Datové struktury I

2. přednáška: BB[α]-stromy, Splay stromy

Jirka Fink

https://ktiml.mff.cuni.cz/~fink/

Katedra teoretické informatiky a matematické logiky
Matematicko-fyzikální fakulta
Univerzita Karlova v Praze

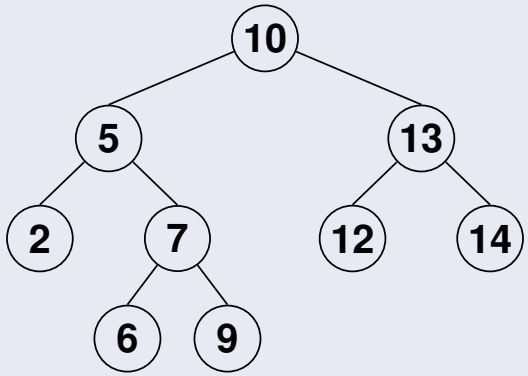
Zimní semestr 2024/25

Binární vyhledávací strom (BVS)

Definice

- Binární strom (každý vrchol obsahuje nejvýše dva syny)
- Klíč v každém vnitřním vrcholu je větší než všechny klíče v levém podstromu a menší než všechny klíče v pravém podstromu
- Prvky mohou být uloženy pouze v listech nebo též ve vnitřních vrcholech (u každého klíče je uložena i hodnota)

Příklad



Složitost

- Paměť: $\mathcal{O}(n)$
- Časová složitost operace Find je lineární ve výšce stromu
- Výška stromu může být až $n - 1$

Váhově vyvážené stromy: BB[α]-strom

BB[α]-strom (Nievergelt, Reingold, 1973)

- Binární vyhledávací strom ①
- Počet vrcholů v podstromu vrcholu u označme s_u ②
- Pro každý vrchol u platí, že podstromy obou synů u musí mít nejvýše αs_u vrcholů ③
- Zřejmě musí platit $\frac{1}{2} < \alpha < 1$ ④

Výška BB[α]-stromu

- Podstromy všech vnuků kořene mají nejvýše $\alpha^2 n$ vrcholů
- Podstromy všech vrcholů v i-té vrstvě mají nejvýše $\alpha^i n$ vrcholů
- $\alpha^i n \geq 1$ jen pro $i \leq \log_{\frac{1}{\alpha}}(n)$
- Výška BB[α]-stromu je $\Theta(\log n)$

Operace BUILD: Vytvoření BB[α]-stromu ze setříděného pole

- Prostřední prvek dáme do kořene
- Rekurzivně vytvoříme oba podstromy
- Časová složitost je $\mathcal{O}(n)$

1. V přednášce budeme předpokládat, že prvky jsou uloženy ve všech vrcholech, i když existuje varianta BB[α]-stromů mající prvky jen v listech.
2. Do s_u započítáváme i vrchol u.
3. V literatuře můžeme najít různé varianty této podmínky. Podstatné je, aby oba podstromy každého vrcholu měli „zhruba“ stejný počet vrcholů.
4. Pro $\alpha = \frac{1}{2}$ lze BB[α]-strom sestrojit, ale operace INSERT a DELETE by byly časově náročné. Pro $\alpha = 1$ by výška BB[α]-strom mohla být lineární.

BB[α]-strom: Operace INSERT a DELETE

Operace INSERT (DELETE je analogický)

- Najít list pro nový prvek a uložit do něho nový prvek (složitost: $\mathcal{O}(\log n)$)
- Jestliže některý vrchol porušuje vyvažovací podmínku, tak celý jeho podstrom znovu vytvoříme operací BUILD (složitost: amortizovaná analýza) ① ②

Amortizovaná složitost operací INSERT a DELETE: Agregovaná metoda

- Jestliže podstrom vrcholu u po provedení operace BUILD má s_u vrcholů, pak další porušení vyvažovací podmínky pro vrchol u nastane nejdříve po $\Omega(s_u)$ přidání/smazání prvků v podstromu vrcholu u (cvičení)
- Rebuild podstromu vrcholu u trvá $\mathcal{O}(s_u)$
- Amortizovaný čas vyvažovaní jednoho vrcholu je $\mathcal{O}(1)$ ③
- Při jedné operaci INSERT/DELETE se prvek přidá/smaže v $\Theta(\log n)$ podstromech
- Amortizovaný čas vyvažovaní při jedné operaci INSERT nebo DELETE je $\mathcal{O}(\log n)$
- Jaký je celkový čas k operací? ④

