Splay stromy

©kuko 8.3.2018

1 Splayovanie

- splay strom je binárny vyhľadávací strom
- operácia splay(x) nájde vrchol x a vybuble ho nahor do koreňa
- bublanie prebieha špeciálnym spôsobom:
  - nech y je otec a z starý otec vrcholu x
  - a) prípad „cik“: ak x nemá starého otca (y je koreň), zrotujeme x
  - b) prípad „cik-cik“: ak x aj y sú obaja ľaví synovia, alebo obaja praví synovia, zrotujeme najskôr y, potom x
  - c) prípad „cik-cak“: ak x je ľavý syn a y pravý syn (alebo naopak), dvakrát zrotujeme x

2 Jednoduchá analýza

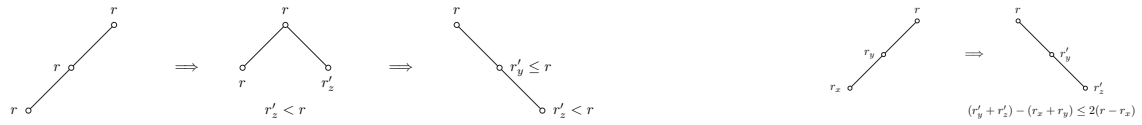
- budeme analyzovať iba operáciu splay
- pre jednoduchosť budeme predpokladať, že kľúče sú $1, \ldots, n$, splayujeme kľúče, ktoré sú v strome
- nech size s_x je počet vrcholov v podstrome x
- nech rank $r_x = \lfloor \log s_x \rfloor$
- **Invariant:** každý vrchol bude mať na účte našetrených r_x\$

- pár postrehov:
  - rotácia vrcholu x s otcom y mení iba r_x a r_y – ostatné ranky sa nemenia
  - rank x po rotácii je rovnaký ako rank otca pred rotáciou: $r'_x = r_y$
  - listy majú $s_x = 1$, teda $r_x = 0$; rank otca je vždy $\geq$ rank syna
  - maximálny rank má koreň; $r_{\text{root}} = \lfloor \log s_{\text{root}} \rfloor = \lfloor \log n \rfloor$
  - aký je súčet všetkých rankov, tzn. koľko majú našetrené všetky vrcholy dokopy?
  - najmenej, ak je strom perfektne vyvážený, je to $\sum_{k=1}^{\lfloor \log n \rfloor} (n/2^k) \cdot k = \Theta(n)$
  - najviac, ak je strom maximálne nevyvážený s hĺbkou n, je to $\sum_{k=1}^{n} \lfloor \log k \rfloor \geq \sum_{k=n/2}^{n} \lfloor \log k \rfloor \geq (n/2) \lfloor \log(n/2) \rfloor = \Theta(n \log n)$
  - ak má vrchol rank r, má pod sebou $\geq 2^r$ vrcholov
  - ak majú dvaja bratia rank r, ich otec musí mať rank $\geq r + 1$ (lebo je pod ním $\geq 2^{r+1}$ vrcholov)
  - ak má otec a syn rank r, potom druhý syn musí mať rank $< r$ (inak dostaneme spor s predchádzajúcim tvrdením)

- intuícia 1:
  - uvažujme cestu od vrcholu x ku koreňu
  - keďže $r_x \in \{0, \ldots, \lfloor \log n \rfloor\}$, rank sa na tejto ceste môže zvýšiť len $\log n$-krát
  - ak sa v jednom kroku splayovania rank zvýši, zaplatíme ho z peňazí na splayovanie
  - ak je rank rovnaký, prístup je pomalší, ale ukážeme, že vrcholy majú našetrené dosť na to, aby tento krok zaplatili – z hladiska amortizovanej analýzy je teda tento krok zadarmo
- intuícia 2:

  - nech y, z sú otec a starý otec vrcholu x; nech α, β sú podstromy pod x a δ, γ podstromy pod y, z
  - ak $r_z > r_x$, tak δ, γ sú zhruba aspoň také veľké ako α, β, takže sme pri hľadaní vylúčili podstatnú časť kľúčov
  - (v skutočnosti je toto tvrdenie nepresné a nemusí platiť kvôli $\lfloor \cdot \rfloor$; ale platí, že ak sa rank na ceste zvýši aspoň o 2, vylúčili sme aspoň konštantný zlomok kľúčov)
  - naopak, ak $r_z = r_x$, znamená to, že podstromy δ, γ sú malé v porovnaní s α, β, t.j. väčšina vrcholov je pod x
  - avšak v tomto prípade sa vyváženosť stromu zlepší, pretože celé α, β budú po rotáciach vyššie; teda väčšina prvkov pod x bude mať menšiu hĺbku

