Datum: 2.11.2018.

Prioritetni red

U ovom materijalu je prikazan podatak apstraktnog tipa prioritetni red (*Priority Queue*, skraćeno - PR). PR se koristi kada postoje prioriteti po kojima treba izvršavati određene sekvence, na primjer, za pohranu procesa koji imaju različite prioritete izvršavanja.

Prioritetni red se može zamisliti kao kontejner pun elemenata poređenih prema prioritetu (podrazumijeva se da se elementi mogu porediti i tako rangirati prema prioritetu). Jedna korisna skala prioriteta može biti veličina (onda najveći element ima najviši prioritet, ili element koji ima najveću vrijednost ima najniži prioritet); druga može biti hitnost procesiranja. No, kako god definisali prioritet, kada uklonite element iz prioritetnog reda, uvijek dobijate element sa najvišim prioritetom.

Definisanje apstraktnog tipa podataka započinjemo definisanjem operacija koje mogu koristiti spoljni korisnici bez obaveze da prikažu strukture podataka ili programiraju operacije.

Definisanje PR

Prioritetni red (PR) je **konačna kolekcija** elemenata **istog tipa** za koje su definisane sljedeće operacije:

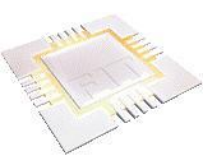
1. **iniciranje** prioritetnog reda, PR, da bude prazan,
2. utvrđivanje je li prioritetni red, PR, **prazan**,
3. utvrđivanje je li prioritetni red, PR, **pun**,
4. **dodavanje** novog elementa, X, u prioritetni red, PR, ako PR nije pun (PushPR),
5. ako je PR neprazan, **uklanjanje** elementa X sa najvišim prioritetom iz PR, (PopPR).

Iz definicije se jasno vidi da se u prioritetni red može umetnuti element sa bilo kojim prioritetom, ali se iz PR može preuzeti samo element sa najvišim prioritetom.

Reprezentacije prioritetnog reda

Postoji više različitih reprezentacija prioritetnog reda. Na primjer, jedna sortirana povezana lista u kojoj su čvorovi poređani po prioritetu u opadajućem poretku predstavlja prioritetni red za čekanje. Uklanjanje elementa iz takvog PR je jednostavno, samo treba da uklonimo prvi čvor povezane liste (sigurni smo da on ima najviši prioritet). Da bi umetnuli novi element u PR, treba da umetnemo novi čvor (prema prioritetu koji ima u redu za čekanje) u listu (uklanjanje je jeftino, a umetanje skupo).

Druga mogućnost je da se PR predstavi kao nesortirani niz. U ovom je slučaju umetanje elementa u PR jednostavno (samo se doda novi element na kraj niza), ali je komplikovanija procedura za preuzimanje elemenata iz PR (prvo pronađemo element najvišeg prioriteta, pa ga izbrišemo, a zatim prebacimo poslednji element niza u novonastalu "rupu"). (Sada je uklanjanje skupo, a umetanje jeftino.)



Postoji i efikasniji metod od ova dva, primjena binarnog stabla sa posebnim karakteristikama.

Primjene ATP prioritetnog reda - Sortiranje elemenata niza

Kada se napravi program za reprezentaciju prioritetnog reda, on se moze iskoristiti za sortiranje elemenata nekog niza. Efikasnost takvog programa zavisi od odabrane reprezentacije. Princip je sljedeći:

```
PRSort(SortNiz A)
        (deklarisati brojac i i prioritetni red PR tipa PRed)
        (Inicirati PR kao prazan red)
        for     (i ima vrjednosti od 1 do 10)       (umetnuti element iz A u PR)
        for     (i ima vrjednosti od 10 do 1)       (element iz PR smjestiti u A)
```

Implementacije prioritetnog reda

Sekvencijalna reprezentacija prioritetnog reda

U sekvencijalnoj reprezentaciji prioritetnog reda (PR) se za implementaciju apstraktog tipa podatka – PR koristi niz za sve funkcije kojima se ATP definiše. Dakle, treba napisati pet funkcija (InicirajPQ, PrazanPQ, PunPQ, PushPQ, PopPQ), pri čemu se pretpostavlja da je prioritetni red PQ definisan kao niz podataka traženog tipa, sa maksimalnom veličinom MaxVelPQ.

Svaki prioritetni red se predstavlja strukturnim nizom koji se sastoji od:

(a) broja elemenata koji su trenutno u prioritetnom redu, vrijednost člana VelPQ i
(b) niza elemenata prioritetnog reda, član PQ.

