Student: Hata Madzak

*Smijer:*Teorijska kompjuterska nauka

Strukture podataka I algoritmi

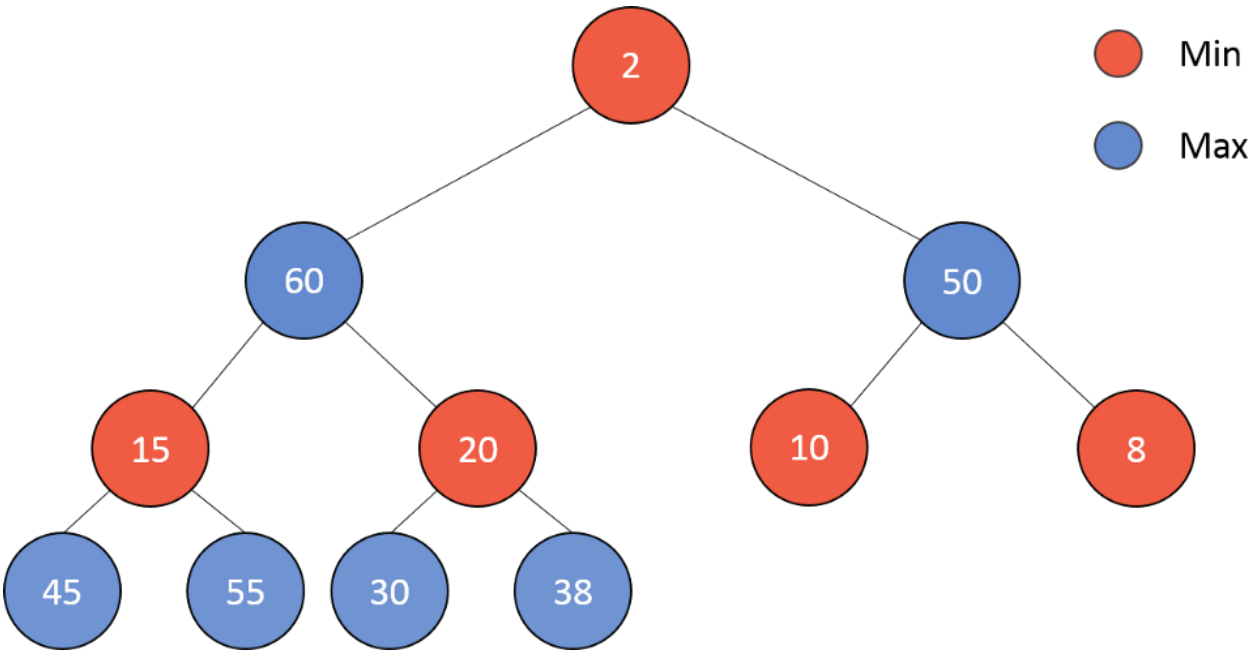
Tema projekta :

Potrebno je napraviti strukturu podataka koja na neki način predstavlja dvostrukiheap, u smislu da je moguće brisanje i najmanjeg i najvećeg elementa u vremenu O(logn). Taj heap implementirati preko niza kao na vježbama, a osobina ovog heapa je da za svaki elementvrijedi da ako je na parnoj dubini onda ja manji od svog roditelja a veći od roditelja svog roditelja, dok za elemente na neparnoj dubini vrijedi da su veći od svojih roditelja, a manji od roditelja svojih roditelja.

1. Osobine strukture :

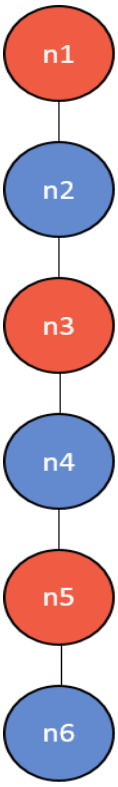
Heap je skoro potpuno binarno stablo, gdje samo zadnji nivo ne mora biti do kraja popunjen. Heap koji je implementiran u projektu je podjeljen na min nivoe I max nivoe. Ako se element nalazi na min nivou (odnosno na max nivou) , onda je on najmanji (najveci) u podstablu ciji je korijen taj element. Prvi element, tj. korijen je min nivo. Ako je neki nivo N1 na min nivou, onda su svi elementi s nizih min nivoa biti manji od elemanta iz N1. Kako je nivo veci, to su veci elementi. Za max nivoe vrijedi, sto je nivo veci, to su elementi tih novia manji. Sto znaci da ce najmanji element biti smjesten na prvi min nivo tj. U korijenu, a najveci element ce se uvijek nalaziti na prvom max nivou, tj. na nivou 1 ovog heapa.

