LABORATORIJSKE VJEŽBE IZ ALGORITAMA I STRUKTURA PODATAKA

Vježba 5

Sortiranje

Heap sort, mjehurićasti sort

5.1. GOMILA (HEAP), HEAPSORT

Gomila

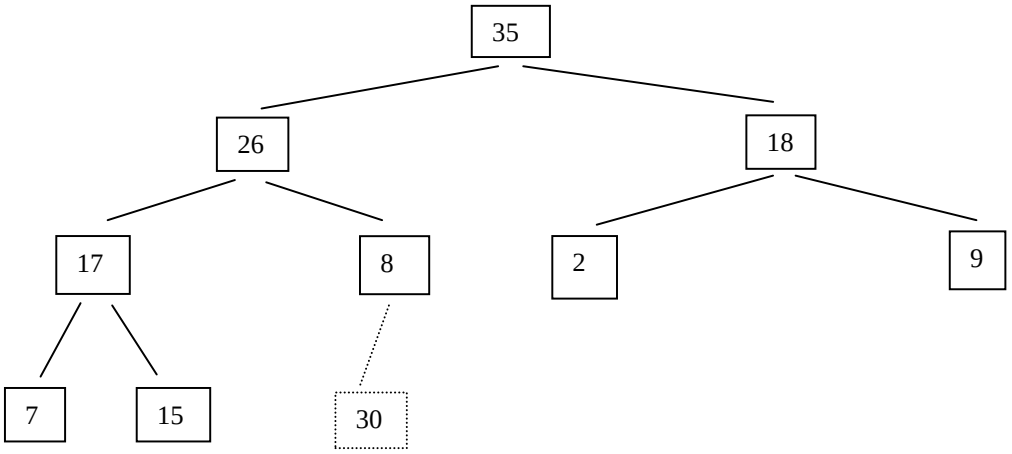
Neka je G potpuno binarno stablo sa n elemenata. Pretpostavit ćemo da je G spremljeno u memoriju računala kao polje STABLO upotrebom sekvencijalnog prikaza (pogledaj vježbu 4: Binarna stabla I). G se zove **gomila (heap, maxheap)** ako svaki čvor N od G ima slijedeće svojstvo *: Vrijednost u N veća je ili jednaka od vrijednosti svakog čvora djeteta od N*. Iz te definicije proizlazi da je vrijednost u čvoru gomile veća ili jednaka od vrijednosti svih čvorova koji su potomci od N. **Minheap** gomila se definira analogno : *Vrijednost u N je manja ili jednaka vrijednosti svakog čvora djeteta od N*.

Oblikovanje gomile

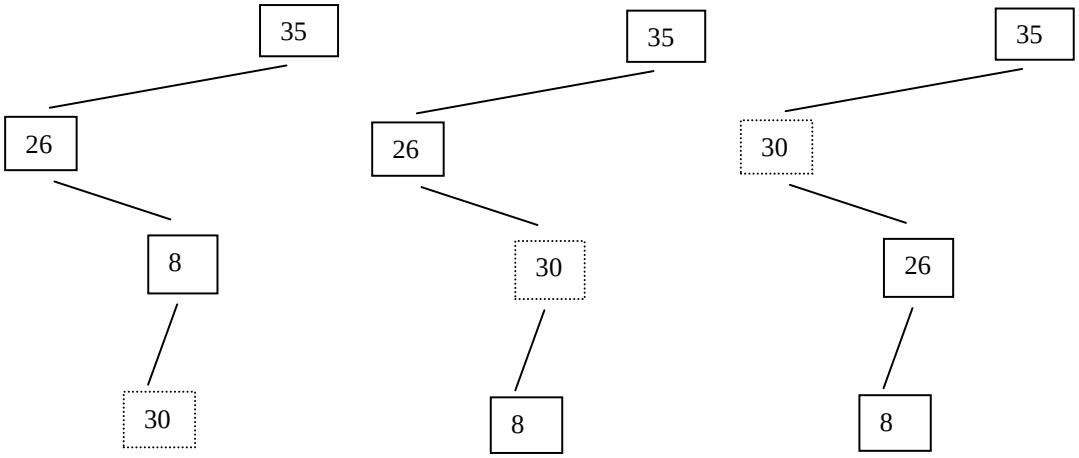
Najjednostavniji način je ubacivanjem jednog po jednog elementa u gomilu, čuvajući svojstvo gomile. Počinje se od prazne gomile. Pretostavimo da je G gomila sa N elemenata i da želimo ubaciti podatak STAVKA. Taj podatak ubacujemo u gomilu na slijedeći način:

(1) Prvo dodajemo podatak STAVKA na kraj gomile G tako da G ostane i dalje potpuno binarno stablo no ne neophodno i gomila (slika 5.1.)

