SEMESTRÁLNA PRÁCA S1

Algoritmy a údajové štruktúry 2

Benjamín Koša

Fakulta riadenia a informatiky | Žilinská univerzita

Obsah

Vyváženie stromu

Pri vymýšľaní algoritmu na vyvažovanie stromu sme si stanovili základnú myšlienku: "**aby bol strom dokonale vyvážený musí byť pre každý podstrom medián uložený na správnom mieste"**. Z nej nám vyplývajú dve otázky. Ako nájsť medián pre každý podstrom? Ako dostať medián na svoje miesto?

Ako nájsť medián pre každý podstrom?
Medián sa nachádza v strede poľa. Ak ma pole 100 prvkov na indexe (100-1)/2 sa nachádza medián. Tento výpočet nám našiel prvý medián a rozdelil pole na dve menšie polia(môžeme si predstaviť aj ako podstromy). Pre tieto polia ďalej počítame mediány až kým sa nedostaneme na polia o veľkosti 1.

Ako dostať medián na svoje miesto?
Pomocou **inorder prehliadky** dostaneme zoradene pole prvkov. Pomocou algoritmu na hľadanie medianov vyberieme prvok z inorder poľa. Vyhľadáme ho v strome a pomocou rotácii ho presúvame hore až kým sa dostane do koreňa alebo kým otec vyvažovaného prvku neje uložený na správnom mieste. Každý uzol ma boolean premennú v ktorej si značíme či je už uzol uložený na správnom mieste.

Po vykonaní vyváženia sme urobili testy na milión rôznych stromoch s počtom prvkov 10 potom 100, 1000 a 10000. Pre každý strom sme si vypočítali teoretickú najlepšiu výšku a porovnali ju s reálnou výškou stromu a **v každom prípade sa výšky zhodovali**.

Zložitosť:

Premenne:
N – počet prvkov v poli
Zložitosti
O(N) In order
O(N) Získanie medianov
O(Log(N)) Vyhľadanie prvku v strome
O(log(N/2)) Ulozenie prvku na svoje miesto:
*N/2 pretože cesta k poslednému uloženému prvku sa stále zmenšuje

Celková zložitost: 2*N+N*(log(N)+ log(N/2))

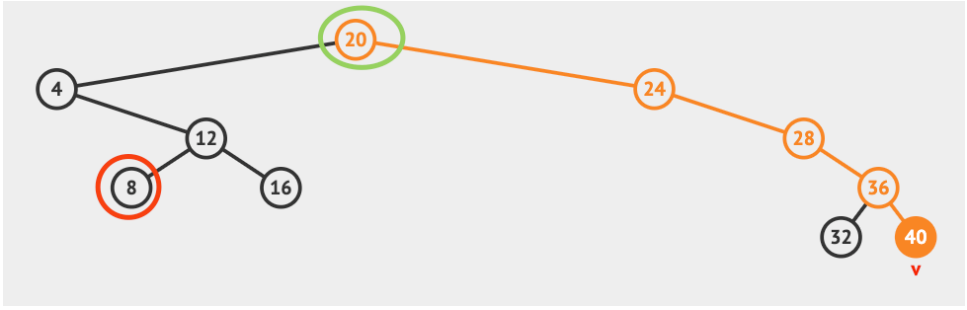
Ukážka

Krok 1

- Získanie inorder a mediánov:
  Inorder: {4, 8, 12, 16, 20, 24, 28, 32, 36, 40}
  Medián: {4, 1, 7, 0, 2, 5, 8, 3, 6, 9}
- Vyhľadáme v strom prvok inorder[medián[0]] tj. 20 a presúvame ho hore kým otec nie je null alebo nie je na svojom mieste
- Poznačíme si že node 20 je už uložený na svojom mieste

Krok 2

- Vyhľadáme v strom prvok inorder[medián[1]] tj. 8 a presúvame ho hore kým otec nie je null alebo nie je na svojom mieste
- Poznačíme si že node 8 je už uložený na svojom mieste

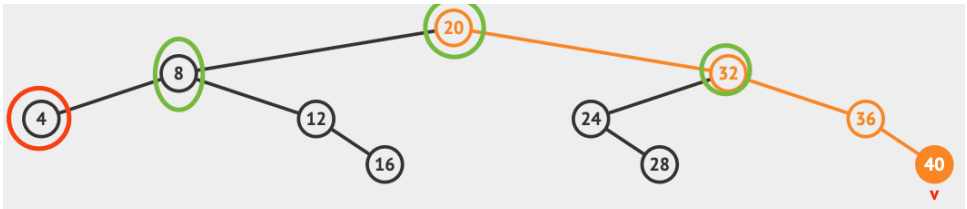


Krok 3

- Vyhľadáme v strom prvok inorder[medián[2]] tj. 32 a poznačíme

Krok 4

- Vyhľadáme v strom prvok inorder[medián[3]] tj. 4 a poznačíme



... krok 10

Vylepšenie operácii insert, find a remove

Insert

Po pridaní prvku C získame prvok B ako B = C.parent, a prvok A ako A = B.parent. Ak prvky B, A majú jedného syna vykonáme rotáciu podľa prvku A.

