Algoritmi i strukture podataka
vežbe 4

Mirko Stojadinović

27. oktobar 2013

1 Hip

Hip je binarno stablo koje zadovoljava uslov hipa: ključ svakog čvora veći je ili jednak od ključeva njegovih sinova. Pored ključa, čvor obično sadrži i druge podatke. Hip se još naziva i **lista sa prioritetom**.

Hip se može realizovati implicitno i eksplicitno. Algoritmi koji su ovde predstavljeni rade sa **implicitnom** realizacijom hipa. Dakle, ako hip ima n elemenata, onda se za smeštanje elemenata koriste lokacije u nizu A sa indeksima $A[1..n]$, tako da ako element indeksa i predstavlja čvor stabla, tada
- element indeksa $2 * i$ predstavlja levo dete čvora
- element indeksa $2 * i + 1$ predstavlja desno dete čvora.

Na primer, deca čvora $A[1]$ su elementi $A[2], A[3]$. Gledano obratno, roditelj elementa sa indeksom j je čvor sa indeksom $\lfloor j/2 \rfloor$. Na hipu su definisane sledeće **2 operacije**:

1.1 Uklanjanje najvećeg elementa iz hipa

Neka je dat hip A sa n elemenata. Treba ukloniti ključ $A[1]$, a potom transformisati niz A tako da opet zadovoljava svojstvo hipa. Najzgodnije je u zameniti vrednosti elemenata $A[1]$ i $A[n]$ i smanjiti veličinu hipa za 1. Problem je što tada možda nije zadovoljeno pravilo hipa. Dovođenje niza u stanje koje zadovoljava pravilo hipa vrši se ponavljanjem sledeća 2 koraka:

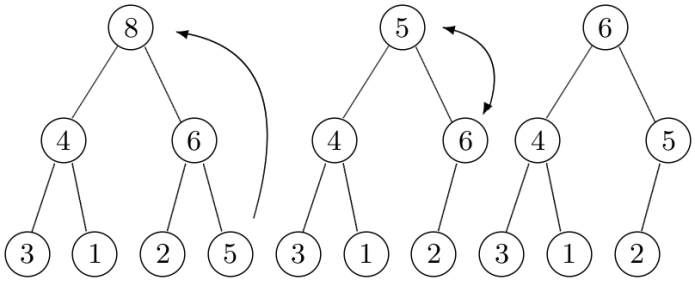
1. Analizirati decu korena hipa koji se trenutno posmatra i odrediti koje dete je veće.

2. Ako je ključ korena veći od ključa većeg deteta postupak je završen; inače zameniti ključeve u korenu i većem detetu i preći na korak 1 ovog puta sa manjim hipom (to je hip u koji se prethodni koren spustio).

Na primer, postupak uklanjanja korena iz hipa koji je implicitno predstavljen nizom 8 4 6 3 1 2 5 je:

```
8 4 6 3 1 2 5
5 4 6 3 1 2 8
5 4 6 3 1 2
6 4 5 3 1 2
```

dok se isti postupak može predstaviti i eksplicitnim stablima:



1.2 Upis elementa u hip

Neka je dat hip A sa n elemenata i element x koji je potrebno dodati u hip. Ako želimo dodati element u prioritetni red možemo ga dodati na kraj niza A. To je ekvivalentno dodavanju krajnjeg desnog lista u stablu. Ta operacija ima za posledicu da se n uveća za 1, ali i da možda stablo ne zadovoljava pravilo hipa. Da bi se zadovoljilo pravilo hipa koriste se naredna 2 koraka:

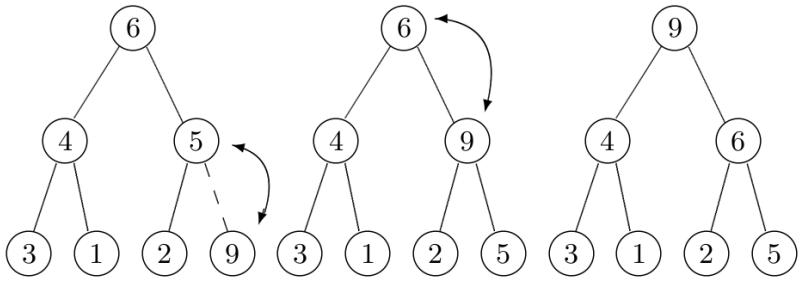
1. Ako je ključ roditelja čvora koji sadrži element x manji od x, zamenjuje se sadržaj ova dva čvora.