- **Veta o prístupe:** Na operáciu splay(x) stačí $3(r_{\text{root}} - r_x) + 1$\$, pričom zachovávame invariant
- označme r'_x rank vrcholu x po jednom kroku splayovania (po jednej dvojitej rotácií, resp. po poslednej jednoduchej rotácií)
- **Lema:** Na každý cik-cik/cik-cak prípad stačí $3(r'_x - r_x)$\$, na posledný prípad cik $(r'_x - r_x) + 1$\$.
- z lemy priamo vyplýva veta o prístupe, pretože keď sčítame všetky kroky splay(x), dostávame teleskopickú sumu $3(r_x^{\text{posledný}} - r_x^{\text{predposledný}} + r_x^{\text{predposledný}} - \cdots + r'''_x - r''_x + r''_x - r'_x + r'_x - r_x) + 1 = 3(r_x^{\text{posledný}} - r_x) + 1$, pričom $r_x^{\text{posledný}} = r_{\text{root}}$ je rank celého stromu
- analyzujme jednotlivé prípady; nech x je prvok, ktorý splayujeme, y a z sú otec a starý otec; r_x, r_y, r_z sú ich ranky pred rotáciami a r'_x, r'_y, r'_z sú ranky po rotáciach (zvyšné ranky sa nezmenia)
- prípad „cik“:
  - pred rotáciou máme našetrených: $r_x + r_y$ (+zvyšné ranky)\$
  - po rotácii potrebujeme mať: $r'_x + r'_y$ (+zvyšné ranky, ale tie sa nezmenili)\$
  - takže na udržanie invariantu potrebujeme $(r'_x + r'_y) - (r_x + r_y)$ mincí + aspoň 1 mincu na zaplatenie rotácií
  - platí: $r'_x = r_y$ (rank po = rank otca pred) a $r'_x \leq r'_y$ (rank syna nie je väčší ako rank otca)
  - teda potrebných $(r'_x + r'_y) - (r_x + r_y) + 1 = r'_y - r_x + 1 \leq (r'_x - r_x) + 1$
- prípad „cik-cik“; analyzujme zvlášť prípad $r_x = r_z$ a $r_x < r_z$:
- a) ak $r_x = r_z$, tak $r_x = r_y = r_z = r'_x = r$
  - čiže na tento krok máme pridelených $3(r'_x - r_x) = 0$ – tento krok by mal byť v zmysle amortizovanej zložitosti zadarmo (všetko platíme z už našetrených peňazí)
  - pozri obr.: po prvej rotácií má x aj y rank r (x sa nezmenil a y je koreň nášho podstromu, teda má rank $r_z = r$)
  - po druhej rotácii sa rank z nezmení a $r'_x = r_z = r$, teda $r'_z < r'_x$,
  - z toho ale vyplýva, že pred rotáciami sme mali $3r$\$, po rotáciach nám stačí $\leq 3r - 1$
  - čiže prechod cez x, y, z a cik-cik rotácie zaplatí vrchol z a invariant ostane zachovaný
- b) ak $r_x < r_z$,
  - pred rot.: $r_x + r_y + r_z$\$, po rot.: $r'_x + r'_y + r'_z$\$, teda potrebujeme $(r'_x + r'_y + r'_z) - (r_x + r_y + r_z) + 1$\$
  - platí: $r'_x = r_z$ (koreň pred a po), $r'_x \geq r'_y$, $r'_x \geq r'_z$ a $r_x \leq r_y$ (syn≤otec)
  - odtiaľ $(r'_x + r'_y + r'_z) - (r_x + r_y + r_z) + 1 = (r'_y - r_x) + (r'_z - r_y) + 1 \leq (r'_x - r_x) + (r'_x - r_x) + 1 = 2(r'_x - r_x) + 1$
  - keďže $r_x < r_z = r'_x \Rightarrow r'_x - r_x \geq 1$, čiže $2(r'_x - r_x) + 1 \leq 3(r'_x - r_x)$
  - na tento krok máme pridelených $3(r'_x - r_x)$\$; $(r'_x - r_x) \geq 1$\$ použijeme na zaplatenie operácií; $2(r'_x - r_x)$ na dorovnanie nasporených peňazí, aby sme zachovali invariant

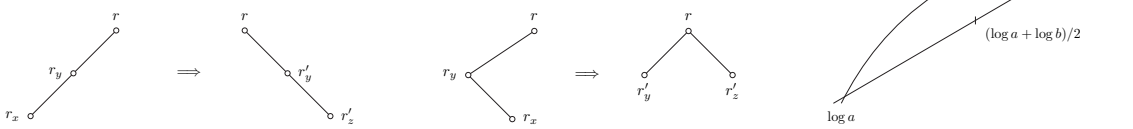


- prípad „cik-cak“: D.Ú.

