. Inak by to odporovalo predchádzajúcemu tvrdeniu.

Ešte predtým ako sa pustíme do formálneho dôkazu, skúsim ponúknuť dva intuitívne dôvody, prečo by mal byť čas splayovania logaritmicky.

**Intuícia I.** Predstavme si cestu od vrcholu $x$ až ku koreňu. Rank $r_x$ môže nadobúdať iba hodnoty z množiny $0, 1, \ldots, \lfloor \lg n \rfloor$, takže pozdĺž tejto cesty sa môže zvýšiť nanajvýš $\lg n$-krát.

Ak sa v niektorom kroku splayovania rank vrcholu zvýši, prácu zaplatíme z peňazí, ktoré sú na operáciu splay priamo pridelené. Naopak, ak rank zostáva rovnaký, tento krok je síce „pomalší“, no ukážeme si, že vrcholy majú na účte vždy dostatočné úspory, aby náklady pokryli. Z pohľadu amortizovanej analýzy sú teda tieto kroky v podstate zadarmo.

**Intuícia II.** Čím je strom horšie vyvážený, tým viac má našetrené, naopak, čím lepšie je vybalansovaný, tým menej potrebuje mať našetrené. Našetrené doláre sú vlastne taký fond opráv; keď sa strom stáva horšie vyvážený, treba prispieť viac, naopak, ak ho trochu napravíme, uvoľnia sa nám nejaké zdroje, ktorými môžeme extra prácu zaplatiť.

Predstavme si, že $y$ je otec a $z$ starý otec vrcholu $x$, pričom $\alpha, \beta$ sú podstromy pod $x$ a $\delta, \gamma$ podstromy pod $y$ a $z$ (ako napríklad v prípade „cik-cik“, pozri obr. 1b).

Ak platí $r_z > r_x$, znamená to, že podstromy $\delta, \gamma$ sú aspoň približne také veľké ako $\alpha, \beta$. Pri hľadaní sme teda vylúčili podstatnú časť kľúčov a rýchlo (Úplne presne toto tvrdenie nemusí platiť kvôli zaokrúhľovaniu $\lfloor \cdot \rfloor$, no ak sa rank na ceste zvýši aspoň o 2, určite sme tým vylúčili konštantný zlomok všetkých kľúčov.)

Naopak, ak $r_z = r_x$, znamená to, že podstromy $\delta, \gamma$ sú v porovnaní s $\alpha, \beta$ malé – väčšina vrcholov je teda sústredená práve pod $x$. V tomto prípade sa však vyváženosť stromu zlepší: po rotáciách sa celé podstromy $\alpha, \beta$ posunú vyššie, a väčšina prvkov pod $x$ tak bude mať menšiu hĺbku než predtým.

**Veta 1 (O prístupe).** *Na operáciu* splay$(x)$ *postačí* $3(r_{koreň} - r_x) + 1\$$*, pričom invariant ostáva zachovaný: každý vrchol $x$ má na účte nasporených $r_x\$$. Zjavne* $r_{koreň} = \lfloor \lg n \rfloor$ *a* $r_x \geq 0$*, takže na operáciu* splay$(x)$ *stačí* $3 \lg n + 1\$$.

**Lema 1.** *Označme $r'_x$ rank vrcholu $x$ po jednom kroku splayovania (teda po jednej dvojitej rotácii alebo po záverečnej jednoduchej rotácii).*

*Potom na každý prípad „cik-cik“/„cik-cak“ stačí* $3(r'_x - r_x)\$$*, a na posledný prípad „cik“* $(r'_x - r_x) + 1\$$.

