Dizajn i analiza algoritama

Lekcija 4

leto 2019/2020

Prof. dr Branimir M. Trenkić

Problem sortiranja niza

- Ako je dat ***niz neuređenih brojeva***, treba preurediti brojeve tog niza tako da oni obrazuju ***rastući niz***

- Ako je dat niz ***a*** od ***n*** elemenata $a_1, a_2, \ldots, a_n$ treba ***naći permutaciju*** svih ***indeksa*** elemenata niza $i_1, i_2, \ldots, i_n$ tako da ***novi prvi*** element a_{i1}, ***novi drugi*** element a_{i2} i tako dalje, novi *n*-ti element a_{in} u nizu ***zadovoljavaju uslov***

$$a_{i_1} \leq a_{i_2} \leq \cdots \leq a_{i_n}$$

Problem sortiranja niza

- Za rešavanje problema sortiranja, u ovom trenutku, predstavljamo ***tri jednostavna algoritma:***
  - Sortiranje ***zamenjivanjem*** (***bubble-sort***)
  - Sortiranje ***umetanjem*** (***insert-sort***)
  - Sortiranje ***izborom*** (***select-sort***)

- Isto ***vreme izvršavanja*** – ***kvadratno*** zavisi od veličine ulaza (broja elemenata niza)

1. Sortiranje zamenjivanjem

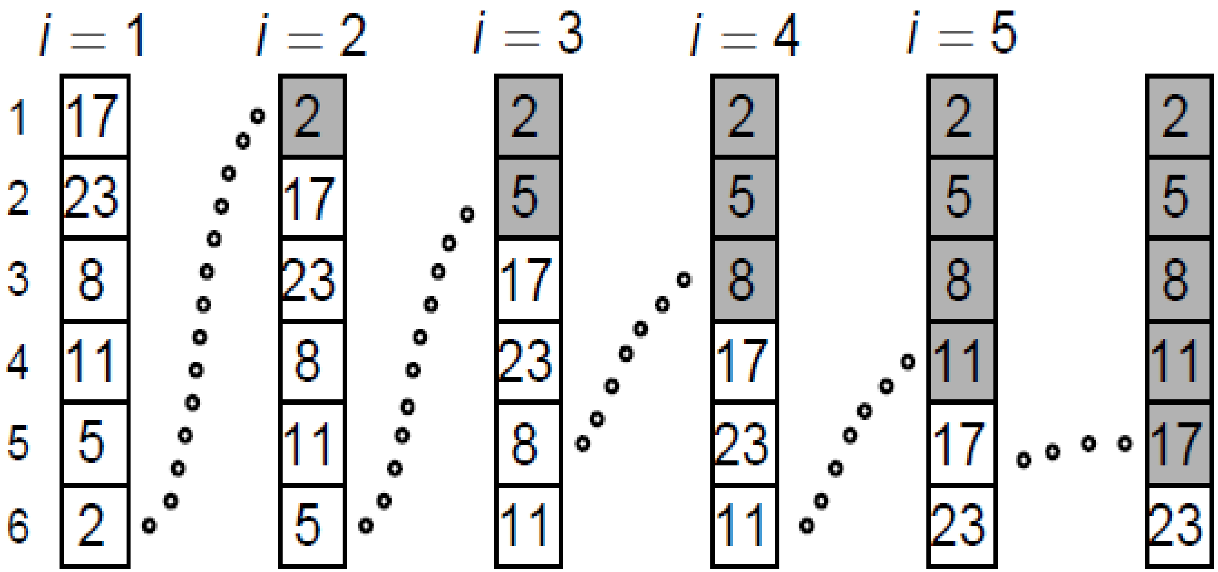
- Ulazni parametri: ***a*** i ***n***
- Izlazni parametri: ***a*** (u prerurеđenom redosledu)
- Primer ***instance problema*** sortiranja niza u rastući poredak:

a = [17, 23, 8, 11, 5, 2] (n = 6)

- ***Rešenje*** ove ***instance***: ***a = [2, 5, 8, 11, 17, 23]***

1. Sortiranje zamenjivanjem

- Najbolji ***način za razumevanje*** – zamislimo da je dati ***niz potopljen u vodu*** i okrenut ***vertikalno***
- Elementi ***"laki"*** po težini (sa malim vrednostima) će ***"isplivavati"*** na površinu (***vrh***) proizvodeći mehuriće (***bubble***)



1. Sortiranje zamenjivanjem

- ***Postupak*** sortiranja zamenjivanjem se sastoji od ***višestrukih prolaza duž datog niza*** ***od dna ka vrhu***
- Ako dva susedna elementa ***nisu u dobrom redosledu*** (manji element je ispod većeg) – ***zamenjujemo im mesta***
- ***Efekat prvog prolaza*** – ***najmanji*** element se penje ***sve do vrha***
- ***Efekat drugog prolaza*** – ***drugi najmanji*** element se penje ***do druge pozicije***

....