Primjer:



2. Umetanje u heap

Ako uzmemo samo nivoe heapa generalno, smatramo da je n1 korijen , n2 svi elementi prvog nivoa itd. :



Slika 1. Dodavanje na max nivo

Slika 2.Dodavanje na min nivo

Dodavanje na max nivo:

Tada vrijedi da je n2 > n4 > n5 > n3 > n1. Kako je n6 na max nivou I ako je manji od n5, trebamo ga zamijeniti sa n5, ova zamjena nece izazvati nikakve promjene na max nivoima , redoslijed velicina n2>n4>n5 je idalje isti tj. Zadovoljava uvjete heapa. Moguce da se desila promjena na min nivoima kada se desila ova zamjena, mi zna se da je n5>n6, ali ne zna se za ove iznad da li je n6 manji od njih pa se treba pozvati funkcija UPHeapMin. Kada se svi elementi na min niovima poslazu, citav heap ce biti sredjen.

Ako je n6<n5, elementi min nivoa se ne moraju pomjerati, jer su tada elementi max nivoa veci od svakog elementa iz min nivoa, samo trebamo urediti max nivoe sto radi funkcija UPHeap_Max.

Dodavanje na min nivo:

Tada vrijedi n2>n4>n3>n1. Kako je n5 na min nivou I ako je veci od n4 (n5>n4) moramo ga zamijeniti, ali tada ostaje poredak na min nivoima jer ostaje da je n5>n3>n1 . Moguce da se promjena desila na max nivoima, znamo da je n5>n4, ali ne znamo da li je n5 vece I od n2 zbog toga pozivamo funkciju UPHeapMax.

Ako je n5<n4, tada nije potrebno vrsiti nikakve zamjene na max nivoima , samo je potrebno uspostaviti red na min nivoima sa funkcijom UPHeapMin.

Algoritam :

Korak 1: Ako X umecemo u niz , dodajem ga na kraj vektora. Pozivam UpHeap nad njim.

Korak 2: Izracunam nivo na kojem se nalazi.

Ako je nivo paran provjeravam da li je element X veci od svog roditelja,

-ako jeste zamjenim I pozivam UPHeap_Max nad njegovim roditeljem jer je on na Max nivou, jer je X bio na Min nivou.

-ako nije pozovem UPHeap_Min nad X .

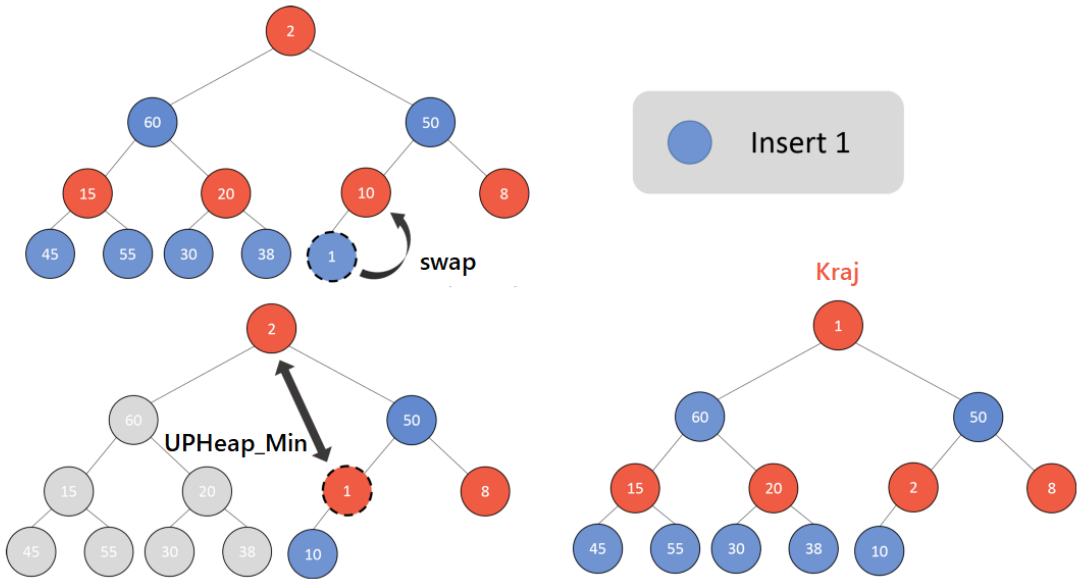
Ako je nivo neparan provjeravam da li je element X manji od svog roditelja,

-ako jeste zamjenim I pozivam UPHeap_Min nad njegovim roditeljem jer je on na Min nivou ,ako je X bio na Max nivou.