Z lemy priamo plynie veta o prístupe: keď zosumarizujeme náklady všetkých krokov v rámci splay$(x)$, dostaneme teleskopickú sumu

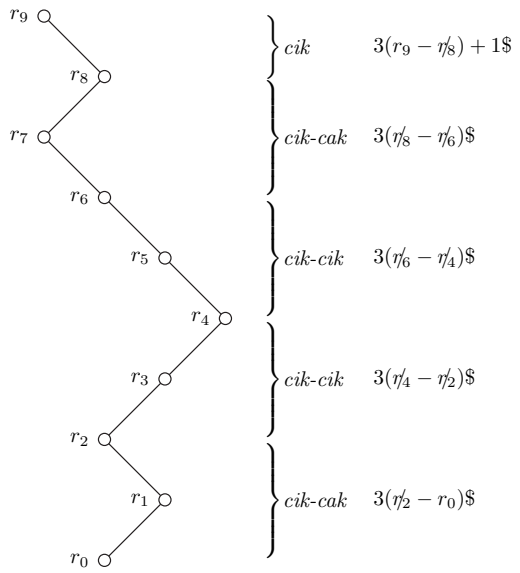
$$\begin{aligned} 3(r_x^{\text{posledný}} &- \cancel{r_x^{\text{predposl.}}} + \cancel{r_x^{\text{predposl.}}} - \cdots + \cancel{r''_x} - \cancel{r'_x} + \cancel{r'_x} - \cancel{r_x} + \cancel{r_x} - r_x) + 1 \\ &= 3(r_x^{\text{posledný}} - r_x) + 1, \end{aligned}$$

pričom $r_x^{\text{posledný}} = r_{\text{koreň}}$ je rank koreňa po skončení splayovania.

■ **Dôkaz lemy.** Budeme analyzovať jednotlivé prípady. Nech $x$ je vrchol, ktorý splayujeme, a $y$, $z$ sú jeho otec a starý otec. Označme $r_x, r_y, r_z$ ranky týchto vrcholov *pred* rotáciami a $r'_x, r'_y, r'_z$ ranky *po* príslušnej (dvojitej alebo záverečnej jednoduchej) rotácii. Ranky všetkých ostatných vrcholov sa pri danom kroku nemenia. S týmto zápisom teraz rozoberieme prípady „cik“ a „cik–cik“ (dôkaz pre posledný prípad „cik-cak“ je podobný a prenechávame ho ako cvičenie čitateľom).

**Prípad „cik“:** Pred rotáciou má dvojica $(x, y)$ uložených spolu $r_x + r_y$ dolárov (ostatné vrcholy ignorujeme, ich ranky sa nemenia). Po rotácii chceme zachovať invariant, takže spolu musia mať $r'_x + r'_y$ dolárov. Okrem toho potrebujeme aspoň 1\$ na zaplatenie samotnej rotácie. Požadovaný prídel je teda

$$(r'_x + r'_y) - (r_x + r_y) + 1.$$




Obr. 4: Dôkaz Vety o prístupe priamo vyplýva z Lemy 1. Ak označíme ranky vrcholov od splayovaného vrcholu ku koreňu $r_0, r_1, r_2, \ldots$, tak podľa lemy stačí, ak na každú dvojrotáciu dostaneme $3(r_{k+2} - r_k)\$$, teda $3\times$ rozdiel rankov vrcholu a jeho starého otca, a na poslednú „cik“ rotáciu stačí $3(r_{k+1} - r_k) + 1\$$. Tieto rozdiely sa pozdĺž cesty ku koreňu nasčítajú tak, že všetky prostredné členy vypadnú a ostane rozdiel medzi rankom koreňa a rankom splayovaného vrcholu $+1$.

Platí, že po rotácii sa $x$ posunie na pozíciu otca, takže $r'_x = r_y$, a zároveň $r'_x \leq r'_y$ (rank syna nie je väčší než rank otca). Preto

$$(r'_x + r'_y) - (r_x + r_y) + 1 = r'_y - r_x + 1 \leq (r'_x - r_x) + 1.$$

Teda na prípad *cik* stačí $(r'_x - r_x) + 1$ dolárov.

