ALGORITMI I STRUKTURE PODATAKA

STUDIJSKI PROGRAMI:

SOFTVERSKO INŽENJERSTVO, RAČUNARSKA TEHNIKA, INFORMATIKA I MATEMATIKA

NASTAVNIK: DOC. DR ULFETA MAROVAC, UMAROVAC@NP.AC.RS

REDOVI

DEFINICIJA REDA

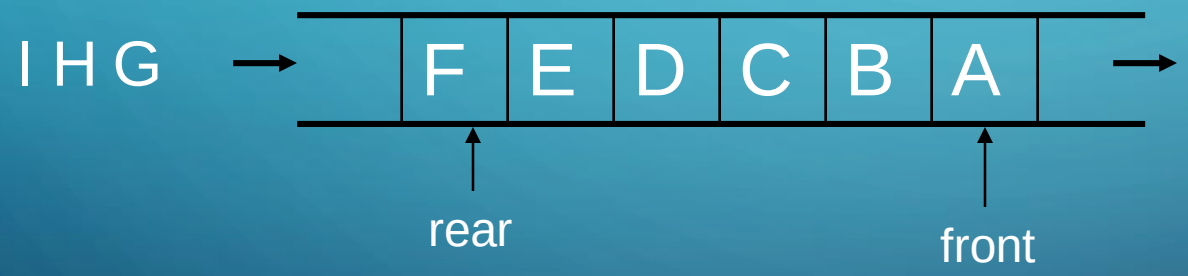
- Red(queue) je struktura koja predstavlja jednu vrstu linearne liste sa posebnim načinom umetanja i brisanja elemenata
- Red je stuktura sa dva pristupna kraja, pri čemu se umetanje vrši na jednom kraju a brisanje na drugom kraju reda
- Mesto na kojem se nalazi prvi element reda se naziva **čelo** (front) reda i predstavlja pristupni kraj sa kojeg se uklanja element iz reda
- Mesto gde se nalazi poslednji element se naziva **začeljem** (rear) reda i iza njega se stavlja novi elemenat pri umetanju

DEFINICIJA REDA

- Red ima dva pristupna kraja
- Linearna struktura koja ne dozvoljava promenu poretka elmenata
- Sledi da se elementi iz reda uklanjaju u istom poretku kojem su i umetnuti
- Elemenat koji je najranije umetnut biva ranije i uklonjen
- Ovakva disciplina pristupa se naziva FIFO (First In First Out)- prvi unutra, prvi napolje
- FIFO disciplina predstavlja tipičnu disciplinu opsluživanja po redosledu dolaska (FCFS)

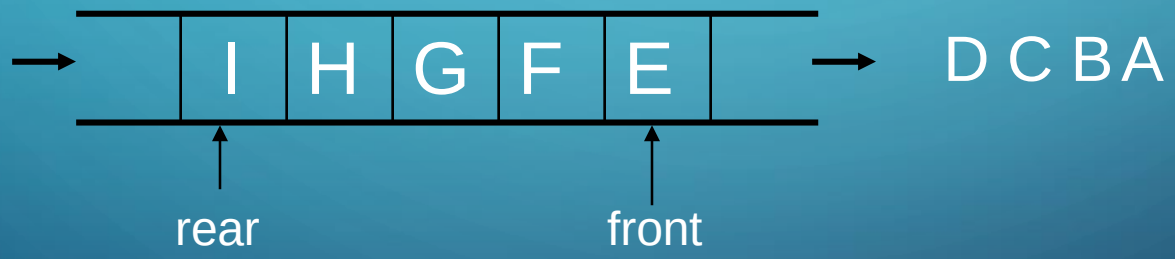
DEFINICIJA REDA

- Na slici je prikazan red sa 6 elemenata gde se A nalazi na čelu(element koji je prvi ušao) a F na začelju kao poslednji umetnuti element



DEFINICIJA REDA

- Četiri operacije brisanje uklanjaju elemente u poretku ABCD, a tri operacije umetanja uvode nove elemente u red u poretku GHI



OSOBINE REDA

- Red nema osobinu, kao stek, da par uzastopnih operacija umetanja i brisanja istog elementa ostavlja isto stanje steka
- U slučaju reda, par uzastopnih operacija umetanja i brisanja istog elementa ostavlja isto stanje reda samo ako je red na početku bio prazan
- Primere reda srećemo u realnom svetu: čekanje pred šalterom, na semaforu, naplatnoj rampi
- Redove srećemo i u računarskom okruženju: sistemi sa raspodeljenim vremenom, kontrola pristupa zajedničkim resursima (magistrala, memorija, U/I uređaji), itd.