1. Při hledání listu pro nový vrchol stačí na cestě od kořene k listu kontrolovat, zda se přidáním vrcholu do podstromu syna neporuší vyvažovací podmínka. Pokud se v nějakém vrcholu podmínka poruší, tak se hledání ukončí a celý podstrom včetně nového prvku znovu vybuduje.
2. Existují pravidla pro rotování BB[α]-stromů, ale ta se nám dnes nehodí.
3. Operace BUILD podstromu vrcholu u trvá $\mathcal{O}(s_u)$ a mezi dvěma operacemi BUILD podstromu u je $\Omega(s_u)$ operací INSERT nebo DELETE do podstromu u. Všimněte si analogie a dynamickým polem.
4. Intuitivně bychom mohli říct, že v nejhorším případě BB[α]-strom nejprve vyvážíme v čase $\mathcal{O}(n)$ a poté provádíme jednotlivé operace, a proto celkový čas je $\mathcal{O}(n + k \log n)$, ale není to pravda. Proč?

Amortizovaná složitost operací INSERT a DELETE: Potenciální metoda

- V této analýze uvažujeme jen čas na postavení podstromu, zbytek trvá $\mathcal{O}(\log n)$
- Potenciál vrcholu u definován

$$\Phi(u) = \begin{cases} 0 & \text{pokud } |s_{l(u)} - s_{r(u)}| \leq 1 \\ |s_{l(u)} - s_{r(u)}| & \text{jinak,} \end{cases}$$

  kde $l(u)$ a $r(u)$ jsou levý a pravý synové u.
- Potenciál BB[α]-stromu Φ je součet potenciálů vrcholů
- Při vložení/smazání prvku se potenciál $\Phi(u)$ jednoho vrcholu zvýší nejvýše o 2 ①
- Pokud nenastane Rebuild, pak se potenciál stromu zvýší nejvýše o $\mathcal{O}(\log(n))$ ②
- Pokud nastane Rebuild vrcholu u, pak $\Phi(u) \geq \alpha s_u - (1-\alpha)s_u \geq (2\alpha - 1)s_u$
- Po rekonstrukci mají všechny vrcholy v podstromu u nulový potenciál ③
- Při rekonstrukci poklesne potenciál Φ alespoň o $\Omega(s_u)$, což zaplatí čas na rekonstrukci
- Dále platí $0 \leq \Phi \leq hn = \mathcal{O}(n \log n)$, kde h je výška stromu ④
- Celkový čas na k operací INSERT nebo DELETE je $\mathcal{O}((k+n)\log n)$

1. Potenciál se změní právě o 2, jestli rozdíl velikostí podstromů se změní z 1 na 2 nebo opačně. Jinak se potenciál změní právě o 1.
2. Potenciál se může změnit pouze vrcholům na cestě z kořene do nového/smazaného vrcholu a těch je $\mathcal{O}(\log n)$.
3. Právě zde potřebujeme, aby potenciál vrcholu byl nulový, i když se velikosti podstromů jeho synů liší o jedna.
4. Součet potenciálů všech vrcholů v jedné libovolné vrstvě je nejvýše n, protože každý vrchol patří do nejvýše jednoho podstromu vrcholu z dané vrstvy. Tudíž potenciál stromu Φ je vždy nejvýše nh. Též lze nahlédnout, že každý vrchol je započítán v nejvýše h potenciálech vrcholů.

Staticky optimální strom

Cíl

Pro danou posloupnost operací FIND najít binární vyhledávací strom minimalizující celkovou dobu vyhledávání.

Formálně

Máme prvky $x_1, \ldots, x_n$ s váhami $w_1, \ldots, w_n$. Cena stromu je $\sum_{i=1}^{n} w_i h_i$, kde h_i je hloubka prvku x_i. Staticky optimální strom je binární vyhledávací strom s minimální cenou.