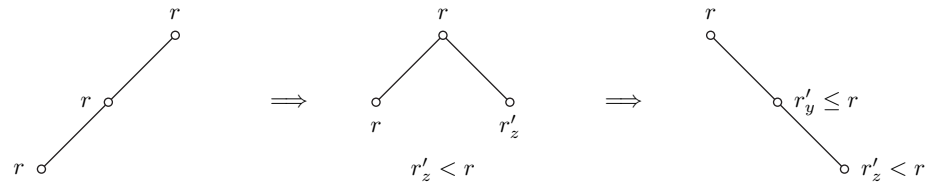
**Prípad „cik-cik“:** Rozlíšime dve situácie podľa vzťahu $r_x$ a $r_z$.

**a)** $\boldsymbol{r_x = r_z}$**.** V tomto prípade $x$, $y$, aj $z$ majú všetky rovnaký rank $r$ a keďže na konci sa $x$ dostane na miesto $z$ pred rotáciami, $r'_x = r_z$. Takže platí $r_x = r_y = r_z = r'_x = r$ a na tento krok máme pridelených $3(r'_x - r_x) = 0$ dolárov. To znamená, že amortizovane by mal byť tento krok „zadarmo“ – nedostaneme naň žiadne financie a všetko treba uhradiť z našetrených peňazí.

Pozrime sa na priebeh: najskôr zrotujeme $y$, takže rank $x$ sa nemení a $y$ sa stáva koreňom daného podstromu namiesto $z$, takže $r'_y = r_z = r$. Čo vieme povedať o ranku $z$? Spomeňme si na vyššie spomínanú vlastnosť

ranku, že ak otec a jeden jeho syn majú rovnaký rank, potom druhý syn musí mať ostro menší rank. To znamaná, že $r'_z < r$!

Následne zrotujeme $x$; rank $z$ sa už nemení, rank $x$ na konci bude rovnaký ako rank $z$ na začiatku, teda $r$. Rank $y$ na konci nevieme povedať presne, môže ostať rovnaký, môže sa zmenšiť – určite bude niekde medzi $r'_z$ a $r$. Každopádne však pred dvojitou rotáciou mala trojica $(x, y, z)$ spolu $3r$ dolárov a po rotáciách jej stačí *menej*. Minimálne vrcholu $z$ sa znížil rank a teda všetku prácu v tomto kroku môžeme zaplatiť z úspor vrcholu $z$ a invariant zostane zachovaný.



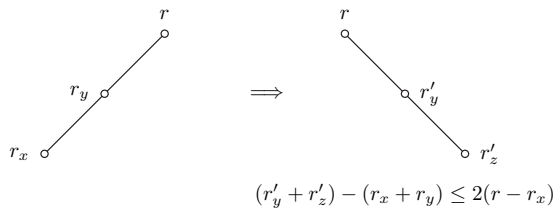

**b)** $\boldsymbol{r_x < r_z}$**.** Pred rotáciami má trojica spolu $r_x+r_y+r_z$ dolárov, po rotáciách potrebuje $r'_x + r'_y + r'_z$, takže na zaplatenie tohto kroku potrebujeme 1\$ na prácu, plus rozdiel rankov, aby sme dorovnali našetrené financie:

$$(r'_x + r'_y + r'_z) - (r_x + r_y + r_z) + 1\$.$$

Platí $r'_x = r_z$ (rank koreňa podstromu ostáva rovnaký; $z$ je koreň pred rotáciami, $x$ po nich). Ďalej $r_x \leq r_y \leq r_z$ a $r'_x \geq r'_y \geq r'_z$. Odtiaľ jednoduchými úpravami dostaneme

$$\begin{aligned}(r'_x + r'_y + r'_z) - (r_x + r_y + r_z) + 1 &= (r'_y - r_x) + (r'_z - r_y) + 1 \\ &\leq (r'_x - r_x) + (r'_x - r_x) + 1 \\ &= 2(r'_x - r_x) + 1.\end{aligned}$$