1. Sortiranje zamenjivanjem

- ***Ključna pitanje*** za realizaciju algoritma –

1. ***Koliko prolaza kroz niz moramo ukupno izvršiti?***
2. ***Da li u svakom prolazu treba ići do samog vrha niza?***

- ***Odgovor na prvo pitanje*** – očigledno: ako je veličina niza ***n*** elemenata – ukupan broj prolaza je jednak ***n - 1***

1. Sortiranje zamenjivanjem

- Primetimo:
- ***U drugom prolazu*** – možemo da ***stanemo do druge pozicije*** (jer se najmanji element već nalazi na prvoj poziciji)
- ***U trećem prolazu*** – možemo da ***stanemo do treće pozicije*** (jer se dva najmanja elementa već nalaze na prve dve pozicije)
- ....
- ***U i-tom prolazu*** – možemo da ***stanemo do pozicije i*** (jer se i - 1 najmanjih elementa već nalaze na prvih i – 1 pozicija)

1. Sortiranje zamenjivanjem

- ***Algoritam*** u pseudo kodu:

```
// Ulaz:   niz a, broj elemenata n niza a
// Izlaz:  niz a sortiran u rastućem redosledu
algorithm bubble-sort(a, n)

   for i = 1 to n-1 do
      for j = n downto i+1 do
         if (a[j] < a[j-1]) then
            swap(a[j], a[j-1]);

   return a;
```

i – broj prolazaka kroz niz

j – pozicije kroz koje se prođe u jednom prolazu

1. Sortiranje zamenjivanjem

- Analiza ***vremena izvršavanja algoritma***

a) Vreme ***izvršavanja swap algoritma*** – konstantno pa ga možemo smatrati ***jediničnom instrukcijom***

b) Broj izvršavanja tela unutrašnje petlje?

```
// Ulaz:   niz a, broj elemenata n niza a
// Izlaz:  niz a sortiran u rastućem redosledu
algorithm bubble-sort(a, n)

   for i = 1 to n-1 do
      for j = n downto i+1 do
         if (a[j] < a[j-1]) then
            swap(a[j], a[j-1]);

   return a;
```

1. Sortiranje zamenjivanjem

- Broj izvršavanja tela unutrašnje petlje

$$(n-1)+(n-2)+\cdots+2+1=\sum_{i=1}^{n-1} i=\frac{(n-1)n}{2}$$

- Vreme izvršavanja *bubble-sort*

$$T(n)=\frac{(n-1)n}{2}\cdot 2+1=n^2-n+1$$

1. Sortiranje zamenjivanjem

- Vreme izvršavanja algoritma ***bubble-sort*** je posmatrano ***u najgorem slučaju***
  - Algoritam ***swap*** se ***izvršava u svakom*** prolazu kroz petlju
- ***Najgori slučaj*** – ulazni niz ***a*** je početno sortiran u ***opadajućem redosledu***
- ***Najbolji slučaj*** - ulazni niz ***a*** je početno sortiran u ***rastućem redosledu***
- Algoritam ***swap*** se ***neće izvršiti ni jednom***
- U tom slučaju: $T(n) = \frac{(n-1)n}{2} \cdot 1 + 1 = 0.5n^2 - 0.5n + 1$

1. Sortiranje zamenjivanjem

- ***Izloženi algoritam*** se može jednostavno ***modifikovati*** tako da ***za "dobre" ulazne podatke*** radi mnogo ***brže***
- Poboljšanje temeljimo na činjenici –

"Ukoliko u nekom prolazu kroz niz ***ni jednom*** ne bude izvršena ***operacija swap*** – ***dodatni prolazi kroz niz nisu potrebni***!"

- Spoljašnju ***for petlju*** zameniti ***drugom petljom*** koja se ***kontroliše indikatorom*** da li je u predhodnom prolazu izvršena operacija swap (bar jednom)

1. Sortiranje zamenjivanjem

- Poboljšana varijanta algoritma

```
// Ulaz:   niz a, broj elemenata n niza a
// Izlaz:  niz a sortiran u rastućem redosledu
algorithm bubble-sort(a, n)

   repeat
      s = false;  // indikator izvršavanja operacije swap
      i = 1;
      for j = n downto i+1 do
         if (a[j] < a[j-1]) then
            swap(a[j], a[j-1]);
            s = true;
      i = i + 1;
   until (s == false);

   return a;
```

