Strukture Podataka i Algoritmi

Lekcija 6

leto 2019/2020

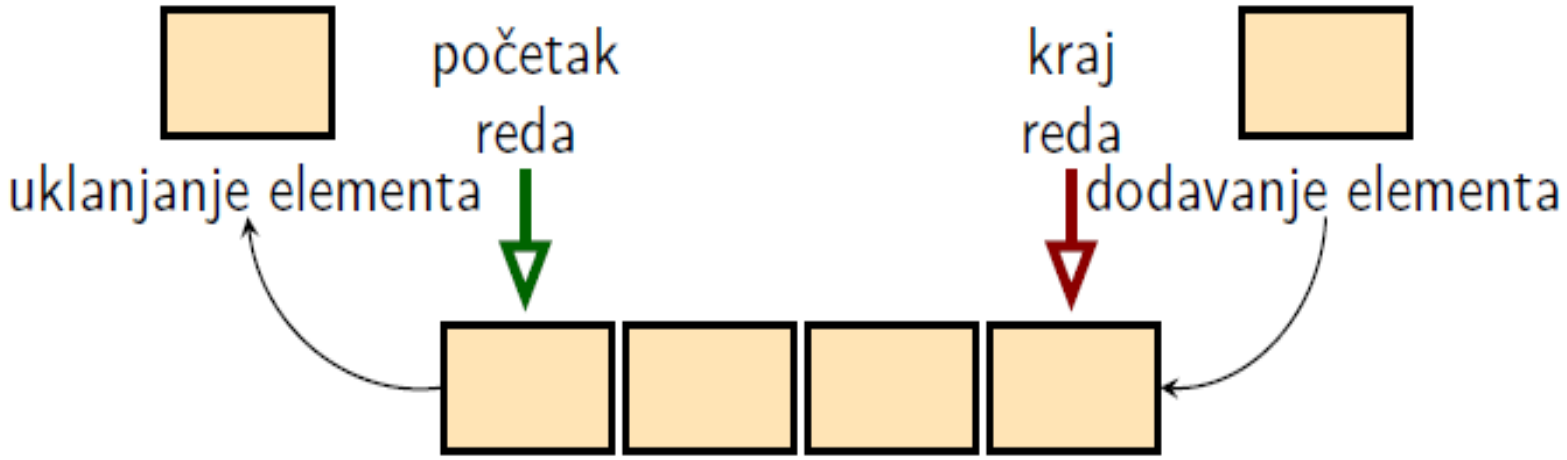
Prof. dr Branimir M. Trenkić

Redovi

- ***Red za čekanje*** ili samo ***red*** (engl. ***queue***)
- (Druga) specijalna ***vrsta liste***
- Korisiti svoja ***oba kraja*** (za razliku od steka)
- ***Intuitivni model*** - red ljudi koji čeka ispred bioskopa da kupi karte

Redovi

- Mesto na kojem se elementi ***dodaju*** logički predstavlja ***kraj (rep) reda***,
- Mesto gde se elementi ***uklanjaju*** označava ***početak (glavu) reda***