-ako nije pozovem UPHeap_Max nad X jer je on Min nivou.

Korak 3: Rekurzija prestaje kada dodjem do vrha heapa tj. Korijena.

Vrijeme izvrsavanja O(lg n) u najgorem slucaju prolazimo cijelom visinom.



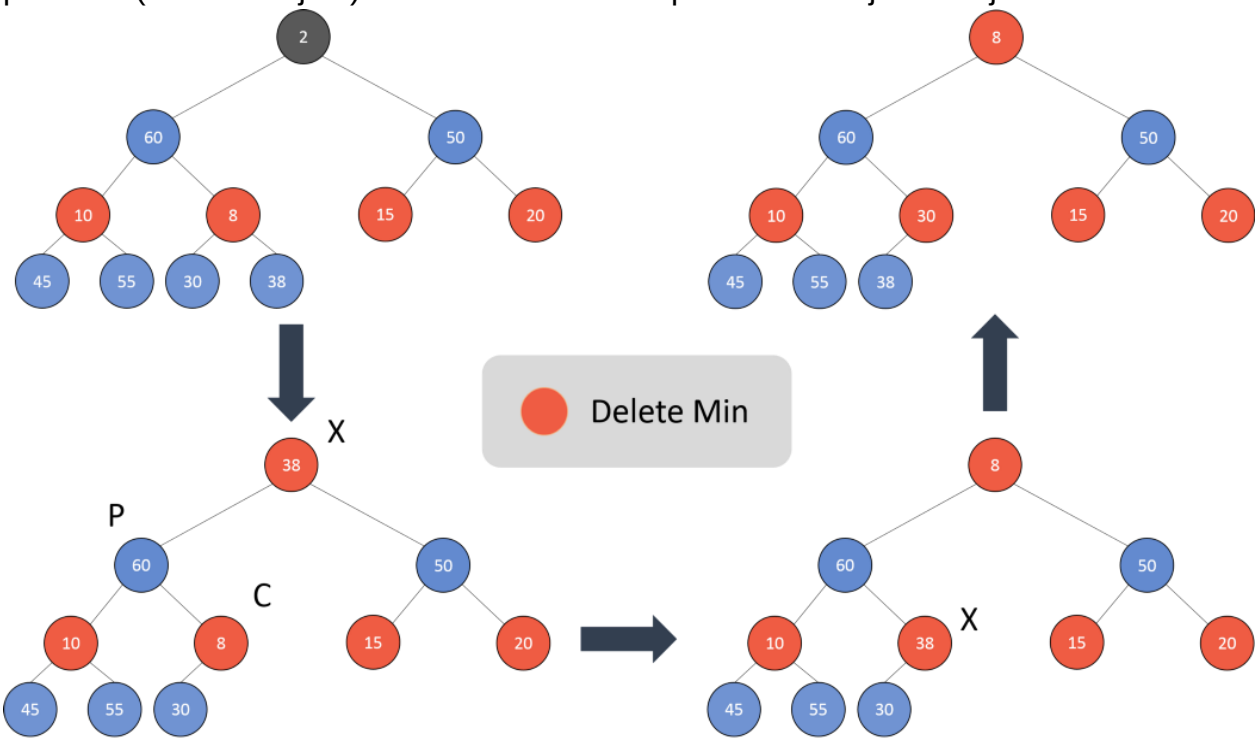
3. Brisanje najmanjeg elementa

Korak 1: Zamjenim korijen sa zadnjim elementom u heapu, i obrisem ga . Sada je narusena osobina heapa, pa pozivam funkciju Downheap za korijen,a kako je korijen u min noviu, za parametar bool Parnost saljem true.

Downheap(0,true);

Korak 2: void DownHeap (int index, bool Parnost) koristi se i za brisanje najmanjeg I za brisanje najveceg elementa u heapu, zbog toga se salje parametar Parnost. Kako je u ovom slucaju Parnost==True to znaci da se radi o min nivoima heapa, trazi se najmanji element od potomaka (Unuci I Djeca) u heapu, kada nadjemo takav zamjenimo ga I pozovemo funkciju DownHeap rekurzivno sada joj prosljedjujem index najmanjeg potomka a parnost ostaje ista, uslov prekida rekurzije kada je trenutni element najmanji od svih svojih potomaka, ili kad je element list tj. Nema roditelja.