Find

Používame rovnakú metódu ako pri insert ale pre nájdený prvok.

Remove

Pred vymazaním prvku spustíme rovnakú metódu ako pre insert ale pre otca vymazávaného prvku.

Zložitosť vylepšenia je O(1)

Testovanie insert 1 000 000 stromov s 1000 prvkami

```
height: 16, occurrences: 34
height: 17, occurrences: 1215
height: 18, occurrences: 8397
height: 19, occurrences: 20616
height: 20, occurrences: 25735
height: 21, occurrences: 21124
height: 22, occurrences: 12654
height: 23, occurrences: 6116
height: 24, occurrences: 2607
height: 25, occurrences: 1007
height: 26, occurrences: 354
height: 27, occurrences: 107
height: 28, occurrences: 27
height: 29, occurrences: 5
height: 30, occurrences: 2
average size: 20.43
result: no errors
```

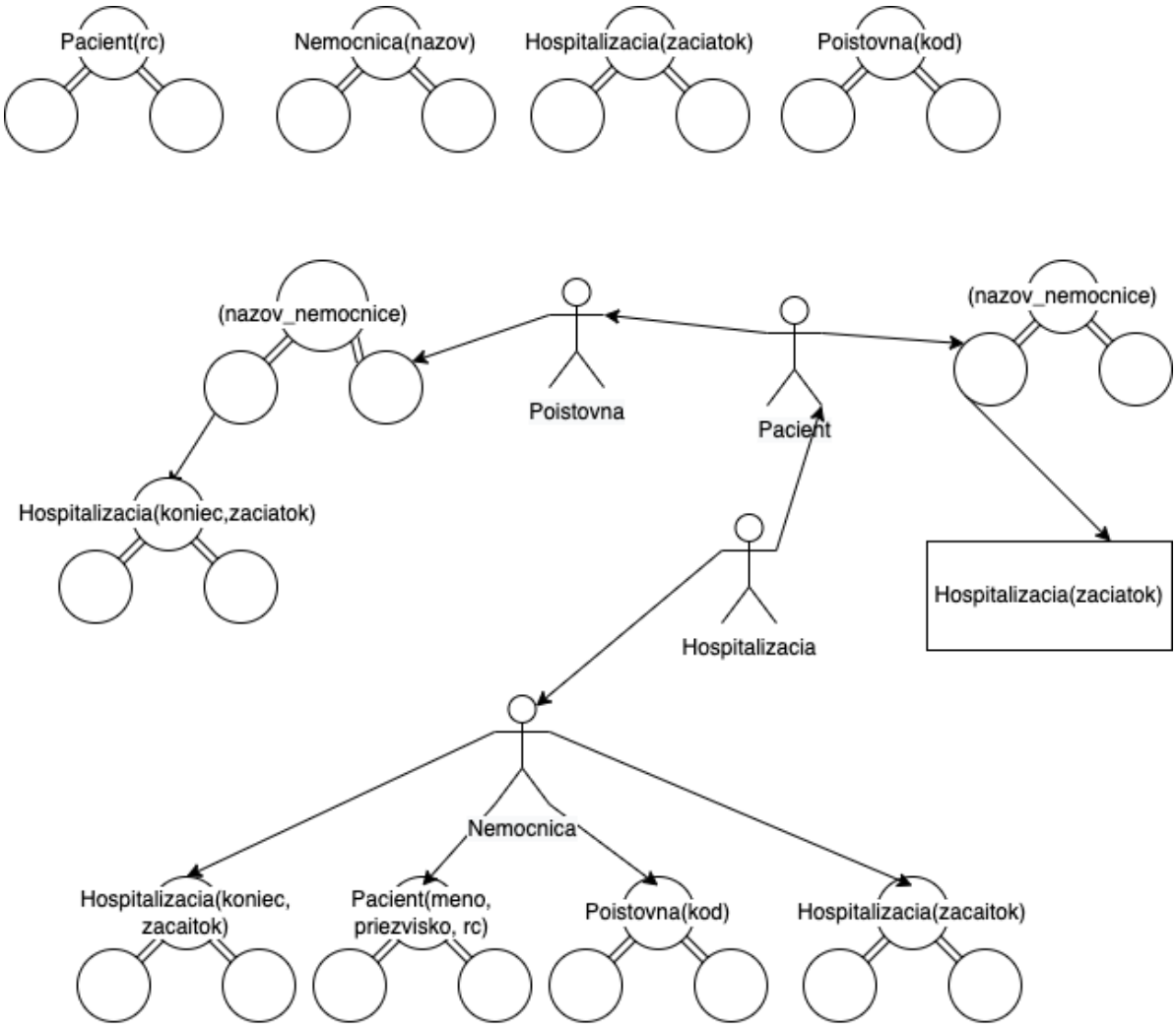
Obrázok 1

```
height: 17, occurrences: 47
height: 18, occurrences: 909
height: 19, occurrences: 5695
height: 20, occurrences: 14690
height: 21, occurrences: 21319
height: 22, occurrences: 20947
height: 23, occurrences: 15877
height: 24, occurrences: 10093
height: 25, occurrences: 5538
height: 26, occurrences: 2770
height: 27, occurrences: 1227
height: 28, occurrences: 543
height: 29, occurrences: 212
height: 30, occurrences: 87
height: 31, occurrences: 30
height: 32, occurrences: 12
height: 33, occurrences: 4
average size: 22.05
result: no errors
```