2. Ako je ključ roditelja praznog čvora veći od elementa x, onda je postupak završen. Inače preći na korak 1.

Prikaz ovog postupka sa stablom koje je implicitno zadato nizom 6 4 5 3 1 2, u koje se umeće broj 9 je:

```
6 4 5 3 1 2
6 4 5 3 1 2 9
6 4 9 3 1 2 5
9 4 6 3 1 2 5
```

dok se isti postupak može predstaviti i eksplicitnim stablima:



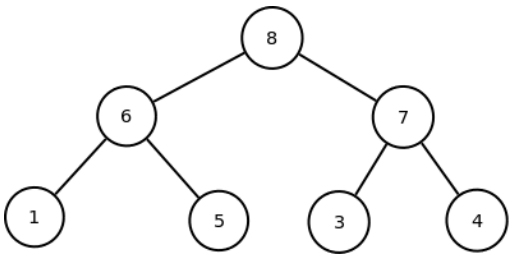
1. Odrediti izgled implicitno predstavljenog hipa koji se dobija umetanjem redom brojeva 5, 1, 3, 9, 6, 4, 7, 8 polazeći od praznog hipa. Prikazati izgradnju hipa tabelom, čiji svaki red odgovara jednoj promeni hipa. Zatim odrediti izgled hipa dobijenog uklanjanjem najvećeg elementa.

```
Korak 1  dodaj 5
5
Korak 2 - dodaj 1
5 1
Korak 3: - dodaj 3
5 1 3
Korak 4  - dodaj 9 i dva podkoraka preuredjivanja hipa
5 1 3 9
5 9 3 1
9 5 3 1
Korak 5  dodaj 6 i jedan podkorak preuredjivanja hipa
9 5 3 1 6
9 6 3 1 5
Korak 6  dodaj 4 i jedan podkorak preuredjivanja hipa
9 6 3 1 5 4
9 6 4 1 5 3
Korak 7  dodaj 7 i jedan podkorak preuredjivanja hipa
9 6 4 1 5 3 7
9 6 7 1 5 3 4
Korak 7  dodaj 8 i dva podkoraka preuredjivanja hipa
9 6 7 1 5 3 4 8
9 6 7 8 5 3 4 1
9 8 7 6 5 3 4 1

Uklanjanje max elementa 9
```

```
Korak 1 - ukloni 9, tako sto se zadnji element kopira u koren, i n smanji za 1
9 8 7 6 5 3 4 1
1 8 7 6 5 3 4
Korak 3  preuredjivanje hipa, max (8,7) zameni sa 1
1 8 7 6 5 3 4
8 1 7 6 5 3 4
Korak 4  preuredjivanje hipa, max (6,5) zameni sa 1
8 1 7 6 5 3 4
8 6 7 1 5 3 4
```

a izgled ovog stabla je:



2. Konstruisati algoritam za formiranje hipa koji sadrži sve elemente dva hipa veličine n i m. Hipovi su predstavljeni eksplicitno (svaki čvor ima pokazivače na svoja dva sina) i može biti praznih pozicija i iznad zadnjeg nivoa. Vremenska složenost treba da bude u najgorem slučaju $O(m+n)$.