2. Sortiranje umetanjem

- Ovaj postupak se sastoji od ***n iteracija***,
- ***U i-toj iteraciji*** ***umeće se element*** a_i na njegovo pravo mesto između prvih ***i – 1*** najmanjih elemenata predhodno sortiranim u rastući poredak

2. Sortiranje umetanjem

- Diskusija algoritamske realizacije
- Obezbeđivanje da ***izlazni uslov*** iterativnog postupka koji realizuje ***zamenjivanje elemenata*** u sortiranom delu niza – ***u jednom trenutku bude zadovoljen***
- Uvodimo ***nulti element niza*** ($\boldsymbol{a_0}$) za čiju se ***vrednost*** predpostavlja da je ***manja*** od bilo koje ***vrednosti elemenata*** niza ***a*** (a_1, a_2, ..., a_n)
- Takva vrednost se naziva ***sentinela*** ($\boldsymbol{a_0 = -\infty}$)

2. Sortiranje umetanjem

- Diskusija algoritamske realizacije kroz konkretni primer

1. iteracija	3	7	4	9	5	2	6	1
2. iteracija	3	7	4	9	5	2	6	1
3. iteracija	3	7	4	9	5	2	6	1
	3	4	7	9	5	2	6	1
	3	4	7	9	5	2	6	1
	3	4	5	7	9	2	6	1
	2	3	4	5	7	9	6	1
	2	3	4	5	6	7	9	1
	1	2	3	4	5	6	7	9

2. Sortiranje umetanjem

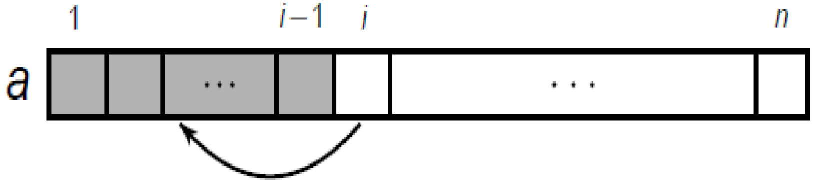
- ***Algoritam*** u pseudo kodu:

```
// Ulaz:   niz a, broj elemenata n niza a
// Izlaz:  niz a sortiran u rastućem redosledu
algorithm insert-sort(a, n)

   a[0] = -∞;
   for i = 1 to n do
      j = i
      while (a[j] < a[j-1]) do
         swap(a[j], a[j-1]);
         j = j - 1;

   return a;
```

2. Sortiranje umetanjem

- Analiza ***vremena izvršavanja algoritma***

Broj iteracija ***while petlje nije očigledan***!
Broj zamenjivanja i-tog elementa prilikom njegovog umetanja na pravo mesto.
U najgorem slučaju:



to je ***i -1 puta***

```
// Ulaz:   niz a, broj elemenata n niza a
// Izlaz:  niz a sortiran u rastućem redosledu
algorithm insert-sort(a, n)

    a[0] = -∞;
    for i = 1 to n do
        j = i
        while (a[j] < a[j-1]) do
            swap(a[j], a[j-1]);
            j = j - 1;

    return a;
```

2. Sortiranje umetanjem

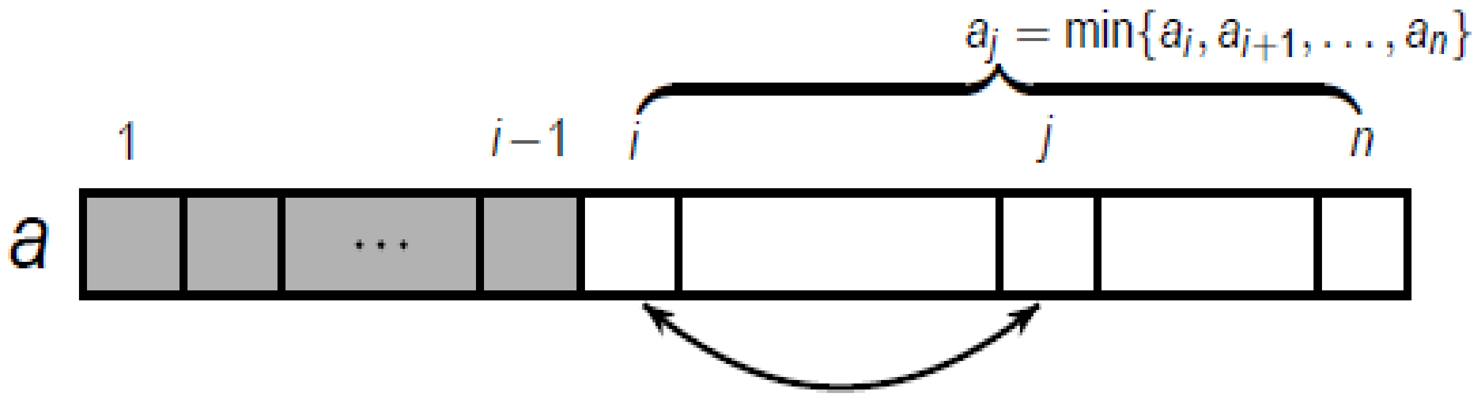
- Analiza ***vremena izvršavanja algoritma***