Obrázok 2

Testovanie prebiehalo na rovnakých seedoch. Po zostavení stromu sme zmerali jeho výšku. Na prvom obrázku vidíme výsledok testovania s vylepšením na druhom bez vylepšenia. Vylepšením sme znížili výšku stromu v priemere o 1.62.

Návrh informačného systému



Vysvetlivky

Zložitosti operácii

1. vyhľadanie záznamov pacienta (identifikovaný svojím rodným číslom) v zadanej nemocnici (identifikovaná svojím názvom). Po nájdení pacienta je potrebné zobraziť všetky evidované údaje.

Premenne:
P: počet všetkých pacientov
N: počet nemocníc v ktorých bol už pacient evidovaný
Zložitosti
Log(P): Vyhľadanie pacienta
Log(N): Vyhľadanie nemocnice
O(1): získanie referencie na všetky hospitalizácie pacienta v nemocnici
Celková zložitosť:
Log(P)+log(N)

2. vyhľadanie záznamov pacienta/ov v zadanej nemocnici (identifikovaná svojím názvom) podľa mena a priezviska. Po nájdení pacienta/ov je potrebné zobraziť všetky evidované údaje zo zadanej nemocnice rozčlenené po pacientoch.

Premenne
N: počet nemocníc
P: počet pacientov v nemocnici
I: interval vyhľadaných pacientov
Zložitosti
Log(N): Vyhľadanie nemocnice
Log(p)+I: získanie intervalu pacientov
Celková zložitosť
Log(N)+log(P)+I

3. vykonanie záznamu o začiatku hospitalizácie pacienta (identifikovaný svojím rodným číslom) v nemocnici (identifikovaná svojím názvom)

Premenne
P: počet pacientov
PN: počet nemocníc pacienta
PNH: počet hospitalizácii v nemocnici pacienta
H: počet hospitalizácii
N: počet nemocníc
NH: počet hospitalizácii v nemocnici
NP: počet pacientov v nemocnici
NO: počet poisťovni v nemocnici
ON: počet nemocníc v poisťovni
ONH: počet hospitalizácii evidovaných v poisťovni pre nemocnicu
Zložitosti
Log(P) vyhľadanie pacienta
Log(N) vyhľadanie nemocnice
Log(H) pridanie do hospitalizácii
Log(PN) vyhľadanie nemocnice pacienta
Log(PNH) pridanie do hospitalizácii pacienta v nemocnici

Log(NH) pridanie hospitalizácie do nemocnice
Log(NP) pridanie pacienta do nemocnice
Log(NO) pridanie poisťovne do nemocnice
Log(ON) pridanie nemocnice do poisťovni
Log(ONH) pridanie hospitalizácie do poisťovne v nemocnici
Celková zložitosť
Log(P)+Log(N)+Log(H)+Log(PN)+Log(PNH)+Log(NH)+Log(NP)+Log(NO)+Log(ON)+Log(ONH)

4. vykonanie záznamu o ukončení hospitalizácie pacienta (identifikovaný svojím rodným číslom) v nemocnici (identifikovaná svojím názvom)

Premenne
P: počet všetkých pacientov
N: počet nemocníc pacienta
NH: počet všetkých hospitalizácii pacienta v nemocnici
Zložitosti
Log(P) Vyhľadanie pacienta
Log(N) vyhľadanie nemocnice pacienta
Celková zložitosť
Log(P)+Log(N)+NH