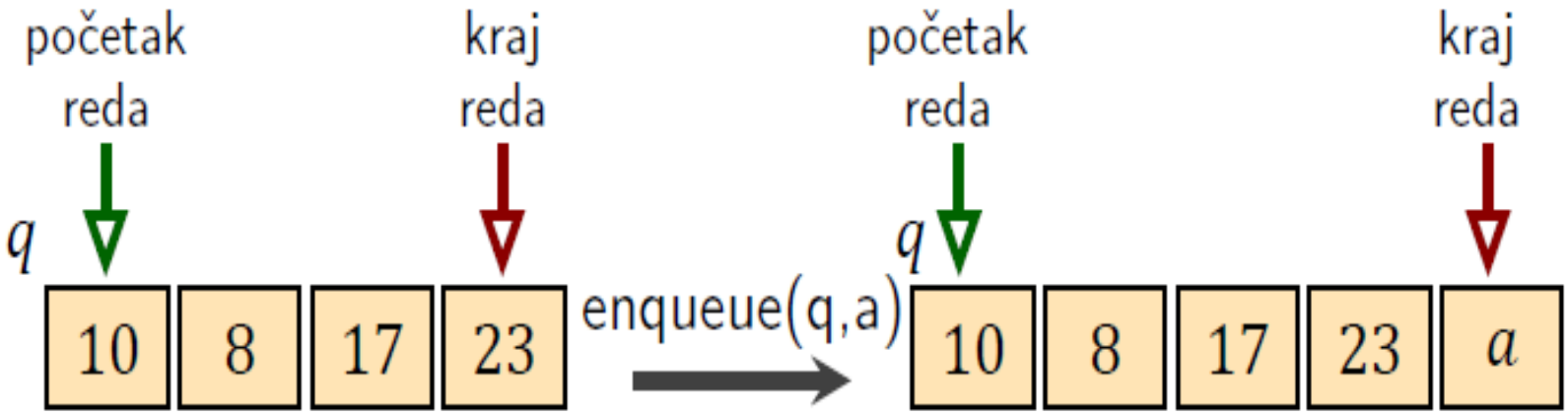


Redovi

- Struktura podataka organizovana po ***principu FIFO*** (*First-In-First-Out*)- ***poredak*** elemenata po kojem se oni ***uklanjaju*** iz reda je ***identičan*** onom po kojem su oni ***dodati*** u red
- Operacija ***dodavanja*** elemenata u red – ***enqueue***
- Operacija ***uklanjanja*** elemenata iz reda zove se ***dequeue***
- Operacija ***kreiranja*** novog (praznog) reda - ***make***

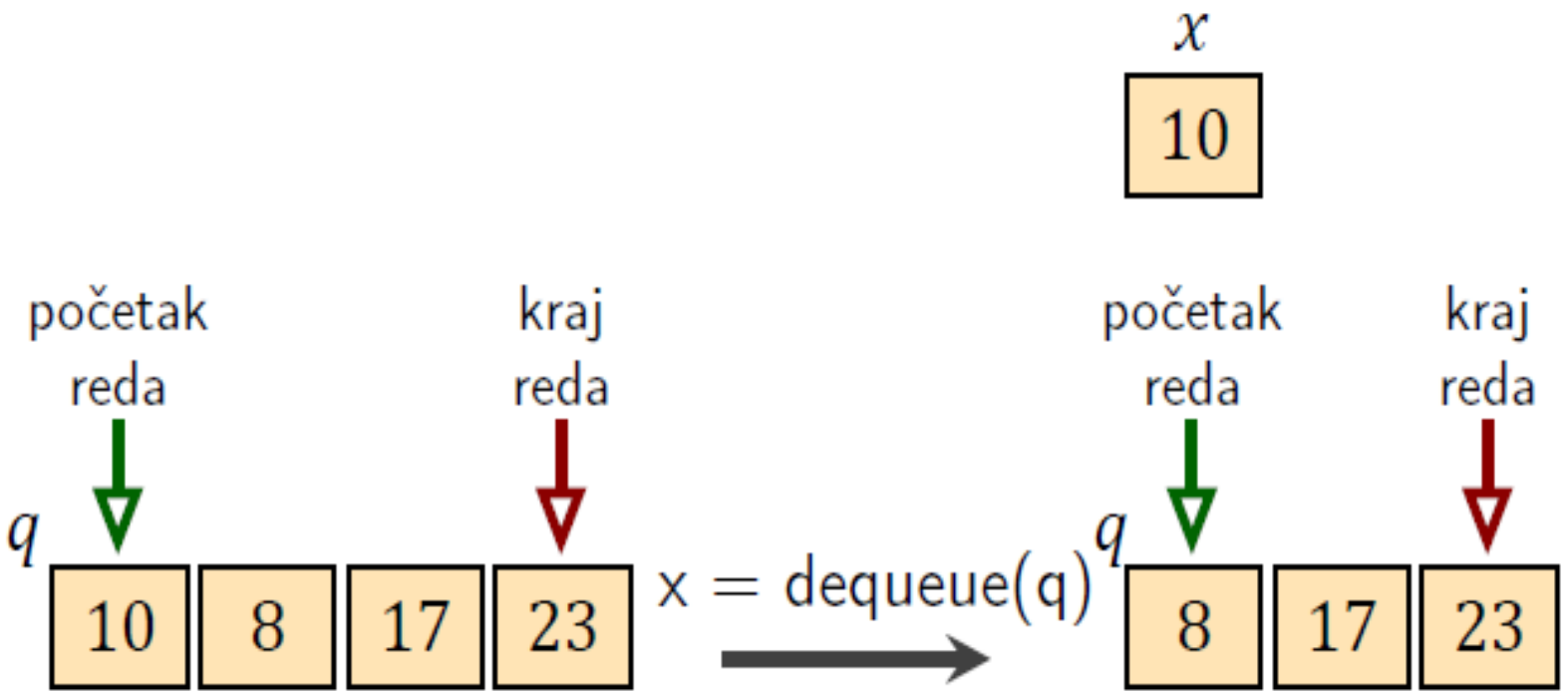
Redovi

- ***Dodavanje*** elementa na kraj reda (***enqueue***)