Konstrukce (cvičení)

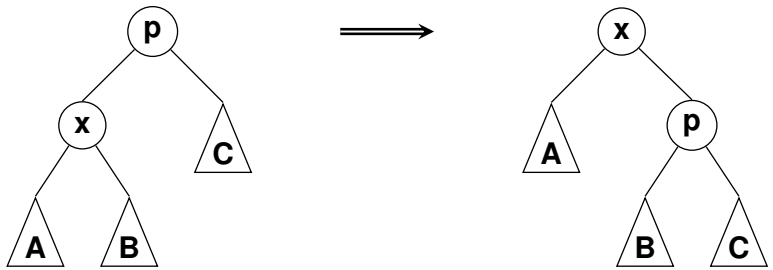
- $\mathcal{O}(n^3)$ – triviálně dynamickým programováním
- $\mathcal{O}(n^2)$ – vylepšené dynamické programování (Knuth, 1971)

Jak postupovat, když neznáme váhy předem?

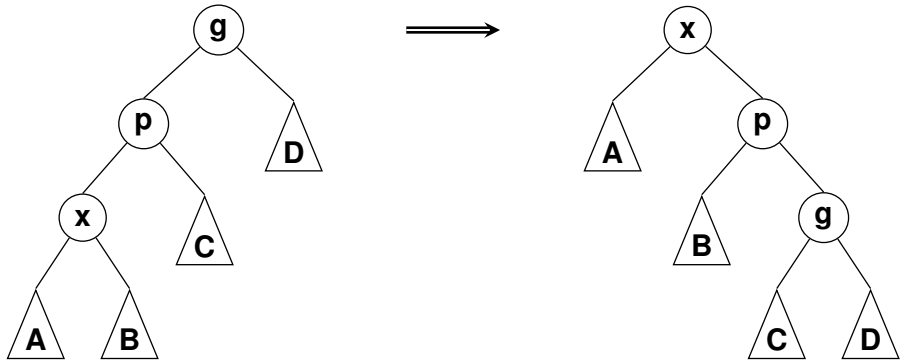
- Pomocí rotací bude udržovat často vyhledávané prvky blízko kořene
- Operací SPLAY „rotujeme“ zadaný prvek až do kořene
- Operace FIND vždy volá SPLAY na hledaný prvek

Splay strom (Sleator, Tarjan, 1985): Operace SPLAY prvku *x*

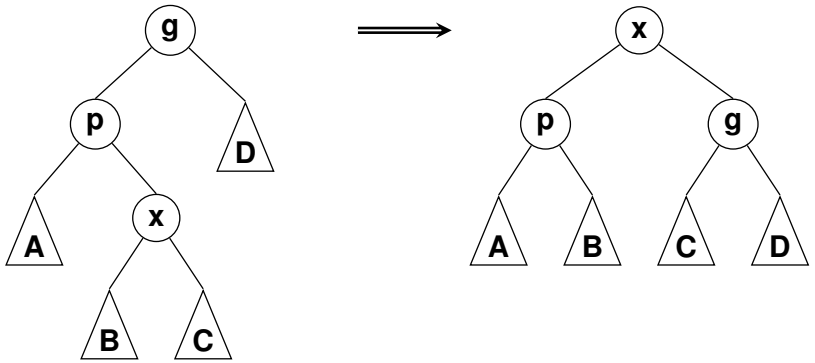
- Zig rotace: Otec *p* prvku *x* je kořen



- Zig-zig rotace: *x* a *p* jsou oba pravými nebo oba levými syny



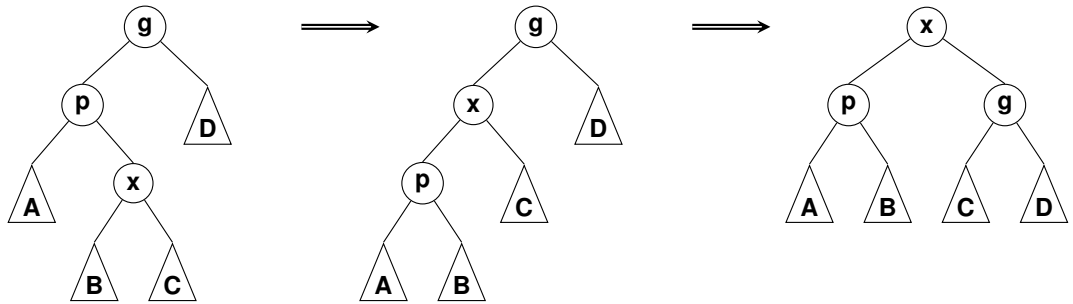
- Zig-zag rotace: *x* je pravý syn a *p* je levý syn nebo opačně



