SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Fibonaccijeva hrpa

Kristijan Verović
Voditelji: *Adrian Satja Kurdija i Marin Šilić*

Zagreb, svibanj 2023.

SADRŽAJ

1. Uvod

Hrpa ili gomila (eng. heap) je poznata struktura podataka koja se najčešće uvodno prikazuje kroz binarnu hrpu (eng. binary heap). U ovom radu će se prezentirati Fibonaccijeva hrpa [3] čije će ime biti uskoro objašnjeno, ona ima bolje asimptotske složenosti operacija $insert$, $decreaseKey$ i $union$. Operacija $union$ je samo spajanje dvaju različitih hrpa zajedno, dok je operacija $decreaseKey$ nešto manje poznatija te čak nije implementirana u C++ovom priority_queue. $decreaseKey$ smanjuje vrijednost postojećeg čvora na hrpi te nakon toga prestrukturira hrpu po potrebi. Na slici 1.1 se nalazi prikaz složenosti operacija po raznim vrstama hrpa, kod Fibonaccija će operacije $decreaseKey$ i $deleteMin$ biti amortizirane, maksimalne složenosti poziva jedne operacije $O(n)$.

Operation	find-min	delete-min	insert	decrease-key	meld
Binary[8]	$\Theta(1)$	$\Theta(\log n)$	$O(\log n)$	$O(\log n)$	$\Theta(n)$
Leftist	$\Theta(1)$	$\Theta(\log n)$	$\Theta(\log n)$	$O(\log n)$	$\Theta(\log n)$
Binomial[8][9]	$\Theta(1)$	$\Theta(\log n)$	$\Theta(1)$[a]	$\Theta(\log n)$	$O(\log n)$[b]
Fibonacci[8][2]	$\Theta(1)$	$O(\log n)$[a]	$\Theta(1)$	$\Theta(1)$[a]	$\Theta(1)$
Pairing[10]	$\Theta(1)$	$O(\log n)$[a]	$\Theta(1)$	$o(\log n)$[a][c]	$\Theta(1)$
Brodal[13][d]	$\Theta(1)$	$O(\log n)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
Rank-pairing[15]	$\Theta(1)$	$O(\log n)$[a]	$\Theta(1)$	$\Theta(1)$[a]	$\Theta(1)$
Strict Fibonacci[16]	$\Theta(1)$	$O(\log n)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
2–3 heap[17]	$O(\log n)$	$O(\log n)$[a]	$O(\log n)$[a]	$\Theta(1)$	?

Slika 1.1: Usporedba složenosti operacija različitih struktura hrpa, preuzeto s [1].

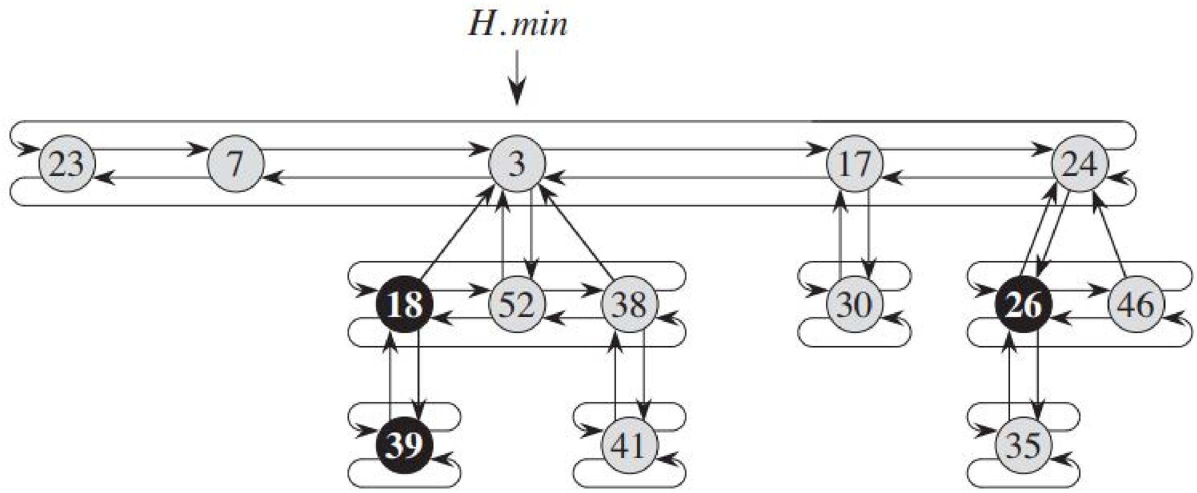
Spuštanje složenosti s $O(logn)$ na $O(1)$ barem teoretski pruža poboljšanja u primjenama od čega prvo pada na pamet Dijkstrin algoritam kojem se složenost pomoću Fibonaccijeve hrpe spušta sa $O((V + E)logV)$ na $O(E + VlogV)$. Može se proučavati i struktura sama za sebe i iz toga izvući potencijalne primjene, recimo slučaj kada je puno jeftinih $decreaseKey$ operacija i malo skupih $deleteMin$ operacija. Primjer takve situacije može se naći u društvenim mrežama gdje se neke objave žele staviti na "trending" te između dva objavljivanja trending sadržaja($deleteMax$) jako puno "lajkanja" sadržaja($increaseKey$). Teoretske prednosti se ne precrtavaju uvijek nužno u praktične prednosti, pogotovo kod Fibonaccijeve hrpe koja će imat visoku vremensku konstantu i veliku potrošnju memorije. U sklopu ovog rada implementirat ćemo te potom izmjeriti vremensko i memorijsko ponašanje oba opisana slučaja implementirana u usporedi s korištenjem obične binarne hrpe. Za kraj uvoda, neka je F_n n-ti Fibonaccijev broj i postavimo da su bazni brojevi Fibonaccijevog niza $F_0 = 1, F_1 = 1, F_2 = 2, F_3 = 3, F_4 = 5$... itd. Prefiks "Fibonaccijeva" u nazivu Fibonaccijeva hrpa dolazi od činjenice da podstablo Fibonaccijeve hrpe koje ima korijenski stupanj d mora imati minimalno F_d čvorova. Pokazati ćemo zašto je to tako, kao što ćemo pokazati druge činjenice vezane uz Fibonaccijevu hrpu te kako sve zajedno ostvaruje dobra asimptotska svojstva.