REŠENJE: Opis algoritma u 2 koraka:

1. Uklanja se veći od 2 korena i izvedu se popravke tog hipa na sličan način opisan onom u sekciji uklanjanja elementa iz hipa (linearno vreme za pronalazak nekog lista i linearno da se novi koren spusti niz stablo, jer može biti nepopunjenih pozicija, tj. u najgorem slučaju je stablo lista).

2. Izabrani element se umeće kao koren novog hipa, tako da njegovi sinovi budu koreni dva stara hipa (O(1)).

Za sve ovo je potrebno $O(max\{m,n\})$ koraka što je manje od $O(m+n)$ koraka.

1.3 Hipsort

Algoritam se sastoji iz dva dela:

1. Elementi iz niza dimenzije n se smeštaju u hip. Ovo se može uraditi na dva načina:

   (a) pristupom odozgo-nadole čija je složenost $O(n \log n)$.

   (b) pristupom odozdo-nagore čija je složenost $O(n)$.

2. Elementi se uklanjaju iz hipa, a svako uklanjanje ima složenost proporcionalnu visini hipa, tj. $O(\log n)$ pa posto niz ima n elemenata ukupna složenost je $O(n \log n)$.

3. Koristeći pristupe odozgo-nadole i odozdo-nagore algoritmom hipsort sortirati niz brojeva 5 3 8 6 7 2 1 9 4.
REŠENJE:

PRISTUP ODOZGO-NADOLE
1. deo - pravljenje hipa odozgo-nadole (elementi se redom dodaju u hip gore opisanim algoritmom)

```
5
5 3
5 3 8
8 3 5
8 3 5 6
8 6 5 3
8 6 5 3 7
8 7 5 3 6
8 7 5 3 6 2
8 7 5 3 6 2 1
8 7 5 3 6 2 1 9
8 7 5 9 6 2 1 3
8 9 5 7 6 2 1 3
9 8 5 7 6 2 1 3
9 8 5 7 6 2 1 3 4
```

2. deo (uklanjanje najvećeg elementa iz hipa dok se ne uklone svi elementi):

```
9 8 5 7 6 2 1 3 4
4 8 5 7 6 2 1 3|9
8 4 5 7 6 2 1 3|9
8 7 5 4 6 2 1 3|9
3 7 5 4 6 2 1|8 9
7 3 5 4 6 2 1|8 9
7 6 5 4 3 2 1|8 9
1 6 5 4 3 2|7 8 9
6 1 5 4 3 2|7 8 9
6 4 5 1 3 2|7 8 9
2 4 5 1 3|6 7 8 9
5 4 2 1 3|6 7 8 9
3 4 2 1|5 6 7 8 9
4 3 2 1|5 6 7 8 9
1 3 2|4 5 6 7 8 9
3 1 2|4 5 6 7 8 9
2 1|3 4 5 6 7 8 9
```

```
1|2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
```

PRISTUP ODOZDO-NAGORE:
1. deo - pravljenje hipa odozdo-nagore (za detalje pogledati profesorovu knjigu)

```
5 3 8 6 7 2 1 9 4
5 3 8 9 7 2 1 6 4
5 3 8 9 7 2 1 6 4
5 9 8 3 7 2 1 6 4
5 9 8 6 7 2 1 3 4
9 5 8 6 7 2 1 3 4
9 7 8 6 5 2 1 3 4
```

2. deo (uklanjanje najvećeg elementa iz hipa dok se ne uklone svi elementi):

```
9 7 8 6 5 2 1 3 4
4 7 8 6 5 2 1 3|9
8 7 4 6 5 2 1 3|9
3 7 4 6 5 2 1|8 9
7 3 4 6 5 2 1|8 9
7 6 4 3 5 2 1|8 9
1 6 4 3 5 2|7 8 9
6 1 4 3 5 2|7 8 9
6 5 4 3 1 2|7 8 9
2 5 4 3 1|6 7 8 9
5 2 4 3 1|6 7 8 9
5 3 4 2 1|6 7 8 9
1 3 4 2|5 6 7 8 9
4 3 1 2|5 6 7 8 9
2 3 1|4 5 6 7 8 9
3 2 1|4 5 6 7 8 9
1 2|3 4 5 6 7 8 9
2 1|3 4 5 6 7 8 9
1|2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
```

