SORTIRANJE NIZOVA

Selection sort

- Jedan od najjednostvnijih algoritama – izbor uzastopnih minimuma

```
void sort(int a[], int n)
{
      int i, j, pom;
      for(i = 0; i < n-1; i++)
          for(j = i+1; j < n; j++)
              if(a[j] < a[i])
                   zameni(a, i, j);
}

void zameni(int a[], int i, int j)
{
      int pom;
      pom = a[i];
      a[i] = a[j];
      a[j] = pom;
}
```

Selection sort

16 23 8 11 5 37 54 62 24 18

5	23	16	11	8	37	54	62	24	18
5	8	23	16	11	37	54	62	24	18
5	8	11	23	16	37	54	62	24	18
5	8	11	16	23	37	54	62	24	18
5	8	11	16	18	37	54	62	24	23
5	8	11	16	18	23	54	62	37	24
5	8	11	16	18	23	24	62	54	37
5	8	11	16	18	23	24	37	62	54
5	8	11	16	18	23	24	37	54	62

Selection sort

- Ako se za meru kompleksnosti algoritma uzme broj poređenja dva broja – po jedno poređenje u svakoj unutrašnjoj petlji za svako $j = i + 1, \ldots, n - 1$

$$
\begin{aligned}
f_1(n) &= \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 \\
&= \sum_{i=0}^{n-2} n - i - 1 \\
&= (n-1) + (n-2) + \cdots + 1 \\
&= \frac{n(n-1)}{2}
\end{aligned}
$$

- Broj poređenja je $O(n^2)$, nezavisno od ulaznog niza
- Ako se za jedinicu mere uzme broj izvršenih zamena, u nekim slučajevima će biti izvršeno $\frac{n(n-1)}{2}$ zamena, a u drugim slučajevima neće biti zamena
- U proseku broj zamena će biti $O(n^2)$

Quick sort

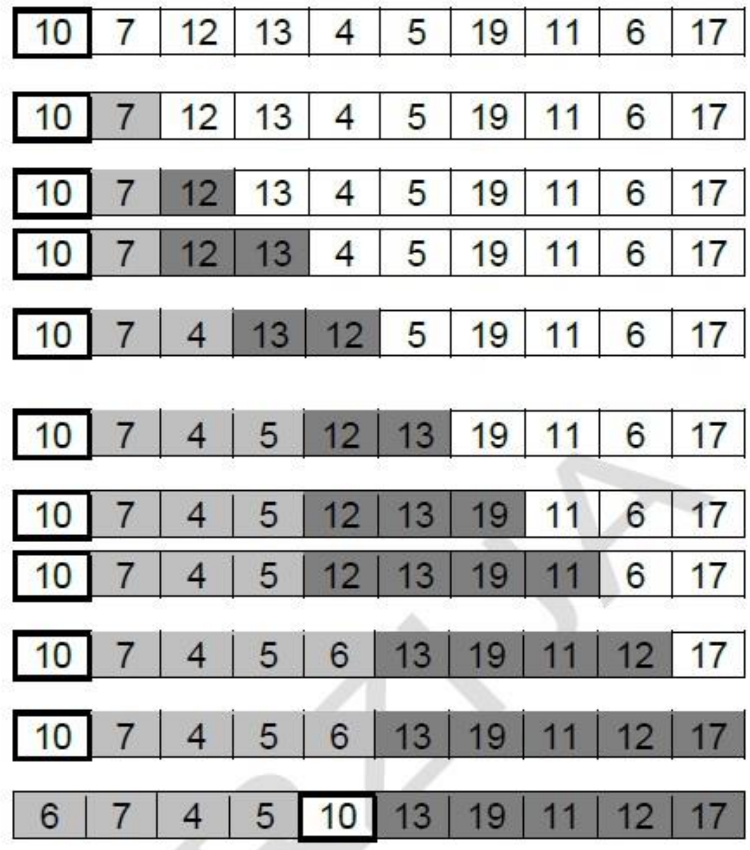
- Algoritam je baziran na strategiji „podeli i osvoj“

- Algoritam se odvija u dva dela
  - Elementi niza se raspoređuju tako da u odnosu na izabrani element - pivot, svi elemeti manji od njega budu sa njegove leve strane, a svi veći od njega budu sa njegove desne strane
  - Zatim se odvojeno sortiraju levi i desni podniz u odnosu na pivot