2. Fibonaccijeva hrpa

U ovom poglavlju će se opisati struktura i operacije te u potpunosti ili obrisno dokazati svojstva Fibonaccijeve hrpe. Iako će se opisivati struktura "min-heap", prebacivanje u "max-heap" je trivijalno.

2.1. Struktura

Slika 2.1 prikazuje strukturu neke Fibonaccijeve hrpe, vidimo da se radi o više hrpa spojenih zajedno od kojih jedna sadrži najmanji element[1] te imamo zaseban pokazivač na taj čvor. Sva djeca nekog čvora su povezana dvostruko povezanom listom (eng. doubly linked list), na isti način su povezani i korijenski čvorovi odnosno čvorovi na dubini nula. Neki čvorovi su označeni crnom bojom kao rezultat operacije $decreaseKey$ što će biti pojašnjeno kasnije.



Slika 2.1: Struktura Fibonaccijeve hrpe in medias res, uzeto iz [2].

Najbolje je prikazanu sliku nadopuniti implementacijskim detaljima prikazanu na

[1]Potencijalno je više čvorova s istom vrijednošću pa i minimalnom, naime hrpa nije matematički skup odnosno može sadržavati više istih vrijednosti.

kodu 2.1. Svrha varijable info je da znamo kojem djelu neke vanjskte strukture pripada cvor jer nam najčešće samo brojčana vrijednost val bez nekog entiteta da ga vežemo uz njega nije korisna. Varijabla marked je rezultat operacije $decreaseKey$ dok su varijable val i deg dovoljno objašnjene komentarima u kodu. Svaki čvor pamti svog roditelja (za čvorove na 0-toj razini roditelja nema pa je nullptr), dok su pokazivači za dvostruko povezanu listu left i right. Čvor pamti samo jedno svoje dijete jer ima sve mogućnosti kao i kad bi pamtio svu djecu, samo što bi to iziskivalo više memorije i operacija spajanja.

Kod 2.1: C++ kod za strukturu cvor

```
template<typename T>
struct cvor{
    T val; //(iliti key) numericka vrijednost(int, long
       long, double)
    int info;  //oznaka kojem cvoru u npr. Dijkstri
       pripada
    int deg;  //stupanj, broj djece
    bool marked; //oznacen da/ne

    cvor* parent,left,right,child; //pokazivaci

    cvor(T _val, int _info){ //konstruktor
        val = _val;
        info = _info;
        deg = 0;
        marked = false;
        parent = left = right = child = nullptr;
    }
};
```

