

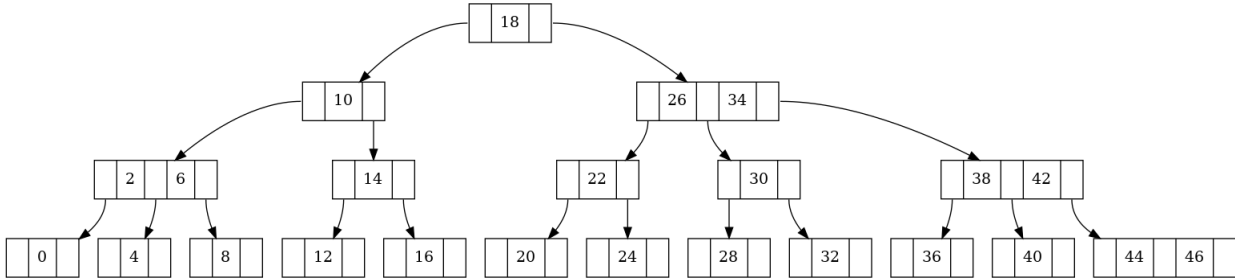
4. cvičení

Datové struktury I, 23. 10. 2025

<https://iuuk.mff.cuni.cz/~chmel/2526/ds1/>

Úloha 1 ((a, b)-stromy na vlastní kůži)

Na obrázku máte ($2, 3$)-strom. Proveďte na něm následující operace (vždy jenom jednu, a poté začněte s novým stromem): INSERT(7), INSERT(48), DELETE(44), DELETE(40), DELETE(32), DELETE(30), DELETE(16).



Řešení

Úloha 2 (Správná volba parametrů)

Z přednášky víme, že libovolná posloupnost m operací INSERT a DELETE na ($a, 2a$)-strom celkem změní jenom $\mathcal{O}(m)$ vrcholů (když začínáme s prázdným stromem). Ukažte, že toto neplatí pro ($a, 2a - 1$)-stromy, tedy pro libovolné m, n navrhnete posloupnost m operací na stromě s $\Theta(n)$ vrcholy, která celkem změní $\Omega(m \log n)$ vrcholů.

Můžete začít s ($2, 3$)-stromy, a potom zobecnit pro libovolné a . Zároveň můžete začít s libovolným (validním) n -vrcholovým stromem, a až na konci ukázat, že jej opravdu lze vyrobit z prázdného stromu.

Řešení

Obecně: strom který má všechny vrcholy skoro prázdné (a synů), jen pravá cesta má všechny vrcholy plné ($2a - 1$ synů). Označíme-li si M jako $1 +$ největší číslo ve stromě, pak operace INSERT(M), DELETE(M) musí vždy všechno nejdříve rozštěpit, a poté spojit zpátky.

Po insertu totiž nejnížší vrchol bude mít $2a$ synů, a tedy se bude muset rozštěpit na dva vrcholy s a syny - tím ale problém vybuchne o úroveň výše, a opět musíme štěpit, až se takto dostaneme do kořene. Když pak provedeme delete, vrchol, který M obsahoval bude mít najednou jenom $a - 1$ synů, a je „podměrečný“. Protože jeho jediný soused má také a synů, nemůžeme si žádného ukrást, a tedy musíme oba sloučit. Tím ovšem snížíme o 1 počet synů vrcholu o úroveň výše, a tohle nám opět probublá až do kořene.

Úloha 3 (Lepší zaplnění)

Zkuste upravit (a, b)-strom a jeho operace INSERT, DELETE tak, abychom mohli mít všechny vrcholy trochu plnější - konkrétně tak, abychom měli $(\frac{2}{3}b, b)$ -strom.

Řešení

Trik je v tom vhodně spojovat (a rozdělovat) vrcholy - normálně spojujeme dva vrcholy do jednoho, když jsou oba malé.

Insert upravíme následovně: prvek vložíme jako obvykle, a pokud máme přeplněný vrchol, pokusíme se jeden z prvků přesunout do sourozence. Pokud je sourozenec také plný, tak potom tyto dva vrcholy, které mají dohromady $2b + 1$ dětí, můžeme rozdělit na tři vrcholy, kde každý má $2b/3$ dětí. (Jednoho sourozence máme vždy, neboť $a \geq 2$.)