4. Koristeći pristupe odozgo-nadole i odozdo-nagore algoritmom hipsort sortirati niz brojeva 1 6 4 2 9 3 5 7 8.

REŠENJE:

PRISTUP ODOZGO-NADOLE
1. deo (pravljenje hipa):

```
1
1 6
6 1
```

```
6 1 4
6 1 4 2
6 2 4 1
6 2 4 1 9
6 9 4 1 2
9 6 4 1 2
9 6 4 1 2 3
9 6 4 1 2 3 5
9 6 5 1 2 3 4
9 6 5 1 2 3 4 7
9 6 5 7 2 3 4 1
9 7 5 6 2 3 4 1
9 7 5 6 2 3 4 1 8
9 7 5 8 2 3 4 1 6
9 8 5 7 2 3 4 1 6
```

2. deo (uklanjanje najvećeg elementa iz hipa dok se ne uklone svi elementi):

```
9 8 5 7 2 3 4 1 6
6 8 5 7 2 3 4 1|9
8 6 5 7 2 3 4 1|9
8 7 5 6 2 3 4 1|9
1 7 5 6 2 3 4|8 9
7 1 5 6 2 3 4|8 9
7 6 5 1 2 3 4|8 9
4 6 5 1 2 3|7 8 9
6 4 5 1 2 3|7 8 9
3 4 5 1 2|6 7 8 9
5 4 3 1 2|6 7 8 9
2 4 3 1|5 6 7 8 9
4 2 3 1|5 6 7 8 9
1 2 3|4 5 6 7 8 9
3 2 1|4 5 6 7 8 9
1 2|3 4 5 6 7 8 9
2 1|3 4 5 6 7 8 9
1|2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
```

PRISTUP ODOZDO-NAGORE
1. deo (pravljenje hipa):

```
1 6 4 2 9 3 5 7 8
1 6 4 8 9 3 5 7 2
1 6 5 8 9 3 4 7 2
1 9 5 8 6 3 4 7 2
9 1 5 8 6 3 4 7 2
9 8 5 1 6 3 4 7 2
9 8 5 7 6 3 4 1 2
```

2. deo (uklanjanje najvećeg elementa iz hipa dok se ne uklone svi elementi):

```
9 8 5 7 6 3 4 1 2
2 8 5 7 6 3 4 1|9
8 2 5 7 6 3 4 1|9
8 7 5 2 6 3 4 1|9
1 7 5 2 6 3 4|8 9
7 1 5 2 6 3 4|8 9
7 6 5 2 1 3 4|8 9
4 6 5 2 1 3|7 8 9
6 4 5 2 1 3|7 8 9
3 4 5 2 1|6 7 8 9
5 4 3 2 1|6 7 8 9
1 4 3 2|5 6 7 8 9
4 1 3 2|5 6 7 8 9
4 2 3 1|5 6 7 8 9
1 2 3|4 5 6 7 8 9
3 2 1|4 5 6 7 8 9
1 2|3 4 5 6 7 8 9
2 1|3 4 5 6 7 8 9
1|2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
```

1.4 Određivanje medijane

Medijana prestavlja srednji po veličini u datom nizu brojeva. Ukoliko je broj elemenata niza paran, medijanom smatramo aritmetičku sredinu dva po veličini središnja elementa. Zadatak je da se korišćenjem 2 hipa postigne da se određivanje medijane niza izvršava u konstantnom vremenu, pri čemu se u niz mogu dodavati novi elementi.