OSOBINE REDA

- Red je **dinamička** struktura jer broj elemenata u njemu varira saglasno operacijama umetanja i brisanja elemenata
- Pri umetanju i brisanju elemenata, čelo i začelje se pomeraju
- Kao i kod steka, elementi reda su obično istog tipa, pa je on najčešće **homogena** struktura

SEKVENCIJALNA REPREZENTACIJA REDA

- Pošto je red linearna stuktura, nameće se sekvencijalna predstava u vidu vektora
- Vektorska implementacija zahteva homogenost reda a najčešće nameće i ograničenje na fiksan broj elemenata
- Na početku se čelo i začelje vežu za početak vektora, a kasnije se pomeraju ka višim adresama saglasno operacijama brisanja i umetanja u redu
- Jednako je ispravno i da se čelo i začelje vezu za kraj vektora, a kasnije se pomeraju ka nižim adresama
- Da bi se pratila pozicija čela i začelja treba održavati dva pokazivača *front* i *rear* koji ukazuju na ova dva pristupna kraja

SEKVENCIJALNA REPREZENTACIJA REDA

- Neka je red predstavljen vektorom Q[1:N]
- Celobrojni pokazivači front[Q] i rear[Q] predstavljaju pozicije čela i začelja reda respektivno
- Pokazivači front[Q] i rear[Q] se pomeraju prema višim adresama
- Red se sastoji od elemenata *Q[front[Q]], Q[front[Q+1]],…, Q[rear[Q-1]], Q[rear[Q]]*
- Opseg front…rear predstavlja niz uzastopnih adresa i važi *front<rear*

SEKVENCIJALNA REPREZENTACIJA REDA

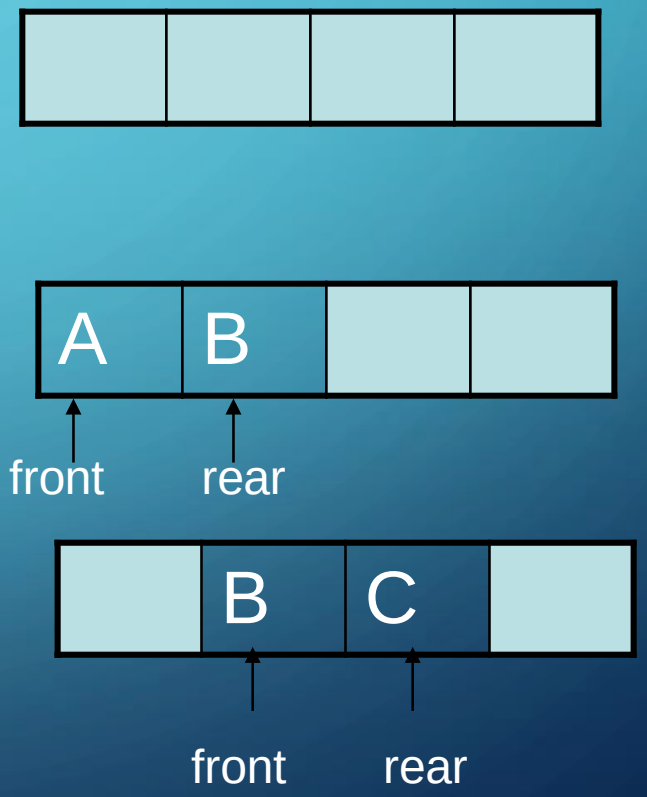
- Konačna dužina kojim se simulira red može da izazove određene implementacione probleme sa prekoračenjem kapaciteta pri umetanju elemenata
- Koncept reda sam nameće problem prekoračenja kapaciteta, kada se red brže puni nego što se prazni

