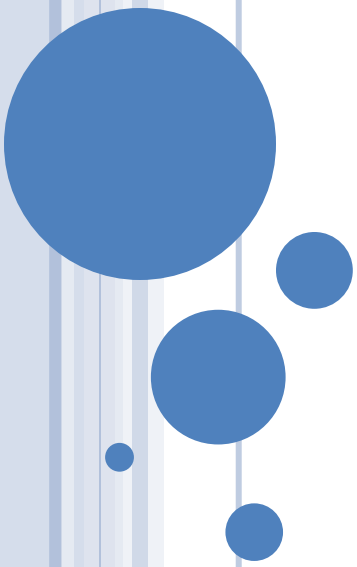


Triedenia

Bublinové, quicksort, mergesort, výberom, haldové

Výpočtová zložitosť problému

Prečo triedenia?

- problém usporiadať postupnosť hodnôt sa dá riešiť mnohými algoritmami s rôznou výpočtovou zložitosťou
- triediace algoritmy sú príkladom rôznych programovacích techník

Definujme problém

Je daná postupnosť n čísiel uložených v poli a. Usporiadajme čísla v poli od najmenšieho po najväčšie.

Poznámky k definícii problému:

- *triediť môžeme nielen čísla, ale hodnoty z ľubovoľnej množiny, na ktorej je definovaná relácia usporiadania*
- *triediť môžeme vzostupne, ale aj zostupne*
- *triedená postupnosť môže byť uložená v poli, ale aj v súbore*

Triedenie výmenami – bublinové

- porovnávajú sa a v prípade potreby vymieňajú **susedné** prvky v poli

2	12	67	5	9	7	398	16	354	90	34	11	14	43	1
2	12	5	9	7	67	16	354	90	34	11	14	43	1	398
2	5	9	7	12	16	67	90	34	11	14	43	1	354	398
2	5	7	9	12	16	67	34	11	14	43	1	90	354	398
2	5	7	9	12	16	34	11	14	43	1	67	90	354	398
2	5	7	9	12	16	11	14	34	1	43	67	90	354	398
2	5	7	9	12	11	14	16	1	34	43	67	90	354	398
2	5	7	9	11	12	14	1	16	34	43	67	90	354	398
2	5	7	9	11	12	1	14	16	34	43	67	90	354	398
2	5	7	9	11	1	12	14	16	34	43	67	90	354	398
2	5	7	9	1	11	12	14	16	34	43	67	90	354	398
2	5	7	1	9	11	12	14	16	34	43	67	90	354	398
2	5	1	7	9	11	12	14	16	34	43	67	90	354	398
2	1	5	7	9	11	12	14	16	34	43	67	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398

Bublinové triedenie – rôzne varianty

- maximum sa presúva dozadu

```
for (int i = n - 1; i > 0; i--) {
   for (int j = 0; j < i; j++) {
      if (a[j] > a[j+1]) { výmena (a[j], a[j+1]) }
   }
}
```

2	12	67	5	9	7	398	16	354	90	34	11	14	43	1
2	12	5	9	7	67	16	354	90	34	11	14	43	1	398
2	5	9	7	12	16	67	90	34	11	14	43	1	354	398
2	5	7	9	12	16	67	34	11	14	43	1	90	354	398
2	5	7	9	12	16	34	11	14	43	1	67	90	354	398
2	5	7	9	12	16	11	14	34	1	43	67	90	354	398
2	5	7	9	12	11	14	16	1	34	43	67	90	354	398
2	5	7	9	11	12	14	1	16	34	43	67	90	354	398
2	5	7	9	11	12	1	14	16	34	43	67	90	354	398
2	5	7	9	11	1	12	14	16	34	43	67	90	354	398
2	5	7	9	1	11	12	14	16	34	43	67	90	354	398
2	5	7	1	9	11	12	14	16	34	43	67	90	354	398
2	5	1	7	9	11	12	14	16	34	43	67	90	354	398
2	1	5	7	9	11	12	14	16	34	43	67	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398

Bublinové triedenie – rôzne varianty

- minimum sa presúva dopredu

```
for (int i = 0; i < n - 1; i++) {
   for (int j = n - 1; j > i; j--) {
      if (a[j-1] > a[j]) { výmena (a[j-1], a[j]) }
   }
}
```