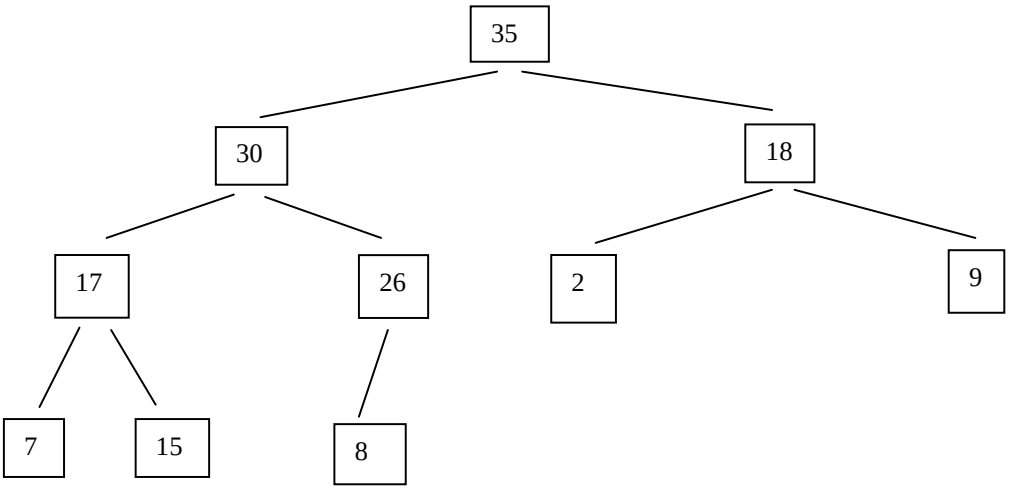
(2) Zatim podižemo podatak STAVKA (uspoređujemo ga sa njegovim roditeljem, praroditeljem, prapraroditeljem itd, dok ne postane manji ili jednak nekoj od tih vrijednosti, slika 5.2.) na "prikladno mjesto" tako da G ponovno tvori gomilu (slika 5.3).



Slika 5.1. Podatak 30 dodajemo na kraj gomile



Slika 5.2. Podizanje podatka po stablu

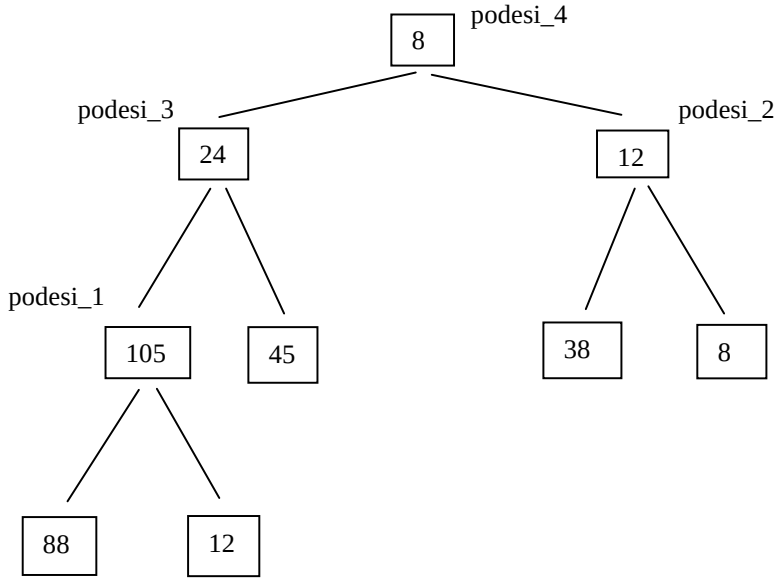


Slika 5.3. Nova gomila

Prosječno vrijeme izvođenja ovog algoritma je $O(n)$ a za najgori slučaj $O(n \log_2 n)$.

Za poboljšanje brzine obavljanja zadanih operacija stvoren je **poboljšani algoritam** koji kreče od roditelja zadnjeg čvora do korijena stabla i poziva funkciju **podesi**. Ta funkcija stvara gomilu čiji je korijen čvor s kojim je funkcija pozvana. Funkcija zahtjeva da lijevo i desno podstablo već oblikuju gomilu no to je osigurano redoslijedom pozivanja funkcije podesi. Slika 5.4. ilustrira pozive funkcije podesi. Prvo će se pozvati podesi_1, zatim podesi_2 pa podesi_3. Uočite da će prilikom poziva funkcije podesi pripadna podstabla oblikovati gomilu.

Kod poziva funkcije podesi, samo korijen može narušavati svojstvo gomile. Ta nepravilnost ispravlja se tako što se vrijednost korijena ubacuje na "prikladno mjesto", na sličan način kao što opisuje prethodni postupak ubacivanja novog elementa u gomilu, no ovdje umjesto da se penjemo po binarnom stabu, mi se spuštamo po njemu sve dok ne nađemo to mjesto.