SEKVENCIJALNA REPREZENTACIJA REDA

- Konačna dužina kojim se simulira red može da izazove određene implementacione probelme sa prekoračenjem kapaciteta pri umetanju elemenata
- Koncept reda sam nameće problem prekoračenja kapaciteta, kada se red brže puni nego što se prazni

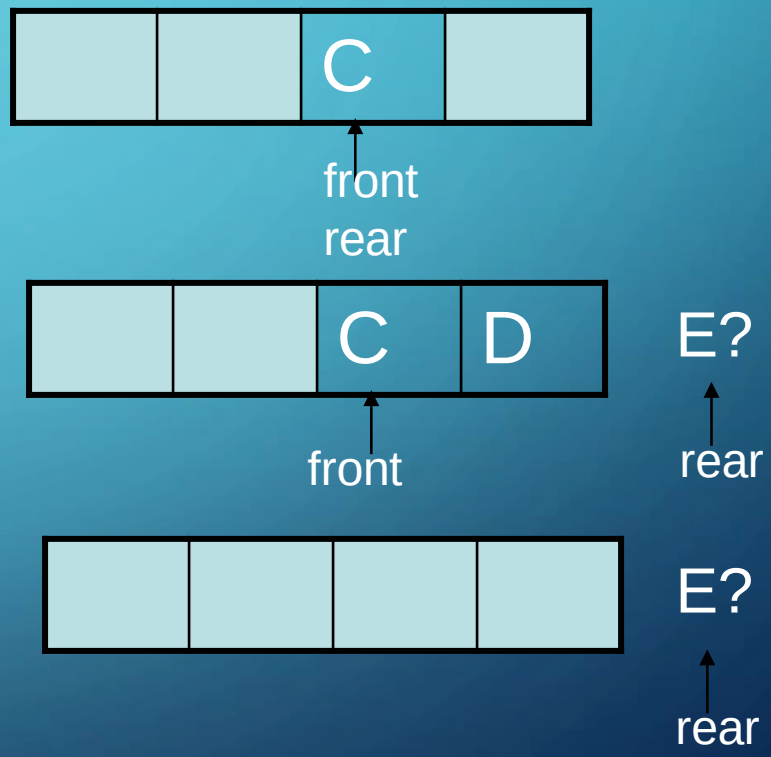
SEKVENCIJALNA REPREZENTACIJA REDA

- Red je predstavljen vektorom Q od 4 elementa
- Javljaju se dva umetanja(A pa B)
- Zatim brisanje elementa A i ubacivanje elementa C
- Čelo se nalazi Q[2] a začelje se nalazi u Q[3]



SEKVENCIJALNA REPREZENTACIJA REDA

- Brisanje (B) i umetanje (D)
- Kad treba da ubacimo elemenat E začelje se nalazi u Q[4], pa nije moguće ubaciti novi element iako postoje 2 prazna mesta Q[1] i Q[2].
- Ako je red potpuno ispražnjen operacijama brisanja C i D, novi element se ipak ne može umetnuti jer je začelje prešlo preko gornje granice niza



SEKVENCIJALNA REPREZENTACIJA REDA

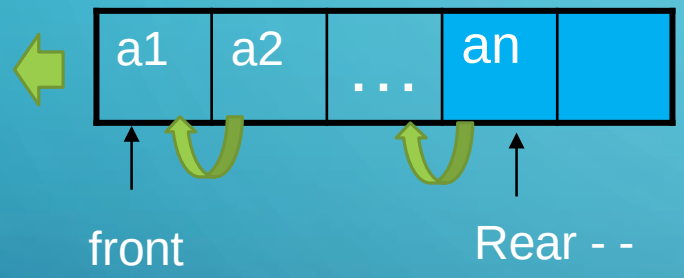
- Prethodna situacija je neprihvatljiva pa treba težiti takvoj implementaciji kod koje se prekoračenje dešava tek pri umetanju u red čiji je broj elemenata jednak dužini vektora
- Jedno rešenje bi bilo da se posle svake operacije brisanja svi elementi reda, od prvog do poslednjeg, pomere za jedno mesto nadole, tako da je čelo uvek vezano za prvi elemenat vektora Q
- U ovom slučaju pokazivač *front* je nepotreban jer je on implicitno jednak indeksu prvog elementa