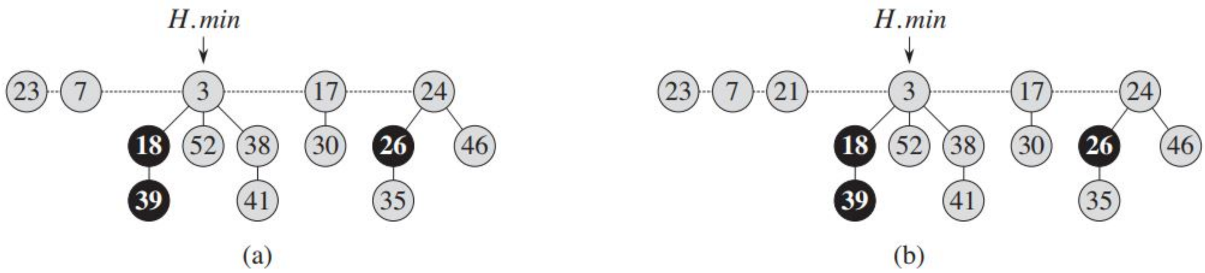
Ovdje bi imalo smisla provjeriti koliko memorije troši jedna instanca čvora. Recimo da je T tipa int, i da za bool moramo instacirati 4 umjesto jednog bajta zbog arhitekture računala odnosno riječi (eng. word) u memoriji. Za val,info,deg i marked. to iznosi 4*4=16 bajtova. Pretpostavimo razumno da radimo na 64-bitnom sustavu i da svaki pokazivač zauzima 8 bajtova, budući da ih je 4 to iznosi 4*8=32 bajtova. Dakle jedna instanca strukture cvor iznosi 16+32=48 bajtova. Ekvivalentna obična binarna hrpa bi trebala sadržavati samo varijable val i info odnosno 4+4=8 bajtova, dakle

Fibonaccijeva hrpa iziskuje čak 6 puta više memorije. Ovo bi moglo u praksi predstavljati problem usprkos boljih asimptotskih složenosti operacija Fibonaccijeve hrpe, stoga ćemo u poglavlju o performansama to i razmotriti.

2.2. Operacije

2.2.1. Insert

Na slici 2.2 je vizualno prikazano što se događa u operaciji insert. Pozivom operacije insert u korijensku razinu se doda novi čvor i potencijalno kaže da je taj čvor novi minimum ukoliko je stvarno najmanji. Budući da ćemo u korijenskoj listi imati samo pokazivač na $minimalniCvor$, spojit ćemo novi čvor odmah do njega te sa njegovim susjedom održavajući strukturu dvostruko povezane liste.



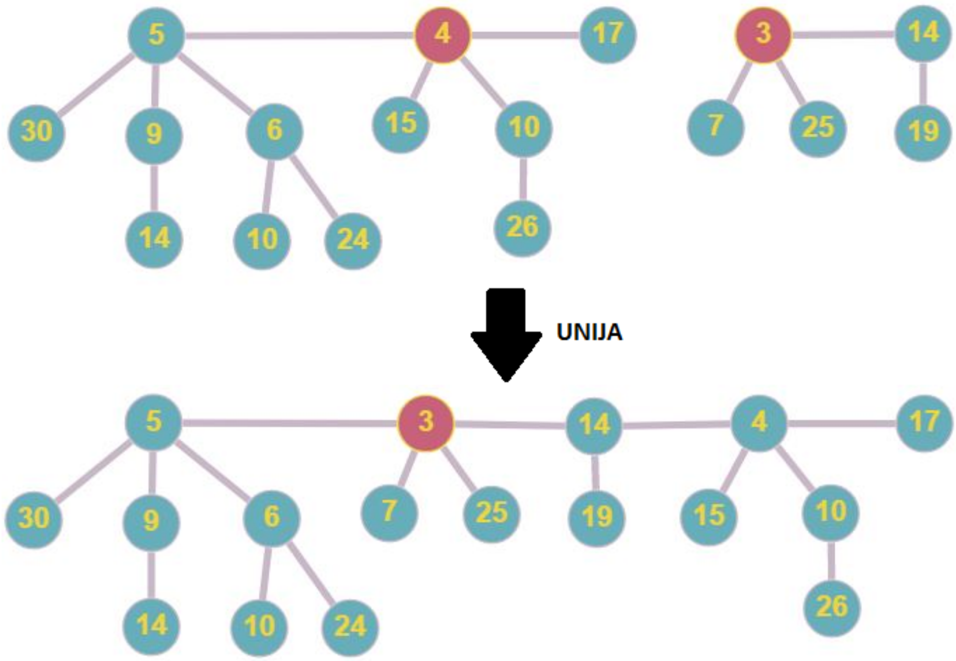
Slika 2.2: Prikaz operacije u kojoj u hrpu a) unosimo čvor s vrijednošću 21 što se vidi na slici b), uzeto iz [2].

Složenost ovog umetanja je naravno O(1), međutim može se dogoditi da nakon puno insert-ova dobijemo dugačak lanac. Operacija $deleteMin$ će u svom pozivu amortizirati složenost insert operacija jer će morati taj lanc sažeti na $logn$ čvorova što ćemo vidjeti u nastavku. I dalje je prinos operacije insert ukupnoj složenosti O(1) jer imamo jednu operaciju spajanja+jednu operaciju prestrukturiranja koju na sebe preuzima $deleteMin$, a znamo da je O(2) i dalje O(1).

