

Osnovni algoritmi za urejanje

- urejanje z izbiranjem
- urejanje z vstavljanjem
- urejanje s premenami

Vsem algoritmom je skupno:

- enostavno urejanje
- preprosta zasnova
- ne najbolj učinkoviti
- primereni za majhne tabele (enostavna koda odtehta učinkovitost)

Urejanje z izbiranjem

Primerno je za manjše tabele, ker za večje tabele obstajajo bolj učinkoviti postopki. Časovna zahtevnost urejanja z izbiranjem je $O(n^2)$.

Tabela je med urejanjem razdeljena na urejeni del, ki mu sledi neurejeni del. Na začetku je urejeni del prazen in neurejeni del celotna tabela. Nas vsakem koraku urejeni del povečamo za en element tako, da poiščemo najmanjši element neurejenega dela in ga zamenjamo z elementom, ki je na začetku neurejenega dela.

Pri urejanju z izbiranjem na vsakem koraku izberemo v neurejenem delu tabele najmanjši element in ga zamenjamo z i-tim elementom. Če ima tabela n elementov, je potrebno ta postopek ponoviti **n-1**-krat.

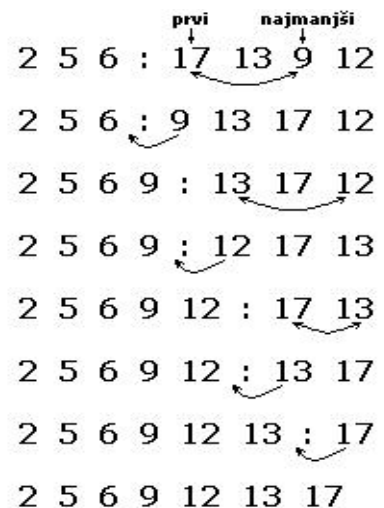
Zato se indeks i spreminja od 0 do n-1.

SPLOŠNO:

- poišči najmanjšega
- zamenjaj s prvim
- od 2 do n išči najmanjšega
- zamenjaj z drugim, itd...
- imamo dva dela: UREJENI DEL / NEUREJENI DEL

ALGORITEM:

- na začetku je urejeni del prazen
- PONAVLJAJ:
 - v neurejenem delu zamenjaj prvi in najmanjši element
 - najmanjši (prvi) element v neurejenem delu preseli v urejen del - urejeni del se poveča za ena
- končamo, ko izčrpamo neurejeni del



```
void uredi_z_izbiranjem(int *a, int n)
{
    int i, j;
    int p, max;
    for (i = 0; i < n-1; i++)
    {
        max = a[i];
        p = i;
        for (j=i+1; j<n; j++)
        {
            if (a[j] < max)
            {
                p = j;
                max = a[j];
            }
        }
        a[p] = a[i];
        a[i] = max;
    }
}
```

Urejanje z vstavljanjem

Primerno je za urejanje podatkov, ki so že skoraj urejeni. Časovna zahtevnost postopka je v najslabšem primeru $O(n^2)$. Najslabši primer je tabela, ki je obratno urejena. Če je pa tabela skoraj urejena se časovna zahtevnost približa $O(k^2n)$, kjer je k število že urejenih delov.

Pri urejanju z vstavljanjem vstavimo na vsakem koraku i -ti element na pravo pozicijo v urejenem delu tabele. Če ima tabela n elementov, je treba ta postopek ponoviti $n-1$ -krat. Zato se indeks i spreminja od 1 do $n-1$.

PRIMER (Urejanje kart):

- v roki imamo nekaj urejenih kart
 - vzamemo novo karto
 - naredimo prostor zanjo
 - vstavimo na ustrezno mesto
- >kar pomeni v našem primeru--> v urejen del vstavimo nov podatek

urejeni del				:	neurejeni del						
2	5	11		:	7	*	*	*	*	*	*
2	5	7	11	:	9	*	*	*	*	*	*
...											

ALGORITEM:

- imamo dva dela: UREJENI DEL / NEUREJENI DEL
- vzemi prvi element iz neurejenega dela in ga vstavi na pravo mesto v urejeni del
- na začetku je urejeni del prvi element
- končamo, ko izčrpamo neurejeni del

```
void urejanje_vstavljanje(int *x,int n)
{
    int i,j,pom;

    for (i=1; i<n; i++)
    {
        pom=x[i]; j=i-1;
        while ((pom < x[j]) && (j>=0))
        {
            x[j+1] = x[j];
            j--;
        }
        x[j+1] = pom;
    }
}
```

Urejanje s premenami (mehurčno urejanje)

Pri urejanju s premenami na vsakem koraku premaknemo en element na pravo mesto, ostale pa pogreznemo eno ali več mest bliže končnemu. Postopek ponovimo **n-1**-krat.

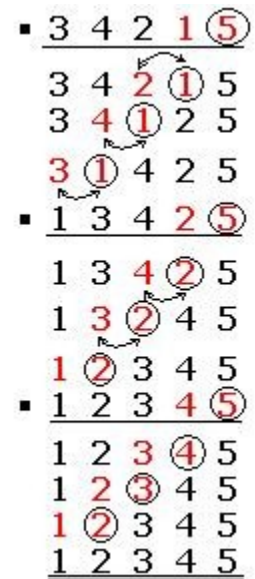
ALGORITEM:

- vzamemo zadnji element in ga primerjamo s svojim predhodnikom:
- če je predhodnik manjši, potem sta elementa v pravem vrstem redu in zgrabimo predhodnika
- če predhodnik ni manjši, ju zamenjamo
- ...
- najlažji element splava na površje
- plavanje - zamenjave

Časovna zahtevnost: **$O(n^2)$** v najslabšem primeru (ko so vrednosti v obratnem vrstnem redu) in **$O(n)$** v najboljšem primeru (ko so vrednosti urejene). Manj časa porabi tudi v primeru, ko so vrednosti skoraj na svojih pravih mestih.

```
void urejanje_premene(int *x, int n)
{
    int i,j,pom;

    for (i=0;i<n-1;i++)
        for (j=n-1;j>0;j--)
        {
            if (x[j-1]>x[j])
            {
                pom=x[j-1];
                x[j-1]=x[j];
                x[j]=pom;
            }
        }
}
```



1. Izdelaj podprogram, ki napolne tabelo velikosti n z naključnimi vrednostmi od 10-100
2. Izdelaj podprogram, ki izpiše tabelo velikosti n
3. Izdelaj podprogram, ki v tabelo velikosti n vrine podatek na i-to mesto
4. Izdelaj podprogram, ki iz tabele velikosti n izbriše podatek na i-tem mestu
5. Izdelaj podprogram, ki uredi tabelo po velikosti
6. Napišite podprogram za urejanje z izbiranjem, ki ne bo upošteval, da so po ASCII tabeli male črke za velikimi.
7. Uredi današnji datum z metodo urejanja z izbiranjem, vstavljanjem in s premenami.