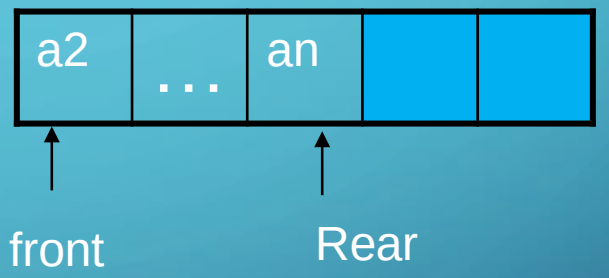
SEKVENCIJALNA REPREZENTACIJA REDA

- Jednostavnost ovakvog metoda plaća se njegovom neefikasnošću
- Za redove sa velikim brojem elemenata ova operacija pomeranja je veoma vremenski zahtevna
- Efikasnost implementacije sa pomeranjem se može popraviti ako se pomeranje radi tek *kad zaglavlje dođe do gornje granice vektora*
- U tom trenutku se pomeranje vrši tako što se čelo dovede u prvi element vektora(pomeri se za front-1 mesta) a zatim se i ostali elementi presele na odgovarajuća mesta za isti pomeraj
- Gornji primer(posle umetanja elementa D, začelje je došlo na gornju granicu. Tada bi se element C prebacio na Q[1] a elemenat D na Q[2] poziciju)

KOLIKA JE PROSEČNA SLOŽENOST IZBACIVANJA ELEMENTA IZ REDA U OVOM SLUČAJU?

- Za svako izbacivanje elementa iz reda moramo izvršiti rear-front pomeranja



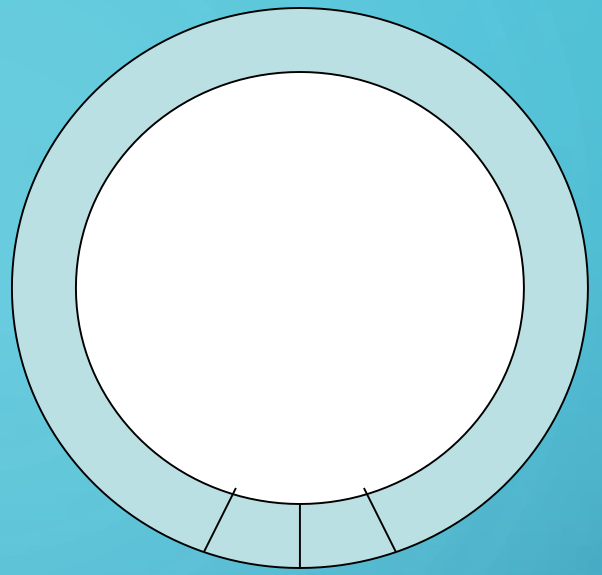


- Pa je prosečna složenost algoritma $\frac{1}{n}\sum_{k=1}^{n-1} k = \frac{1}{n}\frac{n(n-1)}{2} = \frac{n-1}{2}$