2.2.2. Union

Operacija unije je praktički ista kao i operacija insert samo što sad potencijalno sudjeluje 4 čvora u prespajanju umjesto 3 kao kod inserta. Prikazana[2] ilustracija 2.3 prikazuje operaciju unije dviju hrpa.

[2]Crtano u `https://graphonline.ru/en/`

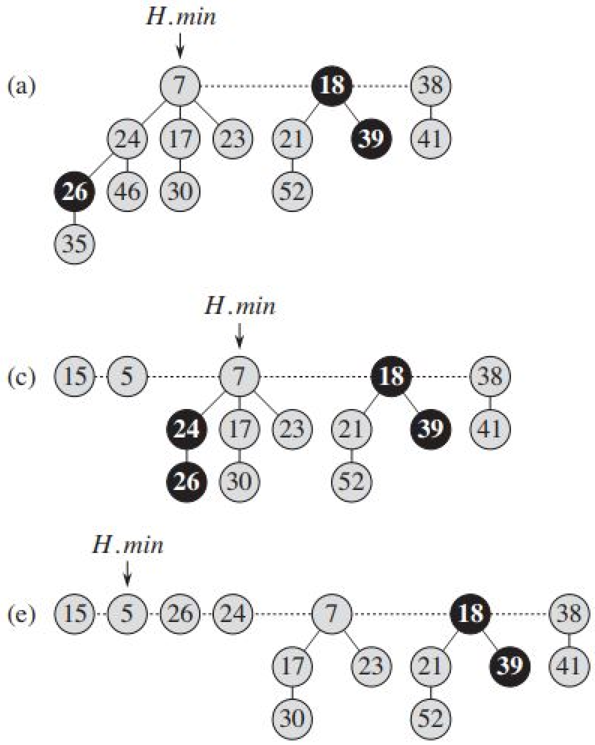


Slika 2.3: Unija dviju hrpa, crvenom bojom označeni minimumi.

Kao i kod inserta, operacija $deleteMin$ morat će amortizirati trošak spajanja dviju hrpa u slučaju da opet dobijemo neki lanac, iako ako se dobro razmisli taj se trošak može ponovno izvorno prepisati operaciji insert. Uzevši u obzir da je složenost spajanja dviju hrpa kod osnovne binarne hrpe jednaka $O(n)$, ovo je drastično ubrzanje. Nadalje ovakva jednostavna operacija spajanja prirodno omogućava paralelizam prilikom izgradnje binarne hrpe. Kako nebismo imali samo lanac, samo na kraju svakog paraleliziranog odsječka napravimo operaciju $deleteMin$ i onda ponovno insertamo taj isti minimum. Ista shema se može općenito primjeniti kad god mislimo da imamo predugačak lanac u korijenskoj razini.

2.2.3. DecreaseKey

Slika 2.4 prikazuje nesto



Slika 2.4: Dva poziva Fibonaccijeve hrpe (a) Početna Fibonaccijeva hrpa. (b) Vrijednost čvora 46 postaje 15, biva odrezan od roditelja i

2.2.4. DeleteMin

Operacija $deleteMin$ se može ostvariti kombinacijom već postojećih operacija kao $decreaseKey(-beskonacno, info)$ + $deleteMin$ što iznosi $O(1 + logn) = O(logn)$.

3. Performanse

Mjere se performanse na gustim i rijetkim grafovima, heap sporiji od stacka

Tablica 3.1: Usporedba prosječnih vremena tri algoritma u sekundama.

N	$2^n - 1$	Karatsuba	FFT
1000	0.0017	0.0021	0.0055
2000	0.006	0.0064	0.011
5000	0.039	0.0207	0.046
10000	0.16	0.051	0.093
20000	0.62	0.149	0.198
50000	4	0.66	0.44
100000	15.34	1.99	0.95
200000	-	6.06	1.79
350000	-	14.17	3.92
500000	-	27	3.88
1000000	-	-	8.42

4. Zaključak

Iako nema puno konkretnih implementacijsko tehnološke primjene gdje sigurno ima smisla koristiti Fibonaccijevu hrpu naspram drugih struktna je od teoretske važnosti. Spušta asimptotsku složenost Dijkstrinog i Primovog algoritma

5. Literatura

[1] URL https://en.wikipedia.org/wiki/Fibonacci_heap. Pristupljeno 21.4.2023.

[2] T. Cormen; C. Leiserson; R. Rivest; C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1990.

[3] M. Fredman; R. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal ofthe Association for Computing Machinery*, 34(3): 596–615, 1987.

6. Sažetak

Opisuje se i dokazuje struktura Fibonaccijeve hrpe. Navode se algoritmi Dijkstra i Prim čija je nadogradnja motivacija korištenja Fibonaccijeve hrpe. Pruža se implementacija u C++u te se mjeri performanse na gustim i rijetkim grafovima. Razmatraju se mogućnosti primjene Fibonaccijeve hrpe i na druge probleme.