Redovi

- ***Uklanjanje*** elementa sa početka reda (***dequeue***)



Redovi

- ***Primena*** redova:
  - Red programa za izvršavanje u OS,
  - Red poruka (paketa) u ruterima
  - Simulacije
  - .....

Implementacija Reda

- ***Implementacija***:
  - ***Statička*** (pomoću nizova)
    - ***Trivijalno*** rešenje
    - Rešenje sa ***"kružnim" nizom***
  - ***Dinamička*** (pomoću pokazivača (ulančane liste))

Implementacija Reda

- Implementacija reda pomoću ***niza***
- Predstavljanje pomoću ***jednog objekta*** koji se sastoji od ***tri polja***
- Neophodnost da se ***stalno vodi računa*** gde su ***glava i rep*** reda (***kursori***)
  - ***Prvo polje*** (*r*) - sadrži ***indeks*** elementa niza koji trenutno predstavlja ***rep (kraj) reda***
  - ***Drugo polje*** (*g*) - sadrži ***indeks*** elementa niza koji trenutno predstavlja ***glavu (početak) reda***

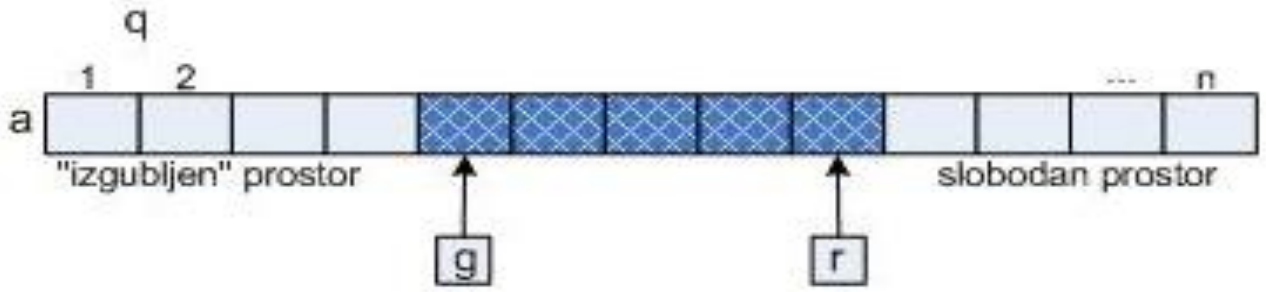
Implementacija Reda

- Implementacija reda pomoću ***niza***
- ***Treće polje*** (***a***) - predstavljeno kao ***niz*** u kome se nalazi ***sadržaj reda***
- Realizacija tog niza može biti različita

1. Niz predstavljen u obliku ***klasičnog niza*** veličine n - ***trivijalno rešenje***
2. Niz predstavljen u obliku ***“kružnog” niza*** veličine n

Implementacija Reda

- Implementacija reda pomoću niza – **trivijalno rešenje**



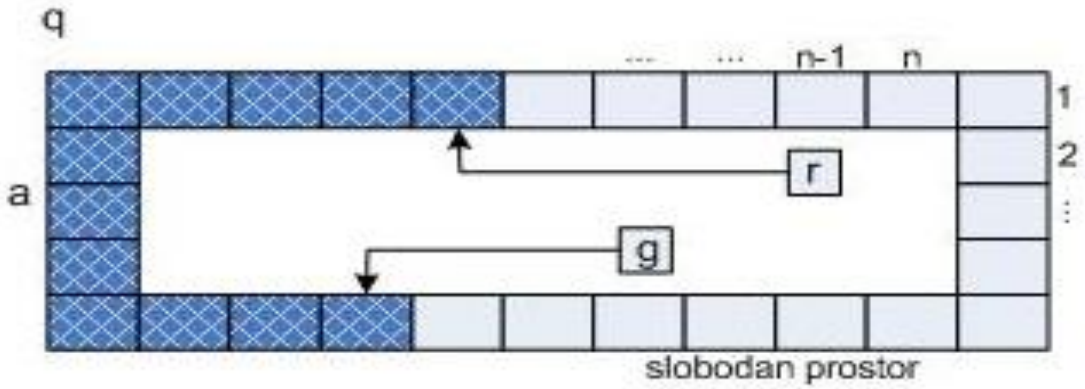
- ***Dodavanjem*** novog elementa u red, kursor ***r*** se pomera ***za jedno mesto u desno***
- ***Uklanjanjem*** elementa iz reda, kursor ***g*** se pomera ***za jedno mesto u desno***