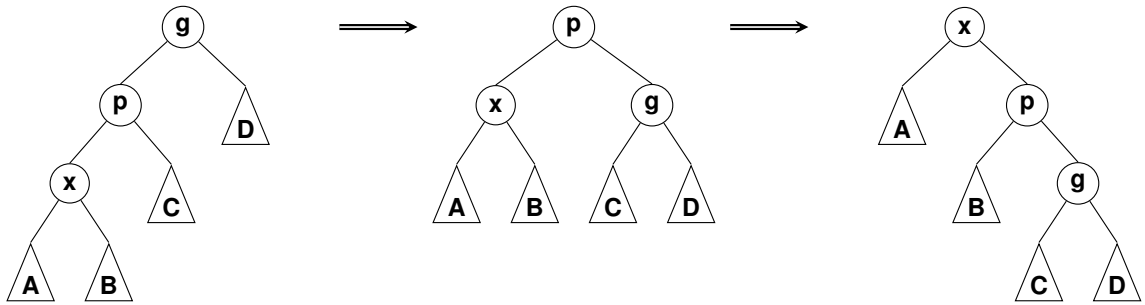
Uvažujeme *buttom-up* verzi, tj. prvek x nejprve najdeme a poté jej postupně rotujeme nahoru, což znamená, že x vždy značí stejný vrchol postupně se přesouvající ke kořeni a ostatní vrcholy stromu jsou sousedé odpovídající dané rotaci.
Existuje též *top-down* verze, která vždy rotuje vnuka kořene, jehož podstrom obsahuje prvek x. Tato verze je sice v praxi rychlejší, ale postup a analýza jsou složitější.

Splay strom: Operace SPLAY prvku *x*

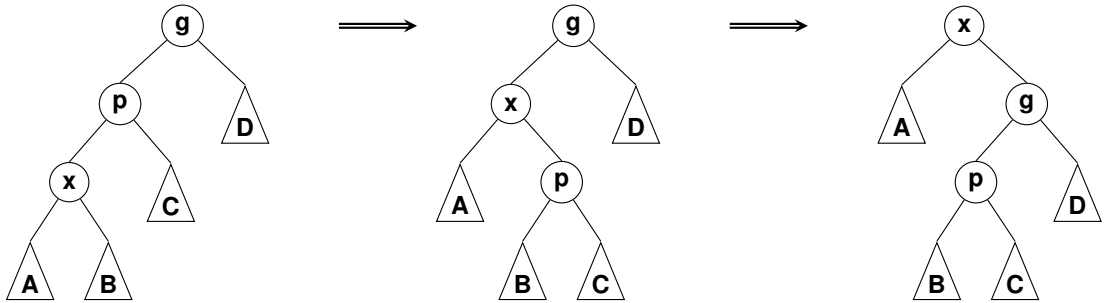
- Zig-zag rotace jsou pouze dvě jednoduché rotace prvku *x* s aktuálním otcem



- Zig-zig rotace jsou taky dvě rotace,



- ale dvě rotace prvku *x* s aktuálním otcem by vedli ke špatnému výsledku



Splay strom: Amortizovaná analýza

Lemma

Pro $a, b, c \in \mathbb{R}^+$ splňující $a + b \leq c$ platí $\log_2(a) + \log_2(b) \leq 2\log_2(c) - 2$.