Kažemo da je prioritetni red **prazan** ako je vrijednost VelPQ jednaka **nuli**, a **pun** kada je vrijednost VelPQ jednaka **MaxVelPQ**.

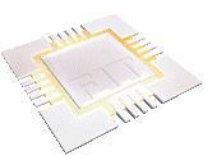
Da bi umetnuli element X u ne-pun prioritetni red PQ, prvo upišemo vrijednost X u niz PQ na poziciju određenu sa VelPQ, a zatim uvećamo član VelPQ za jedan.

Da bi uzeli element X sa najvećim prioritetom iz nepraznog prioritetnog reda PQ:

- postave se pomoćne vrijednosti najveći element u prioritetnom redu, za vrijednost najvećeg elementa MaxV=NizPQ[0] i za indeks najvećeg elementa MaxI=0 (u programu se pretpostavi da je prvi element niza najveći)
- u for petlji se redom upoređuju vrijednosti elemenata niza NizPQ sa MaxV;
    - ukoliko je NizPQ[i] > MaxV, onda se njegova vrijednost dodjeli u MaxV, a njegov indeks u MaxI

(kada se for petlja završi, vrijednost najvećeg elementa niza je sačuvana u MaxV, a njegov indeks u MaxI)

- umanji se broj elemenata u prioritetnom redu VelPQ za 1
- posljednji element u prioritetnom redu se prebaci na poziciju MaxI, jer je niz sekvencijalna struktura i ne može „imati rupe"
- vrati se vrijednost MaxV.



Povezana reprezentacije prioritetnog reda

U povezanoj reprezentacji, PR se predstavlja kao sortirana povezana lista u kojoj su čvorovi poređani po prioritetu u opadajućem poretku.

Uklanjanje elementa iz takvog PR je jednostavno, samo se treba ukloniti prvi čvor povezane liste (pošto je lista sortirana, to je element sa najvišim prioritetom).

Da bi se umetnuo novi element u PR, treba se umetnuti novi čvor u listu na mjesto koje je određeno prema prioritetu koji on ima u redu za čekanje.

Za umetanje novog elementa, X, u PR, prema redosljedu prioriteta, treba razmotriti tri slučaja.

- Kada je PR prazan, onda ga treba zamjeniti novom listom koja ima samo jedan čvor, N, kome je X vrijednost Info polja, a NULL vrijednost pokazivača.
- Kada element koji se umeće, X, ima prioritet veći ili jednak od prioriteta prvog čvora liste PRLista. Tada treba umetnuti novi prvi čvor u PRLista, da mu je X u Info polju.
- Kada je prioritet X manji od prioriteta prvog čvora liste PRLista. Tada se X rekurzivno umeće u rep (tail) liste PRLista (gdje je rep povezana lista u kojoj se nalaze svi čvorovi liste PRLista osim prvog).

Svi kodovi su dati u materijalima za vježbe.

Nelinearna reprezentacija prioritetnog reda

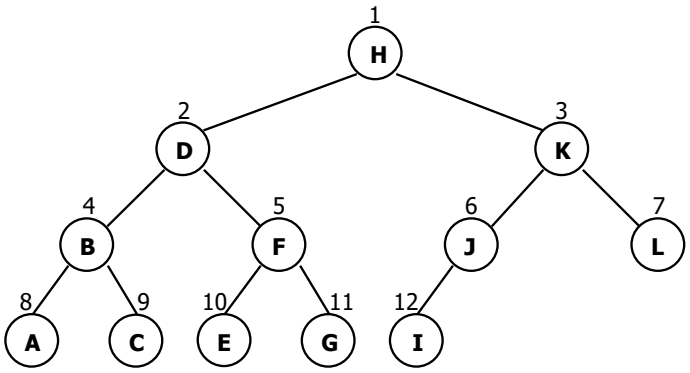
Binarno stablo je ono u kojem svaki čvor ima tačno dva djeteta. Dozvoljeno je da jedno ili oba djeteta budu prazni.

Za binarno stablo se kaže da je **kompletno** ako

- ima listove na jednom nivou ili na dva susjedna nivoa, pri čemu svi vrhovi osim onih u posljednjem nivou imaju svojstvo kompletnosti,
- listovi na najnižem nivou su grupisani lijevo, koliko je god to moguće.

Sekvencijalna reprezentacija binarnog stabla

Kompletno binarno stablo (Slika 1) se može jednostavno predstaviti pomoću niza. Vrhovi stabla se numerišu od korijena naniže (od 1 do 12), po nivoima s lijeva nadesno.