To znamená, že na zaplatenie tohto kroku potrebujeme $2(r'_x - r_x) + 1$ dolárov. Avšak keďže v tomto prípade predpokladáme $r_x < r_z$ a $r_z = r'_x$, rozdiel $r'_x - r_x$ je aspoň 1, a teda potrebujeme $2(r'_x - r_x) + 1 \leq 3(r'_x - r_x)$ dolárov – čo sme chceli dokázať. Ak to zhrnieme: Na tento krok je vyčlenených $3(r'_x - r_x)$ dolárov: z $(r'_x - r_x) \geq 1$ zaplatíme samotné operácie a zvyšných $2(r'_x - r_x)$ použijeme na dorovnanie úspor tak, aby invariant ostal v platnosti.



**Prípad „cik-cak“:** Dôkaz je veľmi podobný prípadu „cik-cik“, takže ho prenechávame ako cvičenie čitateľom. Opäť treba rozlíšiť dve situácie: a) ranky $x$, $y$, aj $z$, sú všetky rovnaké – toto sú tie kroky „navyše“, kedy sa po ceste ku koreňu nezvyšuje rank avšak analýzou rankov pred a po rotáciach zistíme, že v tomto prípade je na konci strom lepšie vyvážený a má menší súčet rankov ako pred rotáciami a teda túto časť cesty môžeme zaplatiť z našetrených peňazí. Prípad b) je, že $r_x < r_z$ – rank na ceste ku koreňu stúpne. Takýchto krokov môže byť len logaritmicky veľa. Analýzou rankov pred a po rotáciach, s využitím vhodných rovností a nerovností medzi rankami sa dá dokázať, že z pridelených $3(r'_x - r_x)$ použiť 1$ na zaplatenie odvedenej práce a $2(r'_x - r_x)$ nám stačí na zachovanie invariantu – potrebujeme vlastne dokázať, že dvomi rotáciami sa síce môže strom stať menej vyvážený, ale nie príliš a do „fondu opráv“ stačí prispieť $2(r'_x - r_x)$ dolárov. □

## Všeobecná analýza

Priraďme každému vrcholu $x$ kladnú váhu $w_x \in \mathbb{R}^+$. Nech $s_x$ označuje *váhu* podstromu $x$, t.j. súčet váh všetkých vrcholov v tomto podstrome. Rank vrcholu definujeme tentokrát ako

$$\mathrm{rank}(x) = r_x = \lg s_x,$$

teda ako logaritmus váhy podstromu bez dolnej celej časti – rank teraz môže byť ľubovoľné, aj záporné, reálne číslo.

Rovnako ako vyššie budeme udržiavať nasledujúci invariant: každý vrchol bude mať na účte našetrených $r_x$\$. Inými slovami, celkový potenciál stromu bude

$$\Phi = \sum_x r_x = \sum_x \lg s_x.$$

**Veta 2 (Zovšeobecnená veta o prístupe).** *Operácia* splay$(x)$ *má amortizovanú zložitosť* $3(r_{koreň} - r_x) + 1$ *pre ľubovoľnú voľbu váh* $w_x \in \mathbb{R}^+$. *Inými slovami, ak* $W = \sum_x w_x$ *je celková váha stromu, potom zložitosť* splay$(x)$ *je* $3(\lg W - \lg w_x) + 1 = O(1 + \lg(W/w_x))$

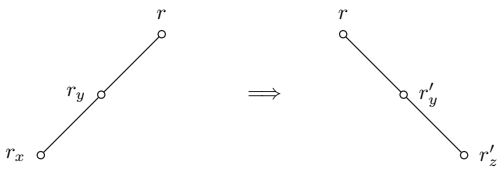
**Lema 2.** *Každý cik-cik/cik-cak prípad má amortizovanú zložitosť* $3(r'_x - r_x)$, *posledný prípad cik má zložitosť* $3(r'_x - r_x)+1$ *(pre ľubovoľnú voľbu váh* $w_x \in \mathbb{R}^+$*).*