Důkaz

- Platí $4ab = (a+b)^2 - (a-b)^2$
- Z nerovností $(a-b)^2 \geq 0$ a $a+b \leq c$ plyne $4ab \leq c^2$
- Zlogaritmováním dostáváme $\log_2(4) + \log_2(a) + \log_2(b) \leq \log_2(c^2)$

Značení

- Nechť velikost $s(x)$ je počet vrcholů v podstromu x (včetně x)
- Potenciál vrcholu x je $\Phi(x) = \log_2(s(x))$
- Potenciál Φ stromu je součet potenciálů všech vrcholů
- s' a Φ' jsou velikosti a potenciály po jedné rotaci
- Předkládáme, že jednoduchou rotaci zvládneme v jednotkovém čase

Lemma můžeme též dokázat pomocí Jensenovy nerovnosti, která tvrdí:
Jestliže f je konvexní funkce, $x_1, \ldots, x_n$ jsou čísla z definičního oboru f a $w_1, \ldots, w_n$ jsou kladné váhy, pak platí nerovnost

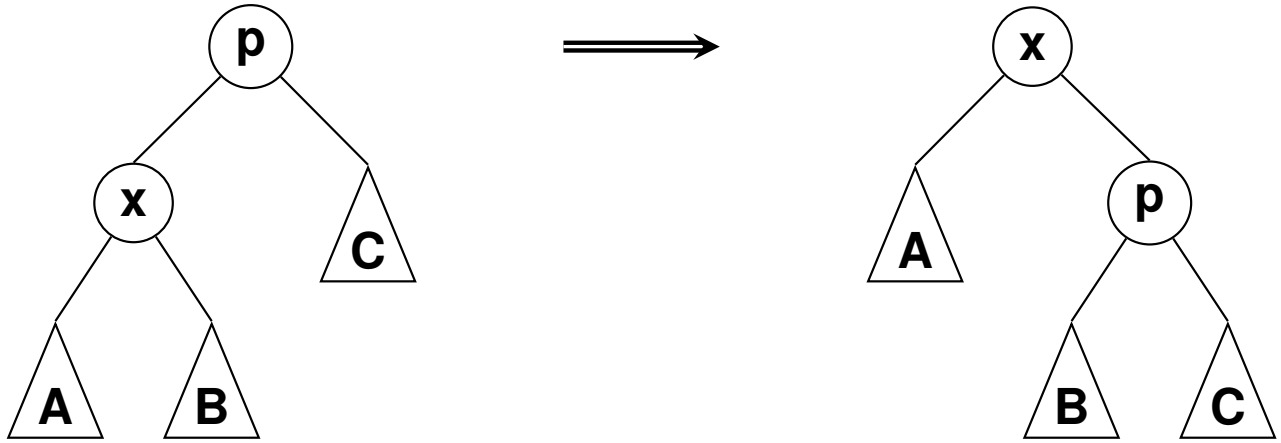
$$f\left(\frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}\right) \leq \frac{\sum_{i=1}^{n} w_i f(x_i)}{\sum_{i=1}^{n} w_i}.$$

Jelikož funkce log je rostoucí a konkávní, dostáváme

$$\frac{\log(a) + \log(b)}{2} \leq \log\left(\frac{a+b}{2}\right) \leq \log(c) - 1,$$

z čehož plyne znění lemmatu.