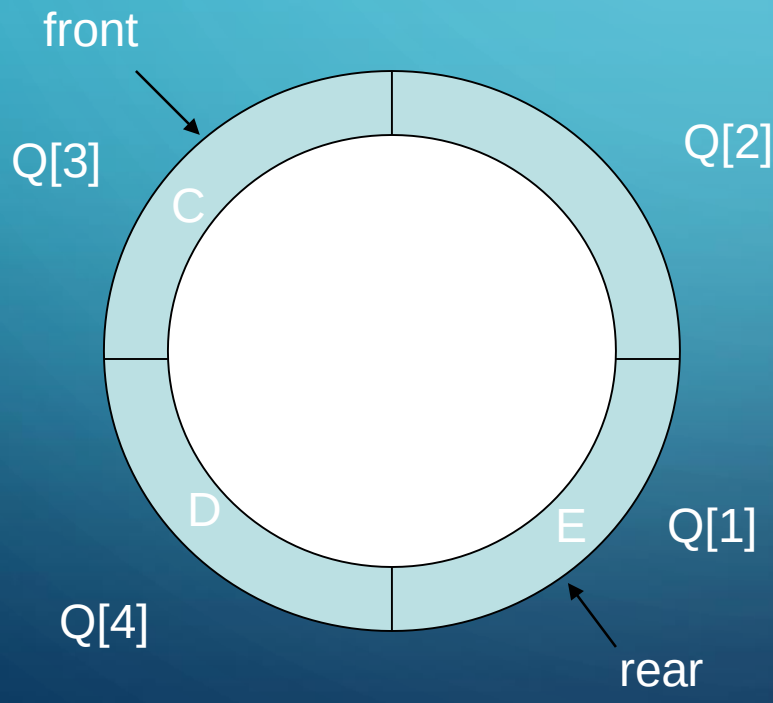
  gde k= rear-front

IMPLEMENTACIJA REDA U VIDU KRUŽNOG BAFERA

- Kružni bafer je vektor kod kojeg se prvi element smatra logičkim sledbenikom poslednjeg elementa, iako su oni fizički razdvojeni
- Logička struktura je takva da Q[1] logički sledi Q[n]
- Sada je umetanje u red može izvršiti i kada je poslednji element vektora popunjen, pod uslovom da je prvi element reda prazan
- Umetanje elementa E, iz predhodnog primera, je moguće izvršiti bez ikakvog pomeranja elemenata što je mnogo efikasnije u odnosu na predhodno rešenje



Implementacija reda u vidu kružnog bafera
Prazan red, front=rear=0



Rešenje problema iz predhodnog primera- umetanje elementa E kada je začelje na kraju reda

OPERACIJA KREIRANJA REDA

- Operacija kreiranja reda maksimalne veličine n.
- Procedura INT-QUEUE(Q,n) rezerviše prostor za niz date veličine ALLOCATE(Q[1:n]).

```
INT-QUEUE(Q,n)
ALLOCATE(Q[1:n])
front[Q]=rear[Q]=0
```

- Pošto je red na početku prazan, pokazivači na čelo i začelje reda se inicijalizuju na nulu(nepostojeći element)

OPERACIJA UMETANJA ELEMENTA U RED (KRUZNI BAFER)

- INSERT(Q, x) umeće novi element sa vrednošću x u red Q
- Prvo se pokazivač na začelje pomeri na sledeći element
- Ukoliko je posle toga začelje sustiglo čelo(front=rear), red je potpuno popunjen, pa se ne može izvršiti
- U sutrotnom, sledeće mesto na koje ukazuje ažurirani pokazivač rear je slobodno i podatak se upisuje u njega
- Posebno se mora voditi računa o slučaju ubacivanja u prazan red

•Ovo je specifičan slučaj kada se pokazivač čela mora postaviti na 1

- Operacija umetanja se nekad nalazi pod imenom

ENQUEUE

```
INSERT(Q, x)
rear[Q]= rear[Q] mod n+1
if(front[Q]= rear[Q]) then
        ERROR(Overflow)  else
         Q[rear[Q]]=X
         if (front[Q]=0) then
               front[Q]=1
         end
_if  end_if
```

OPERACIJE BRISANJA ELEMENTA IZ REDA

- Operacija uklanjanja elementa iz reda prvo proverava da li je red prazan
- Ako red nije prazan, očitava se i vraća vrednost elementa sa čela reda
- Ako je ovo bio jedini element u redu front i rear se postavljaju na 0
- Ako nije bio jedini element, front sada pokazuje na logički naredni element
- Nekad se ova operacija javlja pod imenom DEQUEUE

```
DELETE(Q)
If (front[Q]=0) then return
    underflow
else
    x=Q[FRONT[Q]]
    if (front[Q]= rear[Q]) then
        front[Q]=rear[Q]=0
    Else
    front[Q]= front[Q] mod n + 1
     end_if
     return x
    end_if
```