■ **Dôkaz lemy.** Amortizovaný čas je skutočný čas plus zmena potenciálu:

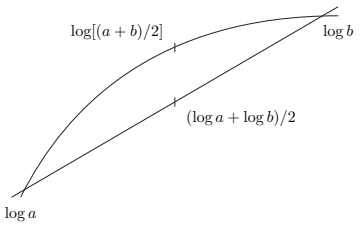
$$T_{\mathrm{amort}} = T_{\mathrm{skutočný}} + \Delta\Phi.$$

**Prípad „cik-cik“:** Skutočný čas sú 2 rotácie, rozdiel potenciálov $\Delta\Phi$ je

$$\begin{aligned}(r'_x + r'_y + r'_z) - (r_x + r_y + r_z) &= r'_y + r'_z - r_x - r_y \\ &\leq r'_x + r'_z - 2r_x\end{aligned}$$




Využijeme konkávnosť logaritmickej funkcie:




priemer logaritmov je menší alebo rovný logaritmu priemeru, takže sŭčet logaritmov (2× priemer) je menší alebo rovný 2× logaritmu priemeru:

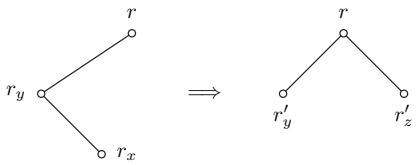
$$r_x + r'_z = \lg s_x + \lg s'_z \leq 2\lg[(s_x + s'_z)/2] \leq 2\lg(s'_x/2) = 2r'_x - 2,$$

teda $r'_z \leq 2r'_x - r_x - 2$, dosadíme vyššie:

$$\Delta\Phi \leq r'_x + (2r'_x - r_x - 2) - 2r_x = 3(r'_x - r_x) - 2.$$

**Prípad „cik-cak“:** Amortizovaný čas je

$$T_{\text{amort}} = T_{\text{skutočný}} + \Delta\Phi = 2 + r'_y + r'_z - r_x - r_y \leq 2 + r'_y + r'_z - 2r_x.$$




Z konkávnosti logaritmu vyplýva

$$r'_y + r'_z = \lg s'_y + \lg s'_z \leq 2r'_x - 2,$$

a po dosadení dostávame

$$T_{\text{amort}} \leq 2 + 2r'_x - 2 - 2r_x = 2(r'_x - r_x)$$

**Prípad „cik“:** Prenechávame čitateľovi. □

### Dôsledky

**Vyváženosť:** Pre $w_x = 1$ dostávame $r_{\text{koreň}} = \lg n$, takže amortizovaná zložitosť je $O(\lg n)$.

**Statická optimálnosť alias Veta o entropii:** Nech $f_x \geq 1$ je frekvencia, s ktorou splayujeme $x$, $m$ je celkový počet operácií, $p_x = f_x/m$; potom zložitosť $m$ operácií je $O(m + \sum f_x \lg(m/f_x)) = O(m + m \sum p_x \lg(1/p_x))$; teda splay$(x)$ má amortizovanú zložitosť $O(1+\lg(1/p_x))$. Hodnota $H = p_x \lg(1/p_x)$ je entropia pravdepodobnostného rozdelenia. Z teórie informácie vyplýva, že ak poznáme $f_x$, najlepší statický strom dosahuje zložitosť zhruba $mH$ – splay strom dosahuje konštantný násobok a to *bez znalosti* $f_x$!

Dôkaz: zvoľme $w_x = f_x$, potom $r_{\text{koreň}} = \lg m$ a $r_x = \lg f_x$, dosadíme do vety o prístupe.

**Veta o statickom prste:** Zvoľme si prst – vrchol $p$; amortizovaná zložitosť splay$(x)$ je $O(\lg(2 + |x - p|))$, kde $|x - p|$ je vzdialenosť (počet prvkov) medzi $x$ a $p$. Inými slovami, ak často pristupujeme k prvkom blízko $p$, prístup je rýchly.

