

RI203

Uvod u računarske algoritme

dr.sc. Edin Pjanić

Pregled predavanja

- Elementarni algoritmi sortiranja
 - Bubble sort
 - Sortiranje odabirom (selection sort)
 - Sortiranje umetanjem (insertion sort)
- Stabilnost sortiranja
- Shell sort (Šelov metod sortiranja)

Algoritmi sortiranja

- Algoritam sortiranja je algoritam koji postavlja elemente liste u određeni poredak. Najčešći kriteriji:
 - numerički poredak,
 - leksikografski poredak.
- Izlaz iz algoritma sortiranja je lista koja je:
 - u neopadajućem redoslijedu u odnosu na ulaz,
 - permutacija ulaza.
- *Ulaz: sekvenca elemenata $\langle a_1, a_2, \dots, a_n \rangle$*
- *Izlaz: permutacija $\langle a_1', a_2', \dots, a_n' \rangle$ ulazne sekvence takva da je:*
$$a_1' \leq a_2' \leq \dots \leq a_n'.$$
- Ulazna sekvenca je najčešće niz, ali može biti bilo koja implementacija liste.
- *Pri sortiranju elemenata u praksi se često kao kriterij poretka koristi poredak tzv. ključeva. Ključ je dio većeg zapisa (elementa) koji, osim ključa, sadrži i druge podatke. Obično su ti drugi podaci mnogo većeg obima od ključa.*

Algoritmi sortiranja

- Razvijeno je mnogo algoritama sortiranja. Najpoznatiji su:

Bubble Sort	Heap Sort
Selection Sort	Merge Sort
Insertion Sort	Quick Sort
- Najčešći kriterij za poređenje je operator $<$.
- Najčešća klasifikacija algoritama sortiranja na osnovu:
 - složenosti u odnosu na broj poređenja ili na broj zamjene (swap) elemenata,
 - stabilnosti,
 - korištene memorije ili drugih resursa, itd.
- **Selection sort** i **Bubble sort** su najsporiji (opravdanost korištenja?):
 - $O(n^2)$
- **Quick sort** je najefikasniji i najbrži:
 - $O(n \lg n)$

Bubble sort

- Kao što mjehurići zraka jedan po jedan izranjaju na površinu vode, tako i u ovom algoritmu jedan po jedan element ide na kraj niza. Najprije najveći element pa prvi manji itd.
- Krećemo od početka niza od n elemenata i poredimo prva dva elementa. Ako je desni element manji od lijevog zamijenimo im mjesta.
- Sad element na drugoj poziciji poredimo sa trećim. Ako je element na 3. poziciji manji od onog na 2. poziciji zamijenimo im mjesta.
- Tako ponavljamo do kraja niza.
- Najveći element će tako "isplivati" na kraj niza.
- Ponovimo postupak ispočetka za $n-1$ elemenata.
- Postupak ponavljamo dok ima zamjena.

Bubble sort - primjer

- | | |
|---------------|---------------|
| ▪ 3 5 4 2 6 1 | ▪ 3 2 4 1 5 6 |
| ▪ 3 5 4 2 6 1 | ▪ 2 3 4 1 5 6 |
| ▪ 3 4 5 2 6 1 | ▪ 2 3 4 1 5 6 |
| ▪ 3 4 2 5 6 1 | ▪ 2 3 1 4 5 6 |
| ▪ 3 4 2 5 6 1 | ▪ 2 3 1 4 5 6 |
| ▪ 3 4 2 5 1 6 | ▪ 2 1 3 4 5 6 |
| ▪ 3 4 2 5 1 6 | |
| ▪ 3 2 4 5 1 6 | ▪ 1 2 3 4 5 6 |
| ▪ 3 2 4 5 1 6 | |

Bubble sort – primjer C++ funkcije

```
template<typename Elem>
void bubble_sort(Elem a[], int n)
{
    for(int i=n; i>1; i--)
        for(int j=1; j<i; j++)
            if(a[j] < a[j-1]) swap(a[j-1], a[j]);
}
```

Broj poređenja u zavisnosti od broja elemenata:

$$T(n) = n-1 + n-2 + \dots + 3 + 2 + 1$$

$$\Rightarrow bc, ac, wc: O(n^2)$$

Bubble sort je NAJGORI metod sortiranja. Koristiti samo u krajnjoj nuždi!