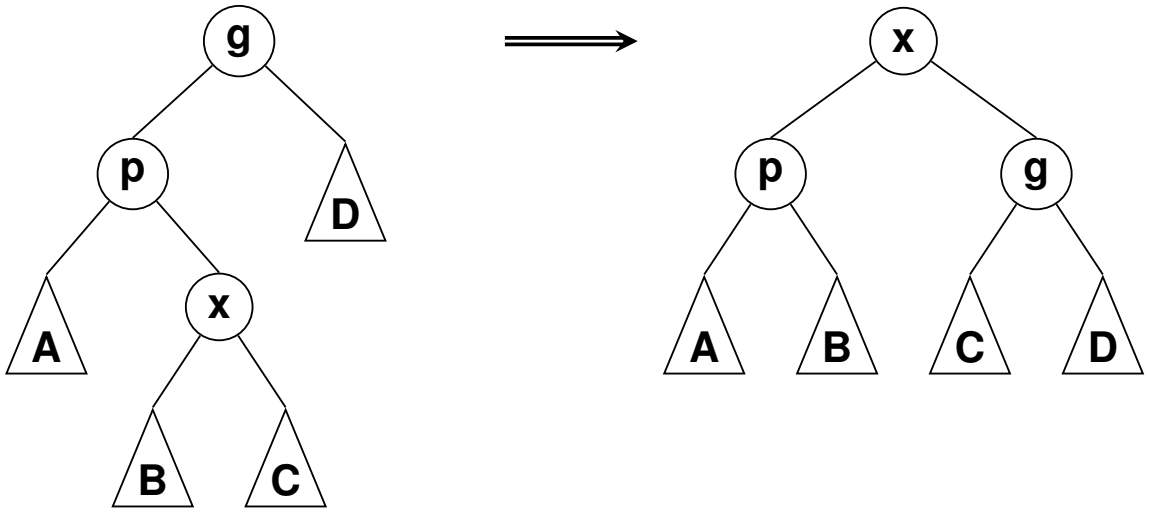
Splay strom: Zig rotace



Analýza

- $\Phi'(x) = \Phi(p)$
- $\Phi'(p) < \Phi'(x)$
- $\Phi'(u) = \Phi(u)$ pro všechny ostatní vrcholy u
- $$\begin{aligned}\Phi' - \Phi &= \sum_u (\Phi'(u) - \Phi(u)) \\ &= \Phi'(p) - \Phi(p) + \Phi'(x) - \Phi(x) \\ &\leq \Phi'(x) - \Phi(x)\end{aligned}$$