Vrijeme izvrsavanja : O (lg n) , kako je heap skoro binarno stablo, tj. Svaki cvor ima dva djeteta osim zadnjeg nivoa koji nema djecu, to ce mu visina biti lg (n) baza je 2 jer je binarno. Kako se uvijek opredjelimo za jedan cvor I njegovo podstablo tj. Njegove potomke (unuka ilI dijete) maximalno sto cemo proci kroz niz jednako je visini stabla.



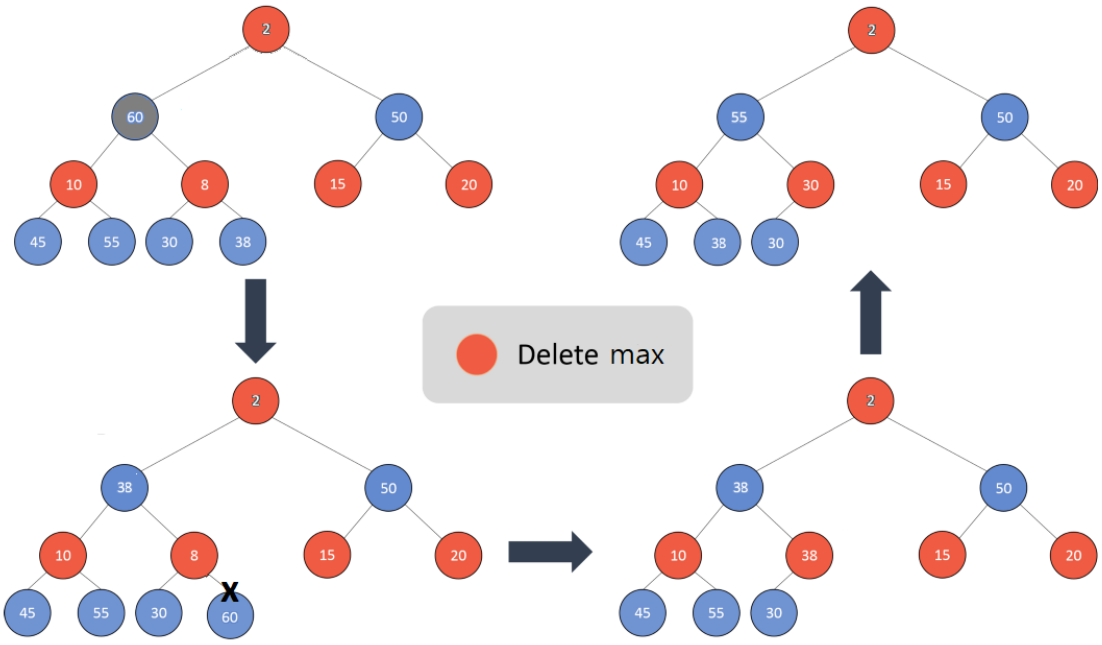
4. Brisanje najveceg elementa

Korak 1: Pronadjem index najveceg elementa u heapu (index_maximuma), to ce biti vece dijete od korijena. Zamjenim ga sa zadnjim elementom u heapu, kojeg obrisem. Sada je narusena osobina heapa, pa pozivam funkciju Downheap za index najveceg elementa,a kako je on sigurno na prvom nivou, za parametar bool Parnost saljem false.

Downheap(index_maximuma,false);

Korak 2: U ovom slucaju Parnost= false pa se radi o max nivoima heapa. Trazim index najveceg elementa od potomaka (unuci I djeca), kada nadjemo takav zamjenimo ga I pozovemo funkciju DownHeap kojoj sada prosljedjujem index najveceg potomka, a parnost ostaje ista. Uslov prekida je kada je element najveci od svojih potomaka, ili kada je element list.

Vrijeme izvrsavanja : O(lg n)



5. Pomocne funkcije

bool ImaLiDijete(int indeks);

Funkcije koje vracaju index:

int Djed (int indeks);

int Roditelj(int indeks);

int LijevoDijete(int indeks);

int DesnoDijete (int indeks);

int LLUnuk (int indeks);

int LDUnuk (int indeks);

int DLUnuk (int indeks);

int DDUnuk (int indeks);

int Min_unuk_dijete(int i);

int Max_unuk_dijete (int i);

int Index_maximalnog();

Funkcije za uspostavljanje trazenih osobina heapa:

void UPHeap_Min(int i);

void UPHeap_Max(int i);

void DownHeap(int index,bool parnost);

Vraca vrijednost :

int Daj_maximalni(); -vraca najveci u heapu , njegovu vrijednost

int Daj_minimalni(); -vraca najmanji u heapu, njegovu vrijednost