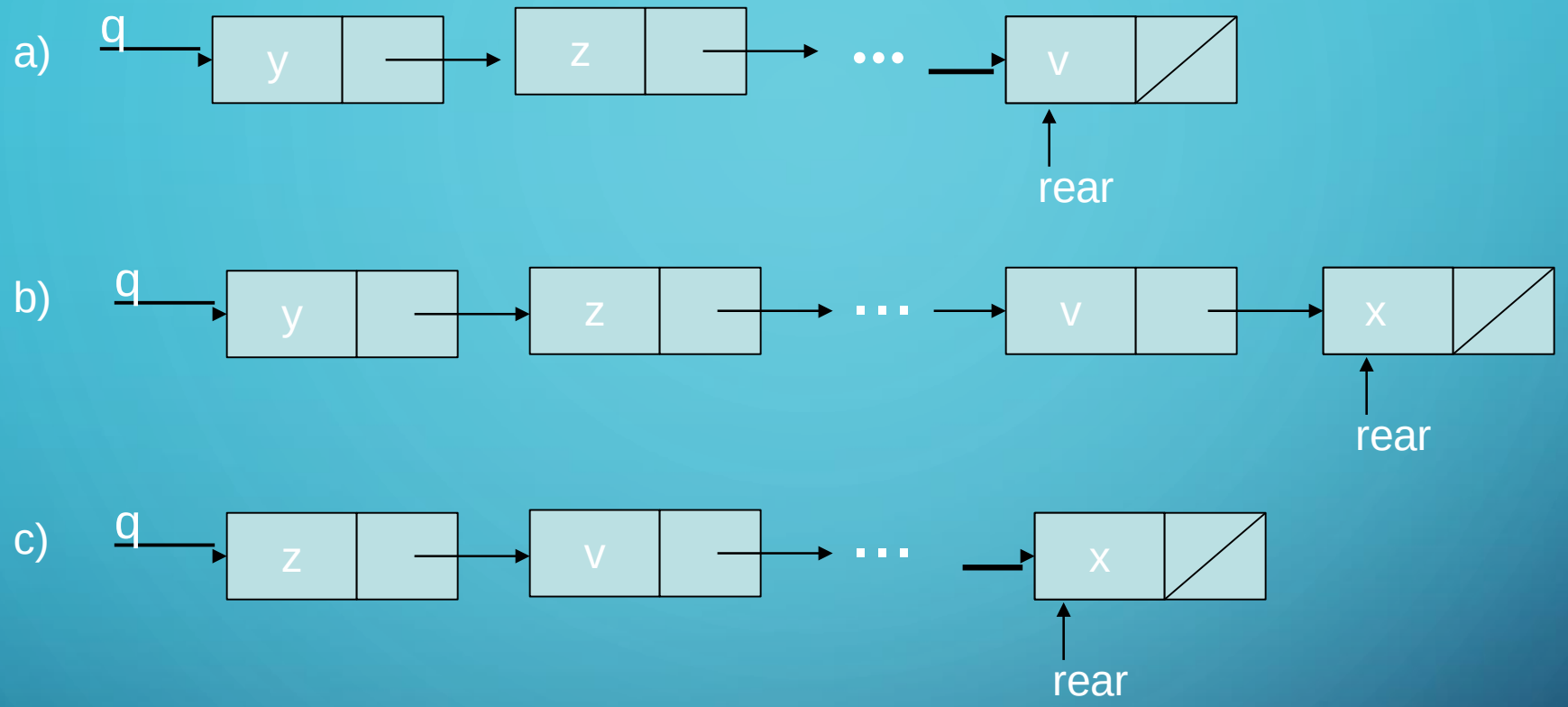
ISPITIVANJE DA LI JE RED PRAZAN OČITAVANJE ELEMENTA SA ČELA REDA

```
QUEUE-EMPTY(Q)
If (front[Q]=0) then
   return true
else
   return false
end_if
```

```
FRONT(Q)
If (front[Q]=0) then
   return underflow
else
   return Q[front[Q]]
end_if
```