- Pivot može biti bilo koji element niza, često se zbog jednostavnosti izabere prvi element

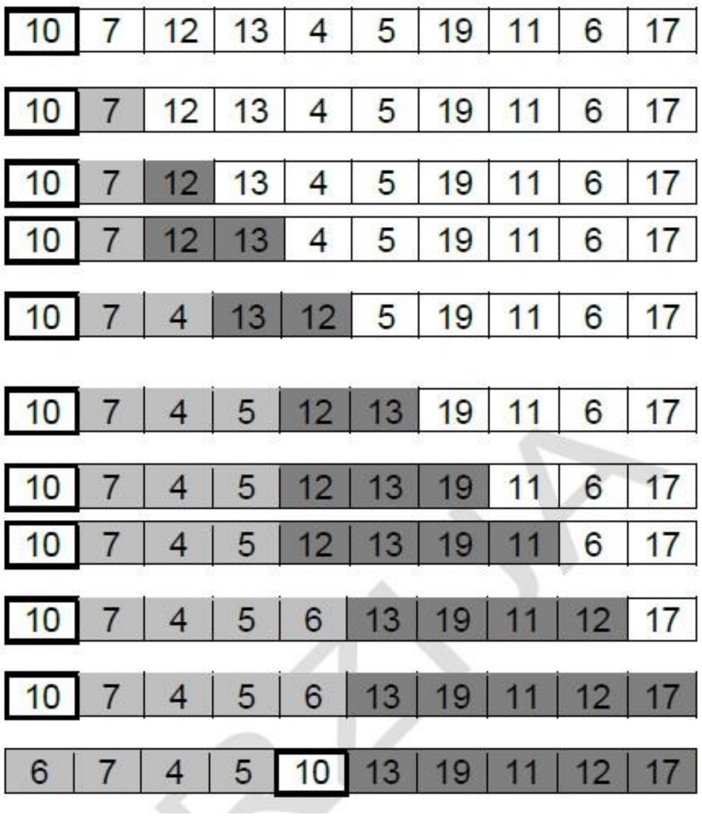
Razvrstavanje elemenata

1. Neka je prvi element niza izabran za pivot
2. Svi elementi niza se proveravaju da li su manji ili veći od pivota
3. Ukoliko je posmatrani element
   a) veći od pivota, nema potrebe za premeštanjem.
   b) manji od pivota, prebacuje se u grupu elemenata koji su manji od pivota, a pozicija pivota se uvećava za jedan
4. Nakon prolaska kroz čitav niz pivot se postavlja na određenu poziciju



Razvrstavanje elemenata

```
int podeli(int a[], int donji, int gornji)
{
      int i, granica;
      granica = donji;
      for(i = donji + 1; i <= gornji; i++)
          if(a[i] < a[donji])
          {
              granica++;
              zameni(a, i, granica);
          }
      zameni(a, donji, granica);
      return(granica);
}
```



Quick

```
void quick(int a[], int donji, int gornji)
{
      int granica;
      if(donji < gornji)
      {
          granica = podeli(a, donji, gornji);
          quick(a, donji, granica-1);
          quick(a, granica+1, gornji);
      }
}

void sort(int a[], int n)
{
      quick(a, 0, n-1);
}
```

Quicksort - kompleksnost

- Jedinica mere kompleksnosti algoritma – broj poređenja dva broja
- Ako pri svakom izboru za pivota bude izabran najmanji (najveći) broj, tada se pri sortiranju niza dužine *k*, sortiraju podnizovi dužine *k*-1 i 0
- Za niz dužine n, broj poređenja *L(n)*, pri svim „lošim" izborima pivota, biće

$$L(n) = L(n-1) + n - 1, n \geq 1,\ L(0) = 0$$

gde je $L(n-1) = L(n-2) + n - 2, \quad L(n-2) = L(n-3) + n - 3, \ldots$

tj.

$$L(n) = 1 + 2 + \cdots + (n-1) = \frac{n(n-1)}{2}$$

- U najgorem slučaju, *quicksort* ima isti broj poređenja kao *selection* sort, $O(n^2)$