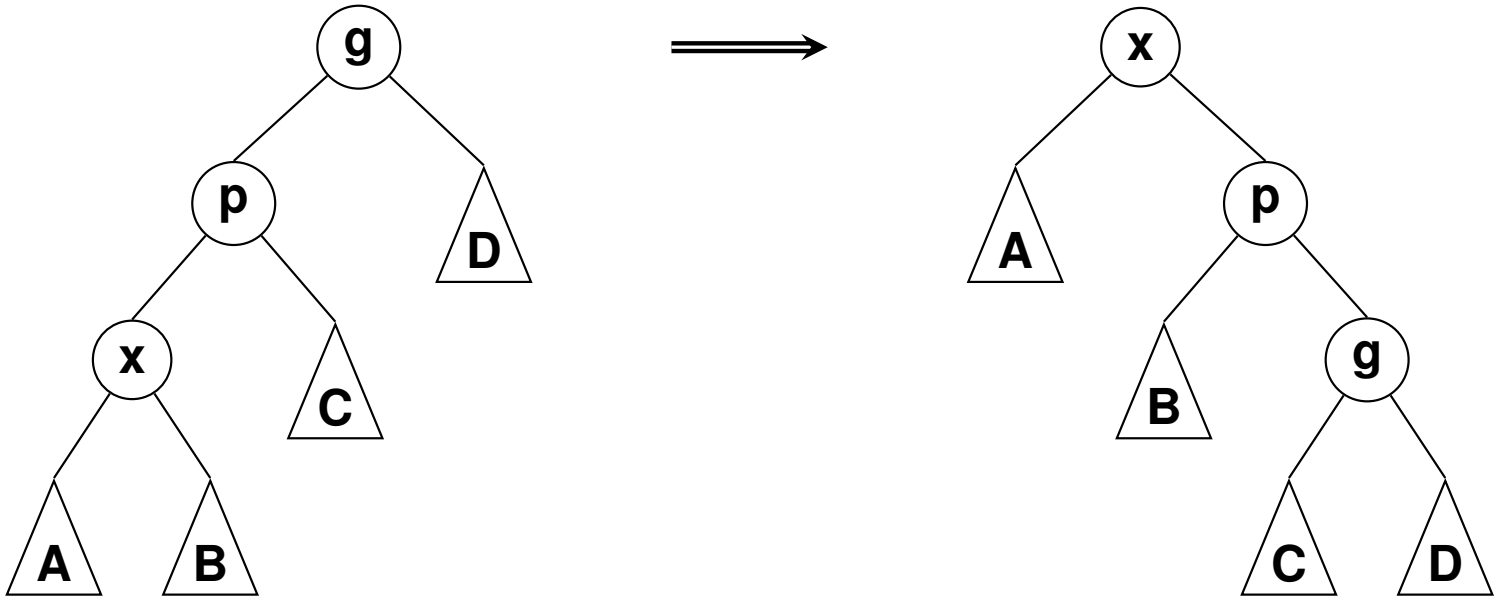
Splay strom: Zig-zag rotace



Analýza

1. $\Phi'(x) = \Phi(g)$
2. $\Phi(x) < \Phi(p)$
3. $\Phi'(p) + \Phi'(g) \leq 2\Phi'(x) - 2$
   - $s'(p) + s'(g) \leq s'(x)$
   - Z lemmatu plyne $\log_2(s'(p)) + \log_2(s'(g)) \leq 2\log_2(s'(x)) - 2$
4. $\Phi' - \Phi = \Phi'(g) - \Phi(g) + \Phi'(p) - \Phi(p) + \Phi'(x) - \Phi(x) \leq 2(\Phi'(x) - \Phi(x)) - 2$

Splay strom: Zig-zig rotace



Analýza

- $\Phi'(x) = \Phi(g)$
- $\Phi(x) < \Phi(p)$
- $\Phi'(p) < \Phi'(x)$
- $s(x) + s'(g) \leq s'(x)$
- $\Phi(x) + \Phi'(g) \leq 2\Phi'(x) - 2$
- $\Phi' - \Phi = \Phi'(g) - \Phi(g) + \Phi'(p) - \Phi(p) + \Phi'(x) - \Phi(x) \leq 3(\Phi'(x) - \Phi(x)) - 2$

Splay strom: Analýza

Amortizovaný čas ①

- Amortizovaný čas jedné zigzig nebo zigzag rotace:
  $T + \Phi' - \Phi \leq 2 + 3(\Phi'(x) - \Phi(x)) - 2 = 3(\Phi'(x) - \Phi(x))$ ②
- Amortizovaný čas jedné zig rotace:
  $T + \Phi' - \Phi \leq 1 + \Phi'(x) - \Phi(x) \leq 1 + 3(\Phi'(x) - \Phi(x))$
- Nechť Φ_i je potenciál po i-tém kroku a T_i je skutečný čas i-tého kroku ③
- Amortizovaný čas (počet jednoduchých rotací) jedné operace SPLAY:

$$\sum_{i\text{-tá rotace}} (T_i + \Phi_i - \Phi_{i-1}) \leq 1 + \sum_{i\text{-tá rotace}} 3(\Phi_i(x) - \Phi_{i-1}(x))$$

$$\leq 1 + 3(\Phi_{\text{konec}}(x) - \Phi_0(x)) \quad ④$$

$$\leq 1 + 3\log_2 n = \mathcal{O}(\log n)$$

- Amortizovaný čas jedné operace SPLAY je $\mathcal{O}(\log n)$

Skutečný čas k operací SPLAY

- Potenciál vždy splňuje $0 \leq \Phi \leq n \log_2 n$
- Rozdíl mezi konečným a počátečním potenciálem je nejvýše $n \log_2 n$
- Celkový čas k operací SPLAY je $\mathcal{O}((n + k) \log n)$

1. Časy na nalezení prvku a jeho SPLAY jsou stejné, a proto analyzujeme počet rotací při operaci SPLAY. Neúspěšná operace FIND dojde až k vrcholu mající NULL v ukazateli, na kterém se vyhledávání zastaví. Na tento poslední vrchol je nutné zavolat SPLAY, aby opakovaná neúspěšná vyhledávání měla amortizovanou logaritmickou složitost.
2. T značí skutečný čas rotace, což je počet jednoduchých rotací k provedení rotace zig, zigzig nebo zigzag.
3. Jedním krokem rozumíme provedení jedné zig, zigzag nebo zigzig rotace.
4. Zig rotaci použijeme nejvýše jednou a proto započítáme „+1“. Rozdíly $\Phi'(x) - \Phi(x)$ se teleskopicky odečtou a zůstane nám rozdíl potenciálů vrcholu x na konci a na začátku operace SPLAY. Na počátku je potenciál vrcholu x nezáporný a na konci je x kořenem, a proto jeho potenciál je $\log_2(n)$.