ULANČANA REPREZENTACIJA REDA

- Kao i kod steka, ulančana reprezentacija reda daje određene pogodnosti
- Prvi čvor liste prestavlja čelo reda, pa spoljašnji pokazivač q na listu praktično implementira pokazivač front(čelo)
- Poslednji element predstavlja začelje reda i na njega ukazuje pokazivač rear
- Pošto operacije rade samo sa prvim i poslednjim elementom i nije potreban pristup proizvoljnom elementu, dovoljno je da lista bude jednostruko ulančana



RED REALIZOVAN KAO ULANČANA LISTA A) POČETNO STANJE B) STANJE POSLE UMETANJA ELEMENTA X C) STANJE POSLE BRISANJA

ISPITIVANJE DA LI JE RED PRAZAN

```
QUEUE-EMPTY-L(q)
if (q=nil) then
  return true
else
  return false
end_if
```

OPERACIJA UMETANJA ELEMENTA U RED

- Umeće se element sa vrednošću x u red predstavljen listom na koju ukazuje spoljašnji pokazivač q
- Predhodna slika b)

```
INSERT-L(q, x)
p=GETNODE
info(p)=x  next(p)=nil
if(rear[q]=nil) then
    q=p
else
    next(rear[q])=p
    end_if
rear[q]=p
```

OPERACIJA BRISANJA ELEMENTA IZ REDA

- Red na koji ukazuje q
- Uklanja se prvi element i vraća se njegova vrednost
- Povezuje se lista i ažurira se pokazivač na listu q
- Ako je lista postala prazna, ažurira se i pokazivač rear na nil

```
DELETE-L(q)
if (q=nil) then  return underflow
else  p = q
 x=info(p)
 q = next(p)  if(q=nil)
 then
      rear[q]=nil  end_if
 FREENODE(p)
 return x  end_if
```

PRIORITETNI RED

- Prioritetni red je struktura kod koje uređenost po sadržaju ima odlučujući uticaj na to koji se element uklanja u operaciji brisanja
- Zato je nepohodno da elementi mogu da se urede bar po jednom polju sadržaja
- Ova relacija uređenja definiše prioritet elemenata
- Dve varijante prioritetenih redova:
  - Rastući prioritetni red (elementi se umeću proizvoljno ali se pri brisanju uvek uklanja najmanji element)
  - Opadajući prioritetni red(elementi se umeću proizvoljno ali se briše najveći element)

ANALOGIJE PRIORITETNIH REDOVA

- Lekar odmah opslužuje hitne slučajeve, ako se pojave, iako postoji red čekanja pred ordinacijom
- Kraćim procesima se daje prednost u odnosu na one kojima je potrebno veće procesorsko vreme
- Zahtevi za magistralom mogu da se opslužuju po prioritetu
- Prihvatanje i obrada prekida su zasnovani na prioritetima
- Simulacija upravljanja događajima se zasniva na prioritetnom redu događaja uređenih po vremenu nastajanja

VEKTORSKA IMPLEMENTACIJA PRIORITETOG REDA

- Prioreitetni red implementiran u vektoru PQ[1:n]
- Pokazivač rear označava poslednju zauzetu poziciju u nizu
- Operacija PQ-INSERT se realizuje tako što se prvo proverava da li je red pun(rear=n) pa ako nije, inkrementira se rear i na tu poziciju se ubaci novi element
- Problem se javlja kod brisanja, kada prvo treba locirati najmanji element, a to se može učiniti ako se pretraži podniz PQ[1:rear]
- Zatim se ovaj element uklanja iz red

VEKTORSKA IMPLEMENTACIJA PRIORITETOG REDA

- Postoji više načina realizacije prioritetnih redova implementiranih kao vektor(uklanjanja elementa iz reda):
  - markiranje izbačenog elementa
  - umetanje novog elementa preko markiranog
  - pomeranje svih elemenata prilikom brisanja
  - održavanje uređenosti elemenata
- Realizacije pomoću vektora su uglavnom neefikasne!