Selection sort

- I ovo je vrlo naivan metod.
- Krećemo od početka niza od n elemenata, idemo do kraja niza i tražimo najmanji element (zapamtimo indeks).
- Kad dođemo do kraja niza zamijenimo element na prvom mjestu sa najmanjim elementom, ako postoji manji od prvog.
- Sad sve ponovimo od elementa na drugoj poziciji, tj. tražimo najmanji element u podnizu veličine $n-1$.
- Ponovimo isto od trećeg elementa itd.
- Tako ponavljamo do kraja niza.

Broj poređenja u zavisnosti od broja elemenata:

$$T(n) = n-1 + n-2 + \dots + 3 + 2 + 1 = n(n-1) / 2$$
$$\Rightarrow bc, ac, wc: O(n^2)$$

Selection sort - primjer

■ 3 4 2 5 6 1

■ 3 4 2 5 6 1

■ 3 4 2 5 6 1

■ 3 4 2 5 6 1

■ 3 4 2 5 6 1

■ 1 4 2 5 6 3

■ 1 4 2 5 6 3

■ 1 4 2 5 6 3

■ 1 4 2 5 6 3

■ 1 2 4 5 6 3

■ 1 2 4 5 6 3

■ 1 2 4 5 6 3

■ 1 2 3 5 6 4

■ 1 2 3 5 6 4

■ 1 2 3 4 6 5

■ 1 2 3 4 5 6

Selection sort – primjer C++ funkcije

```
template<typename Elem>
void selection(Elem a[], int n)
{
    for (int j = 0; j < n-1; j++) {
        int indeks_min = j;
        for (int i = j+1; i < n; i++) {
            if (a[i] < a[indeks_min]) {
                indeks_min = i;
            }
        }

        if( indeks_min != j ) {
            swap(a[j], a[indeks_min]);
        }
    }
}
```

Broj poređenja u zavisnosti od broja elemenata n :

$$T(n) = n-1 + n-2 + \dots + 3 + 2 + 1$$

$$\Rightarrow bc, ac, wc: O(n^2)$$

Koliko se izvrši swap operacija? (uočiti razliku u odnosu na bubble sort)

Insertion sort – sortiranje umetanjem

- Princip je pomalo sličan kao kod bubble sorta ali je algoritam efikasniji.
- Osnovna ideja je da je prednji dio niza uvijek sortiran.
- Krećemo od početka niza od n elemenata i poredimo prva dva elementa. Ako je desni element manji od lijevog zamijenimo im mjesta.
- Uzmemo element na 3. poziciji, idemo prema početku niza i pronalazimo njegovu poziciju s obzirom na to da su prva dva elementa već poredana od manjeg ka većem.
- Dakle, umjesto da najveći izranja na kraj, kao kod bubble-sorta, ovdje se manji umeće naprijed na svoje mjesto.
- Ponavljamo za ostale elemente, jedan po jedan do kraja niza.
- Svaki put posmatrani element postavljamo na poziciju ispred elementa od kojeg je taj element manji.

Insertion sort - primjer

▪ 3 5 4 2 6 1

▪ 3 5 4 2 6 1

▪ 3 4 5 2 6 1

▪ 3 4 5 2 6 1

▪ 3 4 2 5 6 1

▪ 3 2 4 5 6 1

▪ 2 3 4 5 6 1

▪ 2 3 4 5 6 1

▪ 2 3 4 5 1 6

▪ 2 3 4 1 5 6

▪ 2 3 1 4 5 6

▪ 2 1 3 4 5 6

▪ 1 2 3 4 5 6

Insertion sort – primjer C++ funkcije

```
void insertion(Elem a[], int n)
{
    for(int i=1; i<n; i++)
    {
        int j=i;
        while(j>0 && a[j] < a[j-1])
        {
            swap(a[j-1], a[j]);
            --j;
        }
    }
}
```

```
void insertion(Elem a[], int n) // kompaktnija funkcija
{
    for(int i=1; i<n; i++)
        for(int j=i; j && a[j]<a[j-1]; --j)
            swap(a[j-1], a[j]);
}
```

Insertion sort – analiza

$$T(n) = 1 + (1 \mid 2) + (1 \mid 2 \mid 3) + (1 \mid 2 \mid 3 \mid 4) \\ + \dots + (1 \mid 2 \mid \dots \mid n-1)$$

Dakle, imamo $n-1$ iteracija sa po 1 ili 2 ili 3 ... ili $n-1$ poređenja.

$$bc: T(n) = 1 + 1 + 1 + \dots + 1 = n-1$$

$$wc: T(n) = 1 + 2 + 3 + \dots + n-1 = n(n-1)/2$$

$$ac: T(n) = wc/2$$

$$\Rightarrow bc: O(n), ac: O(n^2), wc: O(n^2)$$

U prosjeku, upola kraće od bubble sortiranja.