Ale takto můžeme spojit tři malé vrcholy do dvou - pro každý vrchol, který má po odebrání $a - 1$ synů se nejprve pokusíme ukrást jednoho syna svému sourozenci. Pokud oba sousední sourozenci mají přesně $2b/3$ dětí, pak je spojíme do dvou vrcholů. Tady je také potřeba dát si speciální pozor na případ, kdy jsme odebrali syna nejlevějšímu vrcholu - pak lze například ukrást jednoho syna svému sousedovi, a přesunout se k němu, čímž převádíme na případ s oběma sousedními sourozenci.

Úloha 4 ((a, b)-JOIN)

Navrhnete operaci JOIN pro (a, b)-stromy: máte tedy dva stromy T_1, T_2 s tím, že všechny klíče v T_1 jsou menší než v T_2 , a cílem operace je spojit dva stromy do jednoho. Pozor na to, že stromy mohou být různé vysoké. (Pro analýzu SPLITu se může hodit analyzovat složitost přesněji než jen $\mathcal{O}(\log n)$.) Také se můžete (stejně jako poté u SPLITu) omezit na ($a, 2a$)-stromy.

Řešení

Máme dva stromy, jeden z nich je potenciálně vyšší (BÚNO třeba levý strom není vyšší než pravý). Co tedy uděláme je, že najdeme nejlevější vrchol na vhodné úrovni pravého stromu, a spojíme ho s kořenem levého stromu. Tím nám může jeden vrchol potenciálně hodně narůst, ale ne na víc než $2b$, takže potom můžeme splitovat jako u insertu.

Abychom tohle implementovali, potřebujeme vědět něco navíc: jednak bychom měli vědět hloubku každého stromu (ekvivalentně výšku kořene), abychom nemuseli vždycky zjišťovat hloubku projitím stromu až do listu – tu je ovšem jednoduché udržovat, protože se mění jenom, když se nám kořen štěpí nebo se naopak slepuje s vrcholy pod ním.

Dále ovšem máme druhý problém: protože jsme ve verzi s hodnotami ve všech vrcholech, musíme ve skutečnosti mezi kořen levého stromu a nejlevější vrchol pravého stromu přidat nějakou hodnotu, která se ve stromě vyskytuje (jinak bychom měli mezi dvěma klíči dvě děti). To vyřešíme tak, že si budeme u každého stromu udržovat ukazatel na nejlevější vrchol v nejnižší vrstvě, a zároveň si budeme udržovat invariant, že v tomto vrcholu (obsahující mj. minimum ve stromě) máme alespoň $a + 1$ synů, abychom vždycky věděli, že můžeme minimum takto přesunout bez jakékoliv další práce (to budeme řešit přesouváním klíčů od sourozenců zprava a případně vhodným mergováním klíče z rodiče¹).

Důležité pozorování je, že po joinu nemusíme dělat žádnou další práci, a ukazatel na minimum stále máme: na minimum v levém podstromě jsme nikdy nesahali, a to je po joinu stále naše minimum celého stromu.

Složitost pak je $\mathcal{O}(\log(|T_2|) - \log(|T_1|))$ – stačí, abychom se vždycky podívali jenom do úrovně $\log(|T_2|) - \log(|T_1|)$ na připojení, a pak stejně bubláme nahoru.

Bonusové úlohy

Úloha 5 ((a, b) -SPLIT)

Navrhněte operaci SPLIT: máte tedy (a, b) -strom T a klíč k , a chcete T rozdělit na dva stromy tak, že je v jednom je vše menší než k a ve druhém je zbytek. Chceme, aby tato operace běžela v čase $\mathcal{O}(\log n)$.

Řešení

Najdeme k , podle kterého rozdělujeme (když tam není, dojdeme až do listu). Potom vezmeme všechny podstromy napravo od cesty z kořene do k , a oddělíme je od původního stromu. (Ty oddělené stromy jsou korektní (a, b) -stromy, protože jsme potenciálně rozbili jenom kořeny každého odděleného stromu.)

Napravo odsekané stromy můžeme sjednotit přes join (časová složitost vyjde logaritmická, protože jeden join je v čase rozdílu hloubek, což vyteleskopuje na $\mathcal{O}(\log n)$).

Ještě musíme opravit strom nalevo, což uděláme, jako bychom dělali delete.

Celkem tedy máme složitost $\mathcal{O}(\log n)$. V obou případech si můžeme navíc díky této složitosti dovolit ještě průchod do nejlevějšího listu, abychom si poupravili vrchol s minimem.

¹U $(a, 2a - 1)$ -stromů to nemusí vyjít, takže předpokládáme $b \geq 2a$, což je stejně praktické pro rozumný počet modifikací vrcholů.