3 Všeobecná analýza

- každému vrcholu x priradíme váhu $w_x \in \mathbb{R}^+$
- nech size s_x je súčet váh všetkých vrcholov v podstrome x
- nech rank $r_x = \log s_x$
- **Invariant:** každý vrchol bude mať na účte našetrených r_x\$; inými slovami potenciál $\Phi = \sum r_x$

- **Veta o prístupe:** Operácia splay(x) má amortizovanú zložitosť $3(r_{\text{root}} - r_x) + 1$ (pre ľubovoľnú voľbu váh)
- inými slovami, ak $W = \sum w_x$ je celková váha stromu, potom zložitosť splay(x) je $3(\log W - \log w_x) + 1 = O(1 + \log(W/w_x))$
- **Lema:** Každý cik-cik/cik-cak prípad má amortizovanú zložitosť $3(r'_x - r_x)$, posledný prípad cik má zložitosť $3(r'_x - r_x) + 1$
- amortizovaná zložitosť a je skutočná zložitosť t (2 rotácie) + rozdiel potenciálov $\Delta\Phi$
- prípad „cik-cik“:
  - $a = t + \Delta\Phi = 2 + (r'_x + r'_y + r'_z) - (r_x + r_y + r_z) = 2 + r'_y + r'_z - r_x - r_y \le 2 + r'_x + r'_z - 2r_x$
  - $r_x + r'_z = \log s_x + \log s'_z \le 2\log[(s_x + s'_z)/2] \le 2\log(s'_x/2) = 2r'_x - 2$
  - teda $r'_z \le 2r'_x - r_x - 2$, dosadíme vyššie:
  - $a \le 2 + r'_x + (2r'_x - r_x - 2) - 2r_x = 3(r'_x - r_x)$
- prípad „cik-cak“:
  - $a = 2 + r'_y + r'_z - r_x - r_y \le 2 + r'_y + r'_z - 2r_x$
  - $r'_y + r'_z = \log s'_y + \log s'_z \le 2r'_x - 2$
  - $a \le 2 + 2r'_x - 2 - 2r_x = 2(r'_x - r_x)$



- prípad „cik“: prenechávame čitateľovi

4 Dôsledky

- **Vyváženosť:** pre $w_x = 1$ dostávame $r_{\text{root}} = \log n$, takže amortizovaná zložitosť je $O(\log n)$
- **Statická optimálnosť,** alias Veta o entropii: nech $f_x \ge 1$ je frekvencia, s ktorou splayujeme x, m je celkový počet operácií, $p_x = f_x/m$; potom zložitosť m operácií je $O(m + \sum f_x \log(m/f_x)) = O(m + m\sum p_x \log(1/p_x))$; teda splay($x$) má amortizovanú zložitosť $O(1 + \log(1/p_x))$
  - hodnota $H = p_x \log(1/p_x)$ je entropia pravdepodobnostného rozdelenia
  - z teorie informácie vyplýva, že ak poznáme f_x, že najlepší statický strom dosahuje zložitosť zhruba mH – splay strom dosahuje konštantný násobok bez znalosti f_x
  - dôkaz: zvoľme $w_x = f_x$, potom $r_{\text{root}} = \log m$ a $r_x = \log f_x$, dosadíme do vety o prístupe
  - z tejto vety vyplýva veta o statickom prste
- **Veta o statickom prste:** zvoľme si prst – vrchol p; amortizovaná zložitosť splay(x) je $O(\log(2 + |x - p|))$, 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 $w_x = 1/(x - p + 1)^2$; $s_{\text{root}} < 2\sum_{k=1}^{\infty} 1/k^2 = \pi^2/6 = O(1)$, 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 $O(1 + \log t_i(x_i))$ amortizovane

- inými slovami, ak stále pristupujeme iba k malej „pracovnej“ množine prvkov, čas je logaritmický od veľkosti pracovnej množiny
- dôkaz: váhy budeme meniť; v čase i zvoľme $w_x = 1/t_i(x)^2$
- potom $s_{\text{root}} = \sum_{k=1}^{\infty} 1/k^2 = \pi^2/6$, čiže splay$(x_i)$ trvá $O(1 + \log(O(1)/t_i(x_i)^{-2})) = O(1 + \log t_i(x_i))$
- 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 $\Delta\Phi < 1$
- z tejto vety vyplýva statická optimálnosť

5 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 $O(\log(2 + |x_i - x_{i-1}|))$
  - 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)
  - 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: $O(\log \min_y[t_i(y) + |x_i - y| + 2])$ amortizovane
- Hypotéza o dynamickej optimálnosti: splay strom je len konštantný násobok od najlepšieho možného BST algoritmu, ktorý pozná celú postupnosť prístupov dopredu