$$
\begin{aligned}
T(n) &= 1 + \sum_{i=1}^{n}(1 + 2(i-1)) + 1 \\
&= 1 + \sum_{i=1}^{n} 1 + 2\sum_{i=1}^{n}(i-1) + 1 \\
&= 1 + n + 2\frac{(n-1)n}{2} + 1 \\
&= n^2 + 2
\end{aligned}
$$

3. Sortiranje izborom

- Ponavljajući (iterativni) postupak:
- U ***i-toj*** iteraciji – ***određuje se minimalni element*** među elementima a_i, a_{i+1}, ..., a_n i ***zamenjuje se*** sa elementom a_i

i-ta iteracija:



3. Sortiranje izborom

- ***Ispravnost postupka***:
- ***Posle prve iteracije*** – ***najmanji element*** niza će biti ***na prvoj poziciji***
- ***Posle druge iteracije*** – ***drugi najmanji element*** niza će biti ***na drugoj* pozicij**i

……………………

- ***Nakon n -1 iteracije*** – ***ceo niz*** će biti raspoređen u rastućem redosledu

3. Sortiranje izborom

- ***Algoritam*** u pseudo kodu:

```
// Ulaz:   niz a, broj elemenata n niza a
// Izlaz:  niz a sortiran u rastućem redosledu
algorithm select-sort(a, n)

   for i = 1 to n-1 do
      j = minr(a, i, n);
      swap(a[i], a[j]);

   return a;
```

3. Sortiranje izborom

- <u>Pomoćni algoritam</u> ***minr*** – vraća indeks minimalnog elementa podniza niza ***a*** sa početnom pozicijom ***i***

```
// Ulaz: niz a, pocetni indeks "desnog" podniza, broj elemenata n niza
// Izlaz: indeks najmanjeg elementa "desnog" podniza
algorithm minr(a, i, n)

 m = a[i];        // najmanji element nadjen do sada
 j = i;           // indeks najmanjeg elementa

 k = i + 1;
 while (k <= n) do
        if (m > a[k]) then     // nadjen manji element od privremeno
                               // najmanjeg
             m = a[k];               // zapamti manji broj
             j = k;                  // i njegov indeks
        k = k + 1;          // predji na sledeci element

 return j;        // vrati indeks najmanjeg elementa
```

3. Sortiranje izborom

- Analiza ***vremena izvršavanja algoritma***
- Algoritam ***minr***:

$$T(n) = 1 + 1 + 2 + 4(n - i) + 1 = 5 + 4(n - i)$$

- Algoritam ***select-sort***:
- ***Telo petlje***: ***2*** vremenske jedinice (swap + dodeljivanje) + ***minr***

3. Sortiranje izborom

- Analiza ***vremena izvršavanja algoritma***
- Algoritam ***select-sort***:

```
algorithm select-sort(a, n)

  for i = 1 to n-1 do
     j = minr(a, i, n);
     swap(a[i], a[j]);

  return a;
```

$$T(n)=\sum_{i=1}^{n-1}\left(5+4(n-i)+2\right)+1$$

$$=\sum_{i=1}^{n-1}\left(4(n-i)+7\right)+1$$

$$=4\frac{(n-1)n}{2}+7(n-1)+1$$

$$=2n^2+5n-6$$

Pretraga (sekvencijalna) niza

- Ako su dati niz ***a*** *od* ***n*** ***neuređenih brojeva*** i jedan ***broj*** ***x****,* treba odrediti da li se broj ***x*** nalazi u nizu ***a****.*
- Ukoliko je to slučaj, rezultat treba da bude indeks niza ***i*** takav da je $x = a_i$ *;*
- U suprotnom slučaju, rezultat treba da bude ***0***

Pretraga (sekvencijalna) niza

- ***Algoritam*** u pseudo kodu:

```
// Ulaz:  niz a, njegov broj elemenata n, broj x
// Izlaz: k takvo da x = a_k, ili 0 ako x nije u nizu a
algorithm seq-search(a, n, x)

   for k = 1 to n do  // ispitati sve elemente niza a
      if (a[k] == x) then
         return k;    // x je nađen u k-toj poziciji

   return 0;          // x nije nađen
```

Pretraga (sekvencijalna) niza

- Analiza ***vremena izvršavanja algoritma***

$$T(n) = 2 \cdot n + 1$$