Ovaj zadatak se rešava tako što se održavaju 2 hipa. Ukoliko niz ima paran broj elemenata onda hipovi sadrže po polovinu ovih elemenata a ukoliko je broj elemenata neparan dozvoljeno je da jedan od hipova ima tačno jedan element više od drugog. U prvi hip smeštamo manje a u drugi veće brojeve. Prvi hip je takav da je broj u svakom čvoru veći ili jedna od brojeva u sinovima a u drugom hipu je broj u čvoru uvek manji ili jednak od brojeva u sinovima. To znači da se u slučaju neparnog broja elemenata, u hipu u kome ima jedan element više nalazi medijana. U slučaju parnog broja elemenata, medijana se računa po formuli (koren1+koren2)/2.

Kako dodati novi element u niz. Razlikujemo više slučajeva:

1. ako je element manji od korena prvog hipa, dodati ga u prvi hip.
2. ako je element veći od korena drugog hipa, dodati ga u drugi hip.
3. ako je element veći od korena prvog hipa, a manji od korena drugog hipa onda u slučaju različitog broja elemenata hipova, dodati ga u hip sa manje elemenata. U slučaju istog broja elemenata dodati ga u proizvoljan hip.

Ukoliko je posle dodavanja elementa nastala situacija da jedan hip ima 2 elementa više od drugog hipa, onda iz većeg hipa ukloniti koren i dodati ga u manji hip.

Kolika je složenost dodavanja novog elementa u niz?

2 Heš tabele

Heš tabele se koriste kada je potrebno efikasno obavljati operacije dodavanja, pretrage i brisanja elemenata.

Dobra heš funkcija treba da transformiše skup ključeva (njih n ravnomerno u skup slučajnih lokacija iz skupa $\{0, ..., m-1\}$. Pri tome se smatra da broj lokacija nije mnogo veći (npr. duplo je veći) od broja ključeva koje unosimo (m nije mnogo veće od n). Heš tabele se koriste za implementaciju mapa u programskim jezicima (heš mape), pri radu sa bazama podataka, rečnicima, DNK sekvencama... Ovde ćemo koristiti 3 heš metode:

1. L - *lančanje* ili *odvojeno nizanje* (chaining): funkcijom h(x) = h1(x)
2. LP - *linearno popunjavanje* (linear-probing) funkcijom

$$\text{h(x,i)} = (\text{h1(x)}+\text{i}) \ \% \ \text{n, i=0..n-1}$$

3. DH - *dvostruko heširanje* (double hashing) funkcijom h1 kao heš funkcijom i funkcijom g kao sekundarnom

$$\text{h(x,i)} = (\text{h1(x)} + \text{i}\cdot\text{g(x)}) \ \% \ \text{n, i=0..n-1}$$

5. Skicirajte heš tabelu nakon smeštanja ključeva iz skupa 22, 1, 13, 11, 24, 33, 18, 42, 31 u zadatom poretku. Pretpostavite da je tabela veličine n=11, da je primarna heš funkcija h1(x) = x % 11, kao i da se za razrešavanje kolizija koristi lančanje, linearno popunjavanje i dvostruko heširanje sekundarnom heš funkcijom g(x) = 1+ x %10.

REŠENJE:

	L	LP	DH
0	22 ← 11 ← 33	22	22
1	1	1	1
2	13 ← 24	13	13
3		11	
4		24	11
5		33	18
6			31
7	18	18	24
8			33
9	42 ← 31	42	42
10		31	

U vrsti sa indeksom 3 u koloni LP je 11 jer je h(11,3)=(11%11+3) %11 = 3.
U vrsti sa indeksom 4 u koloni DH je 11 jer je h(11,2)=(11%11+2·g(11))%11=4.

Pretraživanje Izvodi se tako što se pomoću heš funkcije dobije moguća pozicija elementa. Kod lančanja se zatim u listi traži dati element. Kod linearnog popunjavanja se pretražuju redom sledeće pozicije dok element ne bude pronađen ili dok se ne dođe do prazne pozicije. Kod duplog heširanja se pomoću koraka (vrednosti druge heš funkcije) prolazi kroz tabelu dok se element ne nađe ili se ne dođe do prazne pozicije.