Splay strom: Operace INSERT a DELETE

Postup

1. Vložit/smazat daný vrchol stejně jako v BVS
2. SPLAY nejhlubšího navštíveného vrcholu ①

Amortizovaná složitost

- Nalezení potřebných vrcholů má stejnou složitost jako SPLAY: $\mathcal{O}(\log n)$
- Vlastní smazání vrcholu trvá $\mathcal{O}(1)$ a potenciál stromu klesne
- Přidáním listu se potenciál vrcholů $u_1, \ldots, u_h$ na cestě do kořene zvýší o $\sum_{i=1}^{h} \log(s(u_i) + 1) - \log(s(u_i)) \leq \log(n)$ ②
- Amortizovaná složitost operací INSERT a DELETE je $\mathcal{O}(\log n)$

1. Pokud nezavoláme tento splay, tak celková složitost vkládání/mazání posloupnosti $1, \ldots, n$ je $\Theta(n^2)$.
2. Jestliže u_1 je nový list a u_h je kořen, pak $s(u_i) + 1 \leq s(u_{i+1})$ a tedy $\sum_{i=1}^{h} \log(s(u_i) + 1) - \log(s(u_i))$ lze teleskopicky shora odhadnout $\log(s(u_h)) - \log(s(u_1)) = \log n$.

Vlastnosti splay stromů

Věta

Nechť $x_1, \ldots, x_k$ je posloupnost vyhledávání prvků z množiny X a nechť z_i je počet různých prvků vyhledaných před předchozím vyhledání prvku x_i. Celková složitost vyhledání prvků $x_1, \ldots, x_k$ je $\mathcal{O}\left(n \log n + k + \sum_i \log(1 + z_i)\right)$.

Věta (vyhledávání podmnožiny prvků)

Jestliže provedeme k vyhledání prvků z podmnožiny velikosti m, pak celková složitost je $\mathcal{O}(n \log n + k + k \log m)$

Věta (vyhledávání prvků v rostoucím pořadí)

Jestliže posloupnost vyhledávání S obsahuje prvky v rostoucím pořadí, tak celkový čas na vyhledávání S ve splay stromu je $\mathcal{O}(n)$. ①

1. n je opět počet prvků ve stromu a počáteční splay strom může mít prvky rozmístěné libovolně.

Srovnání staticky optimálních stromů a splay stromů

Otázka

- Existuje posloupnost vyhledání, na které je splay strom asymptoticky rychlejší než staticky optimální strom zkonstruovaný pro tuto posloupnost?
- Existuje posloupnost vyhledání, na které je staticky optimální strom asymptoticky rychlejší než Splay strom?

Věta (statická optimalita)

Nechť $x_1, \ldots, x_k$ je posloupnost vyhledávání prvků z množiny X, kde každý prvek X je vyhledán aspoň jednou. Nechť T je statický strom na X a složitost vyhledání prvků v posloupnosti je f. Pak složitost vyhledání ve splay stromu je $\mathcal{O}(f)$.

Dynamická optimalita

- Jsou dvojité rotace ve splay stromu nejlepší možné?
- Existuje dynamický binární vyhledávací strom, který byl asymptoticky lepší než splay strom?
- Můžeme být lepší než splay strom, kdybychom posloupnost vyhledávání znali dopředu?

Splay stromy: Výhody a aplikace

Výhody a nevýhody Splay stromů

+ Nepotřebuje paměť na speciální příznaky ①
+ Efektivně využívají procesorové cache (Temporal locality)
- Rotace zpomalují vyhledávání
- Vyhledávání nelze jednoduše paralelizovat
- Výška stromu může být i lineární ②

Aplikace

- Cache, virtuální paměť, sítě, file system, komprese dat, ...
- Windows, gcc compiler and GNU C++ library, sed string editor, Fore Systems network routers, Unix malloc, Linux loadable kernel modules, ...

1. Červeno-černé stromy potřebují v každém vrcholu jeden bit na barvu, AVL stromy jeden bit na rozdíl výšek podstromů synů.
2. Když vyhledáme všechny prvky v rostoucím pořadí, pak strom zdegeneruje na cestu. Proto splay strom není vhodný v real-time systémech.

Příští týden

- (a,b)-stromy
- Souvislost (2,4)-stromů a červeno-černých stromů