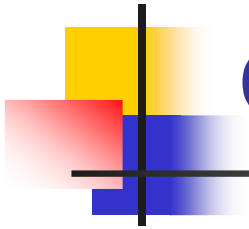
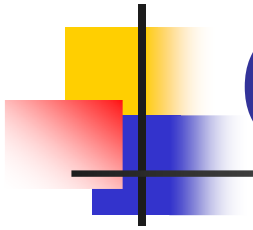
Елементарне методе сортирања



- Сортирање низа у неопадајућем или нерастућем поретку је поступак проналажења једне пермутације елемената низа у којој се елементи појављују у неопадајућем тј. нерастућем поретку
- Елементарне методе сортирања су временске сложености $O(n^2)$
- Једноставне су али неефикасне за велике скупове ($n > 500$)
- Омогућавају изучавање сложенијих
  - Сортирање избором највећег (најмањег) елемента (selection sort)
  - Сортирање уметањем, шелсорт
  - Мехурасто сортирање (bubble sort)

Сортирање избором највећег (најмањег) елемента



- Сортирање низа (x_n) од n елемената
- Поступак
  - Највећи елемент низа доводи се на прво место
  - Највећи од преосталих n-1 елемената доводи се на друго место
  - Највећи од преосталих n-2 елемената доводи се на треће место, ...
  - Већи од преостала два елемента доводи се на претпоследње место
  - На последњем месту остаје најмањи елемент

Сортирање избором највећег (најмањег) елемента

- Глобални алгоритам
  - (а) одредити највећи елемент x_{max}
  - (б) разменити x_{max} и x_1
  - (в) поновити кораке (а) и (б) посматрајући низове $x_2,...,x_n$; $x_{3,...,}$ x_n; ... x_{n-1},x_n

- Псеудопрограм

```
for (i=1; i<n; i++)
{
        naći najveći element u nizu x[i], x[i+1],...,x[n] i zapamtiti
njegov indeks max;
        razmeniti x[i] i x[max];
}
```

Налажење максималног елемента низа

- Нека је дат низ $x[m..n]$ Задатак је наћи индекс j т.д. $x_j=\max(x_m,\ldots,x_n)$
- Поступак
  - Први у низу се прогласи за највећи (max = *m)*
  - Највећи се пореди са свим осталим елементима (са индексима од $m + 1$ до n)
  - Сваки елемент већи од највећег проглашава се за највећи а његов индекс *j* за индекс највећег, *max* (*max* = *j*)

Налажење максималног елемента низа: функција

```
int maxelem(int x[ ], int m, int n)
{
   int max;
   max = m;
   for (j=m+1; j <= n; j++)
        if (x[j] > x[max]) max=j;
   return max;
}
```

Функција сортирање избором највећег (најмањег) елемента

```
void swap(int x[ ], int i, int j);
void maxelem(int x[ ], int m, int n);
void maxsort(int x[ ], int n)
{
    int i,j,max;
    for(i=0; i<n–1; i++)
    swap(x, i, maxelem(x, i, n–1));
}
void swap(int x[ ], int i, int j)
{
    int priv;
    priv = x[i];
    x[i] = x[j];
    x[j] = priv;
}
```

Сортирање избором највећег (најмањег) елемента: сложеност

- Добра својства за низове до 1000 елемената
- Број поређења међу елементима низа је око $n^2/2$ јер се спољашња петља извршава n пута а унутрашња око $n/2$ пута у просеку
- Укупно време извршавања алгоритма је пропорционално *са* n^2