Stabilnost sortiranja - definicija

- Definicija: Sortiranje je stabilno ako pri sortiranju ne dolazi do zamjene relativnih pozicija elemenata koji imaju istu vrijednost ključa.
- To znači da ako elementi **a** i **b** imaju istu vrijednost ključa i ako je **a** bilo prije **b**, nakon stabilnog sortiranja **a** će i dalje biti ispred **b**
- Npr. ako imamo listu studenata sa brojem osvojenih bodova sortiranu po abecednom redu. Ako takvu sortiranu listu sortiramo stabilnim metodom po broju bodova onda će u rezultatu studenti sa istim brojem bodova zadržati abecedni poredak.
- Bubble sort – stabilan
- Selection sort (sortiranje odabirom) – nije stabilan
- Insertion sort (sortiranje umetanjem) – stabilan

Shell sort (Šelov metod sortiranja)

- Baziran na sortiranju umetanjem (insertion sort).
- Autor je Donald Shell, 1959.
- Insertion sort je spor jer vrši zamjenu samo susjednih elemenata:
 - ako su najmanji elementi na kraju niza, treba im približno N koraka da dođu na svoje mjesto
- Insertion sort je dobar kod približno-sortiranih nizova
- Shell sort: ideja je da se poboljša insertion sort na način da manji elementi, umjesto po jedno mjesto, brže dođu na svoje mjesto ili vrlo blizu svog mjesta.
 - uzimamo svaki k -ti element i takvu sekvencu sortiramo insertion sortom (k -sortiran niz)
 - u početnim prolascima kroz niz, k je veće a u kasnijim prolascima je sve manje
 - na kraju imamo $k=1$ (standardni insertion sort)

Shell sort - primjer

18 32 12 5 38 33 16 2 25

$N=9, k=N/2=4$

18 32 12 5 **38** 33 16 2 **25**

18 32 12 5 **25** 33 16 2 **38**

18 **32** 12 5 25 **33** 16 2 38

18 32 **12** 5 25 33 **16** 2 38

18 32 12 **5** 25 33 16 **2** 38

18 32 12 **2** 25 33 16 **5** 38

$k=k/2=2$

18 32 **12** 2 **25** 33 **16** 5 **38**

12 32 **16** 2 **18** 33 **25** 5 **38**

12 **32** 16 **2** 18 **33** 25 **5** 38

12 **2** 16 **5** 18 **32** 25 **33** 38

Shell sort - primjer

$k=k/2=1$ (standardni insertion sort)

12	2	16	5	18	32	25	33	38
2	12	16	5	18	32	25	33	38
2	5	12	16	18	32	25	33	38
2	5	12	16	18	25	32	33	38

GOTOVO!

- Diminishing increment sort – metod opadajućeg inkrementa.
- Izbor sekvence inkrementa indeksa (**k**) je izuzetno bitan za efikasnost algoritma.
- Algoritam je jednostavan ali je analiza složenosti veoma komplikovana.
- Tačna analiza složenosti Shell sorta je još uvijek otvoren problem (neriješen).

Shell sort – izbor inkrementa

- Postoji više predloženih sekvenci inkrementa. Neke su:

Izraz	Vrijednosti	Složenost (wc)	Autor
$N/2^i$	$N/2, N/4, \dots, 4, 2, 1$	$O(n^2)$ za $N=2^p$	Shell
$2^i - 1$	$1, 3, 7, 15, 31, \dots$	$O(n^{3/2})$	Hibbard
$(3^i - 1)/2, < N/3$	$1, 4, 13, 40, 121, \dots$	$O(n^{3/2})$	Knuth
1 pa onda: $4^i + 3 \cdot 2^{i-1} + 1$	$1, 8, 23, 77, 281, \dots$	$O(n^{4/3})$	Sedgewick

- Shell sort je dobar za liste srednjih veličina (do 5000). Za velike liste ovo i nije najbolji algoritam.
- Ipak, algoritam je jednostavan i u većini slučajeva je dobar izbor od kojeg treba krenuti. Ako ne zadovoljava, koristiti quicksort ili neki drugi.

Usporedba metoda sortiranja koji su $O(n^2)$