Quicksort - kompleksnost

- U opštem slučaju, ako su svi elementi niza različiti, postoji $n!$ mogućih ulaza
- $F(n)$ prosečana broj poređenja u qucksort-u i sadrži dve komponente
  - Poređenja pri postavljanu pivota – u nizu od *n* elemenata *n*-1 poređenje
  - Poređenja u rekurzivnim pozivima
- Ako je pivot na poziciji *i*, svaka vrednost za *i* je jednako verovatna $\left(p = \frac{1}{n}\right)$, a rekurzivni pozivi se odnose na nizove dužine *i* i *n*-*i*-1

$$F(n) = n - 1 + \frac{1}{n}\sum_{i=0}^{n-1}\big(F(i) + F(n-i-1)\big),\ n \geq 1,\ F(0) = 0$$

- Kako je

$$\begin{aligned}\sum_{i=0}^{n-1} F(n-i-1) &= F(n-1) + F(n-2) + \cdots + F(0) \\ &= \sum_{i=0}^{n-1} F(i)\end{aligned}$$

- Dobija se

$$F(n) = n - 1 + \frac{2}{n}\sum_{i=0}^{n-1} F(i)$$

Quicksort - kompleksnost

- Množenjem prethodne jednakosti sa *n* dobija se

$$nF(n) = n(n-1) + 2\sum_{i=0}^{n-1} F(i)$$

- Zamenom *n* sa *n*-1 u poslednjoj jednakosti dobija se

$$(n-1)F(n-1) = (n-1)(n-2) + 2\sum_{i=0}^{n-2} F(i)$$

- Oduzimanjem prethodne dve jednakosti, dobija se

$$nF(n) - (n-1)F(n-1) = n(n-1) - (n-1)(n-2) + 2F(n-1)$$

tj.

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

Quicksort - kompleksnost

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

- Uvođenjem zamene

$$F(n) = \frac{n+1}{n} \cdot \frac{n}{n-1} \cdot \frac{n-1}{n-2} \cdots \frac{2}{1} y_n = (n+1) y_n$$

- Dobija se

$$(n+1) y_n = \frac{n+1}{n} n y_{n-1} + 2\frac{n-1}{n}$$

$$y_n = y_{n-1} + 2\frac{n-1}{n(n+1)}\ ,\ n \geq 1,\ \ y_0 = 0$$

- Rešavanjem rekuretne formule dobija se

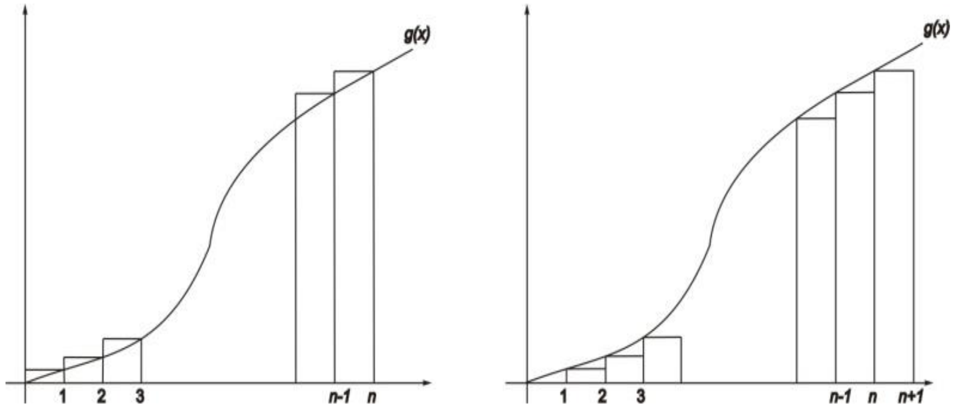
$$\begin{aligned} y_n &= 2\sum_{j=1}^{n} \frac{j-1}{j(j+1)} = 2\sum_{j=1}^{n} \left(\frac{2}{j+1} - \frac{1}{j}\right) \\ &= 2\sum_{j=1}^{n} \frac{1}{j} - \frac{4n}{n+1} \end{aligned}$$

Quicksort - kompleksnost

- Konačno se odbija

$$F(n) = 2(n+1)\sum_{j=1}^{n}\frac{1}{j} - 4n$$

- Koristeći relaciju



$$\int_0^n g(t)dt \le \sum_{j=1}^{n} g(j) \le \int_1^{n+1} g(t)dt \ \text{ za } \ g(t) = \frac{1}{t}$$

Konačno se dobija

$$F(n) \sim 2n \ln n \quad (n \to \infty)$$

Odnos brzina rasta funkcija