Сортирање уметањем: претпоставка, поступак

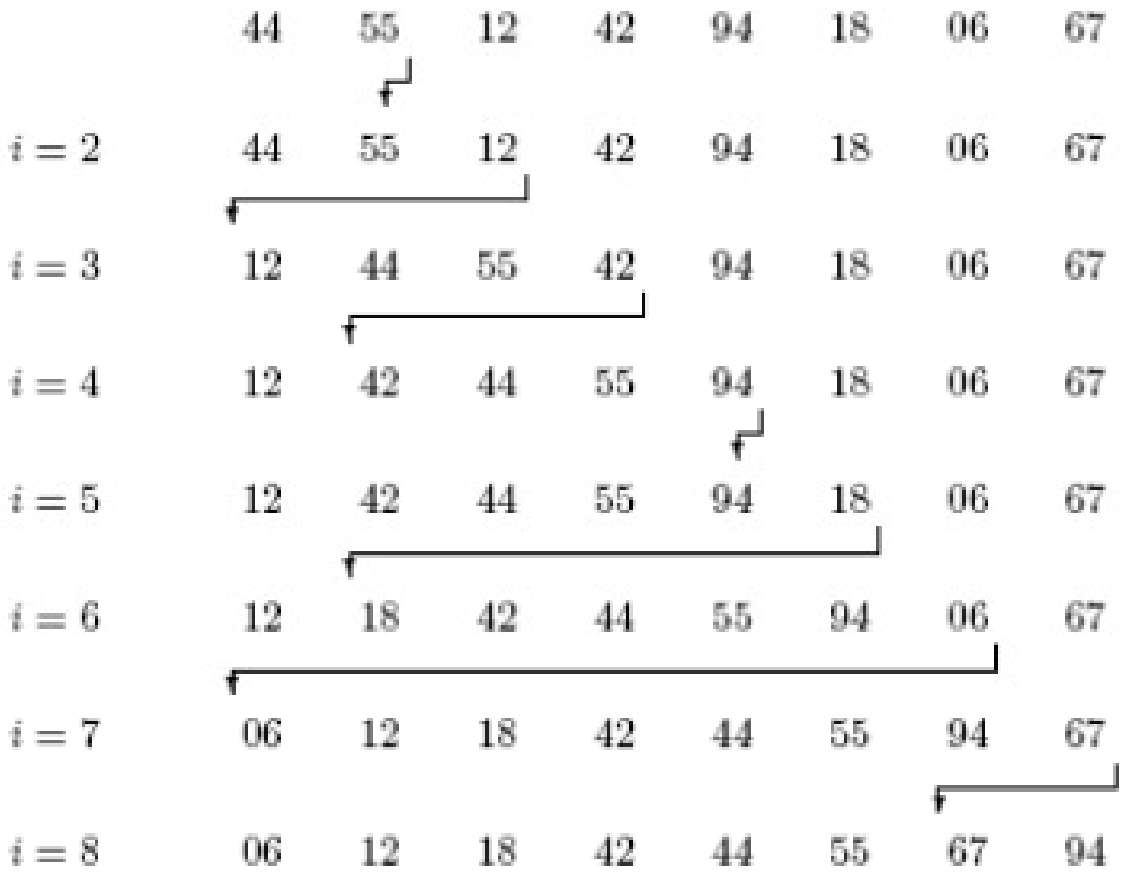
- Имамо уређен почетни део низа $x_{1,...,}x_{i-1}$
- То увек важи за *i=2*
- У сваком кораку, од *i=2,* i-ти елемент се ставља на право место у односу на првих (уређених) *i-1*

- Глобални алгоритам

```
for (i=2; i<=n; i++) {
    v = x[i];
    ”umetnuti” v na odgovarajuće mesto u x1,x2,…, xi
}
```

  - “Уметање” на одговарајуће место: померањем елемената који претходе ($x[j]$, $j = i$-1,...,1) за по једно место удесно ($x[j+1] = x[j]$)
  - Критеријум краја :
    - Нађена вредност x_j *т.д.* $x_j < v$ (v се ставља на j+1. место)
    - Дошло се до левог краја низа

Сортирање уметањем: пример

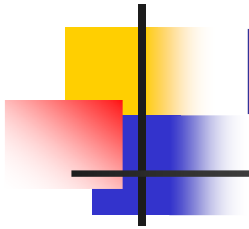


Сортирање уметањем: функција

```
void umetsort(int x[ ], int n)
{
    int v;
    for(i=1; i<n; i++) {
        v=x[i]; j=i–1;
        while (j>=0 && v<x[j]) {
            x[j+1]=x[j]; j--;
        }
        x[j+1]=v;
    }
}
```

Сортирање уметањем: сложеност

- Број поређења, P_i у i–том пролазу је највише i а најмање 1, у просеку је $i/2$.
- Број премештања, M_i је P_i+2
- Укупан број поређења и премештања је
- P_{min} = n-1
- P_{max} = n + (n-1) + (n-2) + … + 1 = n*(n+1)/2 = ½*(n^2+n)
- И слично за M_i
- Најмањи број поређења и премештања је кад је полазни низ уређен (O(n))
- Највећи број поређења и премештања је кад је полазни низ уређен у обрнутом поретку (O(n^2))
- Могућа побољшања алгоритма користе уређеност низа $x_1, x_2, \ldots, x_{i-1}$ за бинарну претрагу – сортирање бинарним уметањем



Мехурасто сортирање

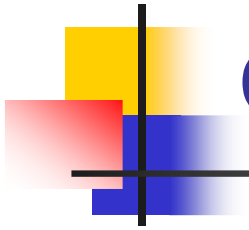
- Поступак:
  - понавља се пролазак кроз низ елемената и размењују се суседни елементи, ако је потребно
  - крај када се заврши пролаз без иједне размене

Мехурасто сортирање

- Функција:

```
void bubblesort(int x[ ], int n)
{
    void swap(int x[ ], int i, int j);
    int j, ind;
    do {
            ind=0;
            for(j=1; j<n; j++)
                    if (x[j–1] > x[j]) {
                            swap(x, j–1, j);
                            ind = 1;
                    }
    } while (ind);
}
```

Мехурасто сортирање: сложеност



- Квадратно време
- Сложенији и мање ефикасан алгоритам од сортирања уметањем
- Унутрашња петља има више исказа и од сортирања уметањем и од сортирања избором