Sada: 1 Příklad: 3 IV003 Algoritmy a datové struktury II

Jméno: Karel Kubíček UČO: 408351

Jméno: Henrich Lauko UČO: 410438

Datovou strukturou, která splňuje podmínky stanovené zadáním je B+ strom se stupněm 4. Každý uzel bude mít tyto atributy: $p[x]$ je ukazatel na podstrom x, pp je ukazatel na rodiče a $small[x]$ je klíč odpovídající nejmenší hodnotě v podstromu x. Naše řešení předpokládá, že funkce dostanou na vstup námi vytvořený strom a ne libovolný 2, 3, 4 strom.

1. Minimum hledáme vždy v nejlevějším podstromu. Rekurzivně se tedy volá Minimum na nejlevější podstrom, dokud se nedosáhne úrovně listů, kde nejlevější potomek je minimum našeho stromu.

   Procedura Minimum(T)

   vstup: 234 strom T, ve kterém máme minimum hledat
   výstup: ukazatel na minimum
   if T je list **then**
       return T *// nalezeno minimum, vracím ukazatel*
   else
       return Minimum ($T.p0$) *// T.p0 je ukazatel na nejlevější podstrom*

   Časová složitost hledání minima je logaritmická, jelikož výška stromu je logaritmická (bude dokázáno níže u insert) a rekurze se v každém zanoření volá na strom s výškou o jedna menší.

2. Insert (k) se skládá ze 2 částí. První částí je nalezení správného místa, na které prvek patří a zařazení na toto místo, druhou fází je v případě překročení limitu 4 potomci na strom rozdělení uzlu s 5 potomky na 2 uzly, jeden se dvěma, druhý se třema potomky a vypropagování kontroly velkého počtu potomků o úroveň výše.

   Procedura Insert(T, k)

   vstup: 234 strom T, do kterého máme přidat klíč k
   $x \leftarrow$ SelectKey (k) *// určí, do kterého podstromu patří zadaný klíč ($k > small[x] \land k \leq small[x+1]$)*
   if T je list **then**
       vlož k na místo za $small[x]$ a posuň zbylé klíče doprava
       ControlNode (T)
   else
       Insert ($T.p[x], k$)

Jméno: Karel Kubíček UČO: 408351

Jméno: Henrich Lauko UČO: 410438

Procedura ControlNode(T)

vstup: 234 strom T, jehož kořenový uzel kontrolujeme
výstup:
if počet klíčů v T je 5 **then**
  rozděl uzel na uzly se 3 a 2 klíči
  if existuje rodič **then**
    vytvoř nový klíč v rodičovském uzlu // *na 1. už existuje, vytvoří se na 2. dvojprvkový uzel*
  else
    vytvoř nového rodiče
    přidej oba ukazatele na rozdelené uzly do rodiče
  ControlNode ($T.pp$) // *zkontroluj rodičovský uzel*
if počet klíčů v T je 1 **then**
  spoj uzel se sousedním uzlem s méně klíči
  odstraň z rodičovského uzlu klíč pro uzel T
  ControlNode ($T.pp.p[x]$) // *zkontroluj uzel vzniklý spojením*

I insert je logaritmické asymptotické časové složitosti. První část algoritmu udělá maximálně $\log_2(n)$ kroků při hledání správného listu, do kterého má klíč zařadit, následující procedura ControlNode v případě plné zaplněnosti všech rodičovských uzlů udělá maximálně $\log_4(n)$ (ale pokud by se dělení uzlů vypropagovalo až ke kořeni, pak by i první část musela proběhnout v $\log_4(n)$ krocích). To dohromady patří do $\mathcal{O}(\log(n))$

3. Delete(x)

Procedura Delete(T)

vstup: 234 strom T, který máme vymazat
výstup:
odstraň z rodičovského uzlu klíč pro uzel T
UpdateKey () // *projde uzly ke kořeni a opraví v nich hodnoty small[x] jestli se změnili*
ControlNode ($T.pp$)
smaž T

Podobná procedura, jako insert, tentokrát nutno kontrolovat, jestli není v rodičovském uzlu klíčů málo (1). To, že operace patří do $\mathcal{O}(\log(n))$

4. DecreaseKey (x, k) se dá realizovat posloupností procedur Delete (x) a následného Insert (k). To v součtu znamená složitost $2 \cdot \log(n) \in \mathcal{O}(\log(n))$.

5. Extract Min se dá realizovat složením procedur Delete (Minimum). To v součtu znamená složitost $2 \cdot \log(n) \in \mathcal{O}(\log(n))$.

U důkazů složitosti jsem vždy předpokládal, že strom bude mít výšku $\log_4(n)$ až $\log_2(n)$. Pokud by tato vlastnost neplatila, pak by složitost operací neodpovídala zadání. Pro

Jméno: Karel Kubíček UČO: 408351

Jméno: Henrich Lauko UČO: 410438

dokázání logaritmické výšky musíme dokázat korektnost procedůry ControlNode. Pro platnost předpokádejme, že počáteční stav stromu jsou alespoň 2 uzly. ControlNode nám zajišťuje, že v každém uzlu budou 2,3 nebo 4 podstromy. Zvýšení výšky nastáva jenom v případe, že vkládame 4^{h-1}. klíč. To nám zaručuje, že všechny klíče jsou v listech stejné úrovně což společne s podmínkou, že v uzlech bodou právě 2,3 nebo 4 podstromy dáva logaritmickou výšku.