5. **výpis hospitalizovaných pacientov v nemocnici (identifikovaná́ svojim názvom) v zadanom časovom období (od, do)**

Premenne
N počet nemocníc
H počet hospitalizácií v nemocnici
HI počet hospitalizácií v nemocnici od minima po koniec hľadaného intervalu
Zložitosti
Log(N) vyhľadanie nemocnice
Log(H) vyhľadanie minima
2*HI získanie intervalu a odfiltrovanie hospitalizácii ukončených pred začiatkom intervalu
celková zložitosť
Log(N)+Log(H)+2*HI

6. pridanie pacienta

Premenne
P: pocet pacientov
O pocet poistovni
Zložitosti
Log(P) vlozenie pacienta
Log(O) vyhladanie poistovne
Celkova zložitost
Log(P)+Log(O)

7. vytvorenie podkladov pre účtovné oddelenie na tvorbu faktúr pre zdravotné poisťovne za zadaný mesiac. Pre každú poisťovňu, ktorej pacient (pacienti) bol v zadaný kalendárny mesiac hospitalizovaní aspoň jeden deň je potrebné pripraviť podklady obsahujúce:

- kód zdravotnej poisťovne
- počet dní hospitalizácii (za všetkých pacientov – napr. 98 dní)
- výpis hospitalizovaných pacientov **v jednotlivé dni mesiaca** spolu s diagnózami

Premenne
N: počet nemocníc
P: počet poisťovni
H počet hospitalizácii v nemocnici
I: počet hospitalizácii od začiatku po hornú hranicu hľadaného intervalu
Zložitosti
Log(H) vyhľadanie začiatku intervalu
celková zložitosť
N*P*(log(H)+2*I)

8. výpis aktuálne hospitalizovaných pacientov vnemocnici (identifikovaná svojím názvom)

Premenne
N počet nemocníc
P počet pacientov v nemocnici
PI počet pacientov v nemocnici s neukončenou hospitalizáciou
Zložitosti
Log(N) vyhľadanie nemocnice
Log(P) vyhľadanie začiatku intervalu
celková zložitosť
Log(N)+Log(P)+PI

8. výpis aktuálne hospitalizovaných pacientov v nemocnici (identifikovaná svojím názvom), ktorí sú poistencami zadanej zdravotnej poisťovne (identifikovaná svojím kódom)

Premenne
N: počet nemocníc
P: počet pacientov hospitalizovaných v nemocnici
Zložitosti
Log(N) vyhľadanie nemocnice
celková zložitosť
Log(N)+P

9. výpis aktuálne hospitalizovaných pacientov v nemocnici (identifikovaná svojím názvom) zotriedený podľa rodných čísel, ktorý sú poistencami zadanej zdravotnej poisťovne (identifikovaná svojím kódom)

Premenne
N: počet nemocníc
P: počet pacientov hospitalizovaných v nemocnici
Zložitosti
Log(N) vyhľadanie nemocnice
P*log(P)utriedenie pacientov pomocou arraylist.sort
celková zložitosť

Log(N)+ P+P*log(P)
pridanie nemocnice
Premenne
N počet nemocníc
Zložitosti
Log(N) vloženie nemocnice
celková zložitosť
Log(N)

13. výpis nemocníc usporiadaných podľa názvov

Premenne
N počet nemocníc
celková zložitosť
N

14. zrušenie nemocnice (celá agenda sa presunie do inej nemocnice, ktorú špecifikuje používateľ (identifikovaná svojím názvom), vrátane pacientov a historických záznamov)

Premenne
N počet nemocníc
H počet hospitalizácii v nemocnici na zamazanie
P počet pacientov v zmazávanej nemocnici
O počet poisťovni v zmazávanej nemocnici
OH počet hospitalizácii v poisťovni pre nemocnicu
PH počet hospitalizácii pacienta
Zložitosti
2*Log(N) vyhľadanie nemocnice na zmazanie a druhej nemocnice
H zmeniť všetkým hospitalizáciám pointer na novu nemocnicu
2*PH získanie hospitalizácii a pridanie do druhej nemocnice
2*OH získanie hospitalizácii a pridanie do druhej nemocnice
2*H získanie hospitalizácii a pridanie do druhej nemocnice
2*O získanie poisťovni a pridanie do druhej nemocnice
2*P získanie pacientov a pridanie do druhej nemocnice
celková zložitosť
2*(log(N)+PH+OH+H+O+P)+H

Zdroje

Zložitosť sortu v arrayliste
https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html#sort%28java.util.List,%20java.util.Comparator%29