Slika 5.4. Pozivi funkcije podesi

Vrijeme izvođenja za najgori slučaj poboljšanog algoritma je $O(n)$. Poboljšani algoritam traži da su svi elementi za stvaranje gomile već prisutni, dok obični algoritam može ubaciti novi element u gomilu bilo kada. Funkcije koje gomila treba brzo obaviti i radi kojih je napravljena ta struktura podataka su ubacivanje novih i brisanje najvećeg elementa iz skupa podataka. Brisanje najvećeg podatka se obavlja izbacivanjem korijena i pozivanjem funkcije podesi, a ubacivanje novih se radi funkcijom ubaci (iz običnog algoritma). Tako se postiže da se obje željene funkcije obavljaju u $O(\log_2 n)$ vremenu.

Heap sort

Pretpostavimo da se polje A sastoji od N elemenata koje želimo sortirati. Heap sort sastoji se iz tri faze :

(1) Od polja A napravi gomilu G

(2) Zamjeni korijen i zadnji elemenat stabla

(3) Podesi stablo tako da ponovno tvori gomilu ali bez zadnjeg elementa; ponovi (2) sve dok se ne dođe do zadnjeg čvora (korijena glavnog stabla).

Nakon oblikovanja gomile, najveći element je korijen stabla. Njega stavljamo na kraj stabla i oblikujemo novu gomilu od preostalih elemenata te opet korijen stavljamo na kraj. Ponavljamo li postupak do korijena stabla, ono će saržavati sortirane vrijednosti.

Vrijeme izvođenja heap sorta je $O(n \cdot \log_2 n)$.

5.2. MJEHURIĆASTI SORT (BUBBLE SORT)

Mjehiričasti sort jedan je od najjednostavnijih i najsporijih algoritama za sortiranje.

Pretpostavimo da u memoriji imamo polje brojeva A[1], A[2], ..., A[N]. Bubble sort algoritam radi na slijedeći način :

(1) Usporedi A[1] i A[2] i poredaj ih tako da bude A[1]<A[2]. Zatim usporedi A[2] i A[3] i poredaj ih tako da bude A[2]<A[3]. Zatim usporedi A[3] i A[4] i poredaj ih tako da bude A[3]<A[4]. Nastavi sa usporedbama sve do usporedbe A[N-1] sa A[N]. (u prvom koraku najveći elemenat uzastopnim zamjenama "kao mjehurić" putuje od početne pozicije na n-tu poziciju).

(2) Ponovi prvi korak bez zadnje usporedbe

(3) Ponovi prvi korak bez zadnje dvije usporedbe

....
....
....

(N-1) Usporedi A[1] sa A[2] i poredaj ih tako da bude A[1]<A[2]

Nakon n-1 koraka lista će biti uzlazno sortirana. Vrijeme izvođenja mjehurićastog sorta je $O(n^2)$.

Postoji i **poboljšanje bubble sorta** kod kojeg se koristi varijabla ZAMJENA koja nam govori da li je nakon koraka došlo do zamjene elemenata (ako je došlo zastavica se postavlja na vrijednost koja nam označava da je došlo do zamjene). Nakon tog

koraka ispituje se varijabla ZAMJENA i ako nije došlo do zamjene ne obavljamo daljnje korake budući da je polje već sortirano.

PRIPREMA ZA VJEŽBU :

Zadatak 1. Zadan je niz ulaznih podataka : 12, 5, 4, 10, 7, 8, 11.

a) od ulaznih podataka stvorite strukturu gomila (nacrtati stablo)
b) na tim podacima ilustrirajte kako radi uzlazni (od manjeg prema većem) heap sort (nacrtati niz stabala)

Zadatak 2. Napisati funkcije za sortiranje cjelobrojnog polja brojeva upotrebom heap sort i bubble sort algoritma. Prototip funkcija je :

```
void heap_sort(int *V, int n);
void bubble_sort(int *V, int n);

n je veličina polja
```

U glavnom programu generirati polje slučajnih brojeva i pozivati funkcije za sortiranje.

Funkciju pozivati više puta za polja različita sadržaja, mjeriti vrijeme sortiranja te nacrtati odgovarajući graf.

LABORATORIJSKA VJEŽBA 5 – ALGORITMI I STRUKTURE PODATAKA
Ime i prezime: _____
Broj indeksa: _____

Zadatak 1.
Nacrtati odgovarajući niz stabala za zadane čvorove:

Zadatak 2.
Za različite N izmjerite kolikotraje sortiranje:

VELIČINA POLJA N	**VR. TR. – BUBBLE SORT [ms]**	**VR. TR. – HEAP SORT [ms]**	**VR. TR.- MERGE SORT [ms] (za veću ocjenu)**

Nacrtajte ovisnot vreman izvođenja o broju podataka N