Implementacija Reda

- Implementacija reda pomoću niza – **trivijalno rešenje**
- **Nedostatak:**
  - Neracionalno ***korišćenje memorije***
  - U procesu rada sa redom generiše se ***deo niza*** koji je praktično "***izgubljen" za dalje korišćenje***

Implementacija Reda

- Implementacija reda pomoću niza
- ***Drugi način*** realizacije niza ***a*** je u obliku ***„kružnog“*** (cirkularnog) niza veličine n



- Elementi reda se nalaze u lokacijama ***g, g+1, ..., r***
- ***„Susedne“?*** - U smislu da ***lokacija 1 sledi neposredno iza lokacije n*** u kružnom redosledu

Implementacija Reda

- Implementacija reda pomoću niza
- Kada se ***dodaje neki element***
  a. ***kursor r*** se ***uvećava za jedan*** i
  b. novi element se ***upisuje*** u poziciju na koju ***ukazuje kursor***
- Kada se neki ***element uklanja***
  a. ***kursor g*** se samo uvećava za jedan
- Kako se elementi dodaju i uklanjaju, ceo ***red se*** tokom vremena ***pomera u nizu*** u smeru kretanja kazaljke na satu

Implementacija Reda

- Implementacija reda pomoću niza
- **Nedostatak:**
  - ***Nemogućnost*** da se na osnovu vrednosti kursora r i g ***razlikuje prazan od punog reda***
- ***Rešenje:***
- Da se u okviru objekta kojim se predstavlja red uvede ***dodatno četvrto polje*** (***l***) koje sadrži aktuelni broj elemenata reda u nizu ***a***

Operacije nad redom

- ***Kreiranje praznog reda***. Ova operacija postavlja promenljive objekta u inicijalno stanje

```
// Ulaz: red q

// Izlaz: početno stanje reda q

algorithm queue-make(q)

    q.g = 1;

    q.r = n;

    q.l = 0;

    return;
```

Operacije nad redom

- ***Provera da li je red prazan***. Sastoji se od proste provere vrednosti dodatnog polja ***l***

```
// Ulaz: red q

// Izlaz: tačno ako je q prazan, inače netačno

algorithm queue-empty(q)

    if (q.l == 0)

        return true;

    else

        return false;
```

Operacije nad redom

- ***Provera da li je red pun***. Sastoji se od proste provere vrednosti dodatnog polja ***l***

```
// Ulaz: red q

// Izlaz: tačno ako je q pun, inače netačno

algorithm queue-full(q)

    if (q.l == n)

        return true;

    else

        return false;
```

Operacije nad redom

- ***Dodavanje novog elementa (enqueue)***. Ova operacija se realizuje u ***dva koraka***

```
// Ulaz: novi element x, red q

// Izlaz: red q sa elementom x na kraju

algorithm enqueue(x, q)

    if (queue-full(q) == true) then

            return "overflow";

    else

            q.r = q.r % n + 1;

            q.a[q.r] = x;

            q.l = q.l + 1;

            return;
```

1. korak
Provera da li red ulazi u stanje "**overflow**" ili ne

2. korak - dodavanje
Koristi se sabiranje po modulu kako bi se realizovala "kružna" priroda niza

Operacije nad redom

- ***Uklanjanje elementa(dequeue)***. Ova operacija se realizuje u ***dva koraka***

```
// Ulaz: red q

// Izlaz: element x uklonjen iz reda q na početku

algorithm dequeue(q)

    if (queue-empty(q) == true) then

            return "underflow";

    else

            x = q.a[q.g];

            q.g = q.g % n + 1;

            q.l = q.l - 1;

            return x;
```

1. korak
Provera da li red ulazi u stanje "**underflow**" ili ne

2. korak - uklanjanje
Koristi se sabiranje po modulu kako bi se realizovala "kružna" priroda niza