PRVI NAČIN

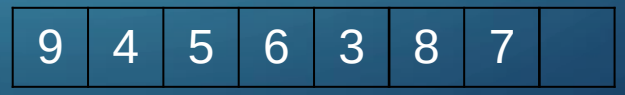
- **Brisanje se najbrže obavlja ako se samo mesto uklonjenog elementa posebno markira kao nevažeći element.**

- **Pri narednim pretraživanjima na najmanji element ignorišu se markirani**
  ! Iako se markirani elementi ignorišu, prilikom brisanja im se mora pristupiti (da bi se utvrdilo da su markirani)

  - **! Lako može da se prepuni niz koji sadrži mnogo markiranih elemenata Tada mora da se prepakuje niz (izbace markirani elementi)**

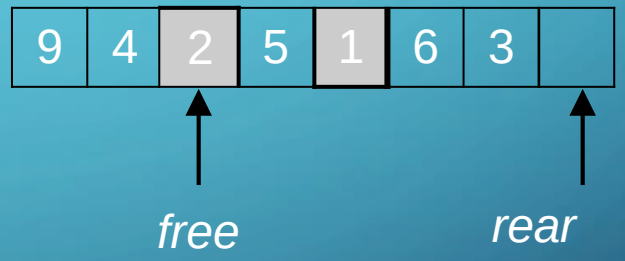
- **Dodavanje elementa vrednosti 8**

Dodavanje elementa vrednosti 7



DRUGI NAČIN

- **S obzirom da su markirane pozicije praktično slobodne, prethodni način može da se modifikuje tako što se, umesto iza poslednjeg, dozvoljava umetanje na poziciju prvog nevažećeg elementa (ako takav postoji).**



- **! Neefikasno umetanje: mora da se pretraži ceo vektor**
- **Može da se uvede pokazivač *free* koji pokazuje na prvu slobodnu lokaciju. Umetanje se onda vrši na mestu gde pokazuje *free*.**

TREĆI NAČIN

- Da bi se izbegla pojava markiranih elemenata, pri brisanju se mogu pomeriti svi elementi iza onog koji se briše za jednu poziciju nadole i dekrementirati rear.
  - Ako red nije sortiran, neefikasno je dohvatanje elementa: mora da se prođe kroz ceo vektor i pronađe odgovarajući element.
  - Radi efikasnog dohvatanja – poželjno je držati red sortiran
- ! Neefikasno umetanje: ako se ne umeće na kraj, moraju da se pomeraju elementi
- ! U proseku se pomera ½ elemenata.
- Moguće rešenje: ostaviti slobodnog prostora pre početka reda.
  - Pomerati elemente pri početku reda ako ima manje elemenata od mesta umetanja do početka reda
    - Pomerati elemente pri kraju reda u suprotnom

ČETVRTI NAČIN

- . Moguće je održavati red rastuće uređenim tako da fizički poredak elemenata odgovara njihovim logičkom poretku po prioritetu

- Pri brisanju se tretira kao obični red sa pokazivačima front i rear Element se uklanja sa pozicije front
- Pri umetanju se mora naći i osloboditi mesto koje mu po prioritetu pripada
- To se radi pomeranjem manjih elemenata nadole ili viših elemenata nagora u zavisnosti od toga kojih ima manje

- Pretraživanje u ovako uređenom redu pri umetanju je jednostavnije nego pretraživanje najmanjeg elementa u neuređenom redu

ULANČANA IMPLEMENTACIJA PRIORITETOG REDA

- Prioreitetni red se može implementirati kao jednostruko ulančana lista
- Ukoliko je lista neuređena, umetanje se vrši na početak, a pri brisanju treba pretraziti čitavu listu
- Traži se minimalni element, čuva pokazivač na njega, kao i na element ispred njega, da bi se na kraju taj element uklonio
- Ako se usvoji uređena lista, operacija umetanja elemenata mora da očuva uređenost liste i ubaci novi čvor na tačno određeno mesto koje mu po vrednosti odgovara