Brisanje Izvodi se tako što se prvo traži element, pa se briše ako je pronađen. Kod lančanja je to jednostavno, samo se obriše čvor. Kod druga dva pristupa se obično stavlja indikator da je element obrisan. Ovaj indikator dozvoljava da se na tu poziciju unese novi element, a pri pretrazi se smatra da je ta pozicija puna (kako bi ostali elementi i dalje bili dostupni). Kada broj ovakvih pozicija postane velike, može se izvršiti ponovno heširanje (*rehashing*), tj. prepisivanje elemenata u novu tabelu uz ponovno heširanje. Ponovno heširanje je potrebno izvršiti i kada tabela dostigne neki stepen popunjenosti, jer tada sve operacije postaju mnogo sporije.

Odabir heš funkcija Najteži zadatak kod heširanja je odabir dobrih heš funkcija. "Vrlo lako" je izabrati pogrešnu heš funkciju koja raspoređuje ključeve neravnomerno po tabeli ili čak sigurno ostavlja veliki broj slobodnih mesta da budu prazna. Ne postoji neko opšte pravilo kako konstruisati dobru heš funkciju, već sve zavisi od slučaja do slučaja (tj. najviše od očekivanih podataka sa kojima će se raditi). Kaže se da je dobar izbor heš funkcije više umetnost nego nauka. Najčešće se podaci preslikavaju u neki broj, koji se zatim funkcijom koja vraća ostatak po modulu nekog prostog broja.

Složenost Iako je vreme za sve operacije (unos, pretraga, brisanje) u najgorem slučaju $O(n)$, pri dobrom izboru heš funkcije se dobija da je prosečno vreme ovih operacija $O(1)$!!!

Odabir heširanja Ne postoji ni pravilo koje od 3 navedena heširanja koristiti. Često je teško proceniti koje od njih bi dalo dobre rezultate. Ako je memorijski prostor bitan onda je bolje izbegavati lančanje jer koristi dodatan memorijski prostor. Ako je brisanje česta operacija onda je uglavnom bolje koristiti lančanje jer se brisanje u tom slučaju efikasno implementira.

6. Skicirajte heš tabelu nakon smeštanja kljuceva 10, 22, 31, 4, 15, 28, 17, 88, 59 u zadatom poretku. Pretpostavite da je tabela veličine n=11, da je primarna heš funkcija h(x) = x %11, kao i da se za razrešavanje kolizija koristi dvostruko heširanje sekundarnom heš funkcijom g(x) = 1+ x % 10.

7. Pretpostavimo da se u heš tabeli umesto povezanih listi koriste binarna

stabla pretraživanja za razrešavanje kolizija odvojenim ulančavanjem. Koliko je vreme izvršavanja operacija umetanja i pretraživanja u najgorem slučaju? Kolika je ukoliko se koriste balansirana stabla?

8. Potrebno je smestiti podatke o korisnicima u Srbiji (ime, prezime, broj telefona) u elektronski telefonski imenik koji ima 10000 pozicija (pretpostavka je da neće biti više od 5000 brojeva). Koja je mana heširanja gde se informacije o korisniku smeštaju na poziciju određenu sa prve 4 cifre telefonskog broja?

Odgovor: problem je što će svi podaci smeštaju na pozicije koje za cifru hiljada imaju 0 (počinju npr. sa 011, 063, 064, itd). Sve pozicije koje počinju sa ostalim ciframa ostaće upražnjene pri pokušaju heširanja primarnom heš funkcijom. To znači da će vrlo često dolaziti do kolizija pri heširanju.

Slično ovom primeru, jedna loša heš funkcija koja smešta informacije o studentima u bazu bi bila ona koja bi koristila brojeve na pozicijima 5,6 i 7 u JMBG-u (jer su to cifre rođenja studenata i sve će biti u uskom segmentu.