Slika 1: Kompletno binarno stablo sa numerisanim čvorovima

Dalje se kreira niz A[0:12] sa sadržajem čvorova stabla po odgovarajućem redosljedu:

A:		H	D	K	B	F	J	L	A	C	E	G	I
	0	1	2	3	4	5	6	7	8	9	10	11	12

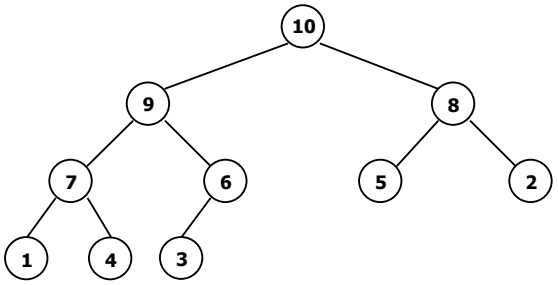
Za prepoznavanje elemenata binarnog stabla u sekvencijalnoj reprezentaciji se koriste aritmetičke operacije nad indeksima niza A, prikazane u Tabeli 1:

Za pronalaženje	koristi se	pod uslovom
Lijevog djeteta A[i]	A[2*i]	2*i ≤ n
Desnog djeteta A[i]	A[2*i+1]	2*i +1≤ n
Roditelja A[i]	A[i/2]	i>1
Korijena	A[1]	A je neprazan
Je li A[i] list	True	2*i >n

Tabela 1

Termin *heap* se upotrebljava i za kompletno binarno stablo sa vrijednostima u čvorovima uređenim na određeni način.

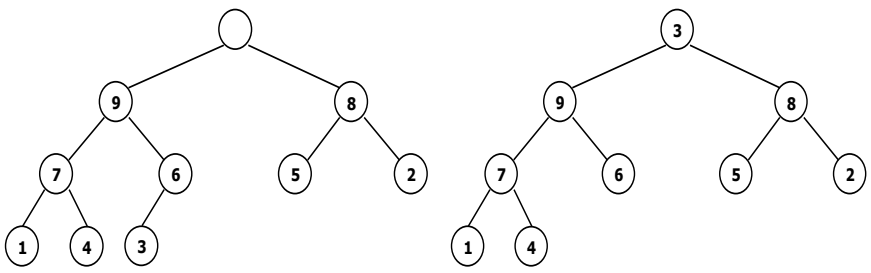
Heap je kompletno binarno stablo čije se vrijednosti čuvaju u čvorovima sa svojstvom da je svako djete u relaciji sa roditeljem (na primjer, nijedno djete ne nosi vrijednost veću od roditelja). Na Slici 2 je prikazan primjer heap-a.

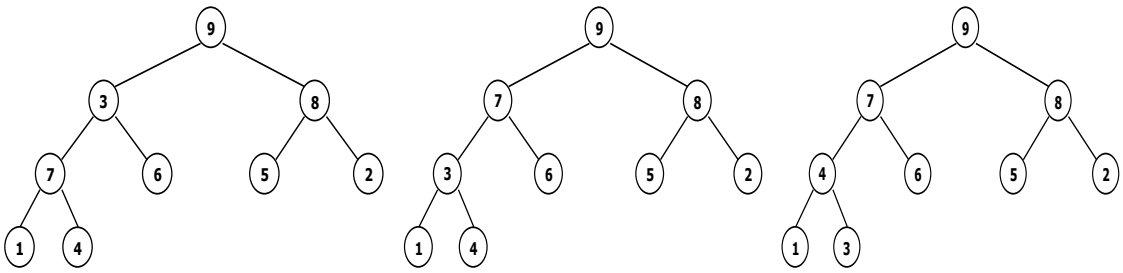


Slika 2: Jedan primjer *heap*-a

Ako se koristi heap za reprezentaciju prioritetnog reda za čekanje, onda je jednostavno odrediti element sa najvišim prioritetom – on je u korjenu. Ipak, ako se ukloni vrijednost iz korijena, mora se resruktuirati stablo da ponovo postane heap. Restruktuiranje se postiže na slijedeći način:

- prvo se obriše vrijednost iz najdesnijeg lista u najnižem redu,
- zatim se ta vrijednost smjesti u korijen,
- onda se ponovo uspostavlja svojstvo heap-a među preostalim čvorovima tako što se počevši od korijena, razmjenjuje vrijednosti sa većom vrijednošću u djetetu, sve dok su takve razmjene moguće (Slika 3).





Slika 3: Proces restruktuiranja heap-a