Dôkaz: zvoľme $w_x = 1/(|x - p| + 1)^2$; $s_{\text{koreň}} < 2 \sum_{k=1}^{\infty} 1/k^2 = \pi^2/6 = O(1)$, dosadíme do vety o prístupe.

**Veta o pracovnej množine:** Nech $t_i(x)$ je počet rôznych prvkov (vrátane $x$), ktoré sme splayovali odkedy sme naposledy vysplayovali $x$ pred časom $i$; potom splay$(x)$ trvá $O(1 + \lg t_i(x_i))$ amortizovane. Inými slovami, ak stále pristupujeme iba k malej „pracovnej“ množine prvkov, čas je logaritmický od veľkosti pracovnej množiny.

Náčrt dôkazu: Váhy budeme meniť; v čase $i$ zvoľme $w_x = 1/t_i(x)^2$. Potom $s_{\text{koreň}} = \sum_{k=1}^{\infty} 1/k^2 = \pi^2/6$, čiže splay$(x_i)$ trvá $O(1 + \lg(O(1)/t_i(x_i)^{-2})) = O(1 + \lg t_i(x_i))$. Treba ešte overiť, že sme s meniacimi sa váhami nepodvádzali; ako sa zmenia váhy? Všetkým prvkom, ktoré sme splayovali od posledného splay$(x_i)$ sa $t_i(y)$ zvýši o 1; vrcholom, ktorých sme sa odvtedy nedotkli sa váha nezmení a prvok $x_i$ bude mať váhu 1. Inými slovami, ak je $t_i(x_i) = k$, potom $t_{i+1}(y)$ sa zmení takto: $k \to 1$, $1 \to 2$, $2 \to 3$,$\ldots$ $k - 1 \to k$. Teda $w_{x_i}$ vzrastie o $< 1$ a ostatné váhy klesnú alebo ostanú nezmenené, teda $\Delta\Phi < 1$.

### Ďalšie vlastnosti splay stromov

**Veta o skenovaní:** Ak pristupujeme postupne k prvkom $1, 2, 3, \ldots, n$, celkový čas je $O(n)$.

**Veta o dynamickom prste:** prístup ku $x_i$ trvá $O(\lg(2 + |x_i - x_{i-1}|))$, teda prístup blízko predošlému prvku je rýchly. Z tejto vety vyplýva veta o skenovaní aj veta o statickom prste. Dôkaz je veľmi ťažký.

**Hypotéza o obojsmernej fronte:** Ak splay strom používame ako deque, teda vkladáme a vyberáme prvky iba zo začiatku alebo konca, čas bude $O(m)$ (amortizovane). Zatiaľ najlepší dokázaný odhad je $O(m\alpha(m))$.

**Hypotéza o split strome:** Split strom je dátová štruktúra, ktorá podporuje operácie make$(x_1, \ldots, x_n)$ – vytvorenie stromu a split$(x)$ – vráti $x$ a rozdelí strom na 2 split stromy s prvkami $< x$ a $> x$. Existuje algoritmus, kde make a $n\times$split trvá $O(n)$; predpokladá sa, že splay strom dosahuje rovnakú zložitosť. Zatiaľ najlepší dokázaný odhad je $O(n\alpha(n))$.

**Zjednotená hypotéza:** Zovšeobecňuje vlastnosť pracovnej množiny a dynamického prsta: ak sme nedávno pristupovali k prvku, ktorý je blízko, prístup bude rýchly: $O(\lg \min_y[t_i(y) + |x_i - y| + 2])$ amortizovane.

**Hypotéza o dynamickej optimálnosti:** Splay strom je len konštantný násobok od najlepšieho možného BST algoritmu, ktorý pozná celú postupnosť prístupov dopredu.