2	12	67	5	9	7	398	16	354	90	34	11	14	43	1
1	2	12	67	5	9	7	398	16	354	90	34	11	14	43
1	2	5	12	67	7	9	11	398	16	354	90	34	14	43
1	2	5	7	12	67	9	11	14	398	16	354	90	34	43
1	2	5	7	9	12	67	11	14	16	398	34	354	90	43
1	2	5	7	9	11	12	67	14	16	34	398	43	354	90
1	2	5	7	9	11	12	14	67	16	34	43	398	90	354
1	2	5	7	9	11	12	14	16	67	34	43	90	398	354
1	2	5	7	9	11	12	14	16	34	67	43	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398

Zníženie počtu porovnaní v bublinovom triedení

- mení sa smer porovnávania (pretriasanie – shakesort)

2	12	67	5	9	7	398	16	354	90	34	11	14	43	1
2	12	5	9	7	67	16	354	90	34	11	14	43	1	398
1	2	12	5	9	7	67	16	354	90	34	11	14	43	398
1	2	5	9	7	12	16	67	90	34	11	14	43	354	398
1	2	5	7	9	11	12	16	67	90	34	14	43	354	398
1	2	5	7	9	11	12	16	67	34	14	43	90	354	398
1	2	5	7	9	11	12	14	16	67	34	43	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398
1	2	5	7	9	11	12	14	16	34	43	67	90	354	398

- sleduje sa, či sa uskutočnila výmena, ak nie, cyklus sa ukončí
- zapamätá sa index poslednej výmeny, porovnáva sa len po tento index
- nemá vplyv na počet výmen

Počet výmen v bublinovom triedení

- **inverzia** je dvojica $a[i]$, $a[j]$ taká, že $i<j$ a zároveň $a[i]>a[j]$
- pri výmene dvoch susedných prvkov sa počet inverzií zmenší práve o 1

počet inverzií pre jednotlivé prvky triedenej postupnosti:

2	12	67	5	9	7	398	16	354	90	34	11	14	43	1
0	0	0	2	2	3	0	2	1	2	4	7	6	4	14

Počet výmen v bublinovom triedení

- počet výmen v bublinovom triedení sa rovná počtu inverzií vo vstupnej postupnosti (jedna výmena odstráni jednu inverziu)

3	1	4	5	9	10	11	6	2	14	15	7	8	12	13	
0	1	0	0	0	0	0	3	7	0	0	5	5	2	2	25
1	3	4	5	9	10	6	2	11	14	7	8	12	13	15	
1	3	4	5	9	6	2	10	11	7	8	12	13	14	15	
1	3	4	5	6	2	9	10	7	8	11	12	13	14	15	
1	3	4	5	2	6	9	7	8	10	11	12	13	14	15	
1	3	4	2	5	6	7	8	9	10	11	12	13	14	15	
1	3	2	4	5	6	7	8	9	10	11	12	13	14	15	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

Časová výpočtová zložitosť bublinového triedenia

- počet inverzií v najhoršom prípade:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
14	13	12	11	10	9	8	7	6	5	4	3	2	1	15
13	12	11	10	9	8	7	6	5	4	3	2	1	14	15
12	11	10	9	8	7	6	5	4	3	2	1	13	14	15
11	10	9	8	7	6	5	4	3	2	1	12	13	14	15
10	9	8	7	6	5	4	3	2	1	11	12	13	14	15
9	8	7	6	5	4	3	2	1	10	11	12	13	14	15
8	7	6	5	4	3	2	1	9	10	11	12	13	14	15
7	6	5	4	3	2	1	8	9	10	11	12	13	14	15
6	5	4	3	2	1	7	8	9	10	11	12	13	14	15
5	4	3	2	1	6	7	8	9	10	11	12	13	14	15
4	3	2	1	5	6	7	8	9	10	11	12	13	14	15
3	2	1	4	5	6	7	8	9	10	11	12	13	14	15
2	1	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$$T(n) = 0 + 1 + \ldots + n - 1 = \frac{(n-1).n}{2} = O(n^2)$$

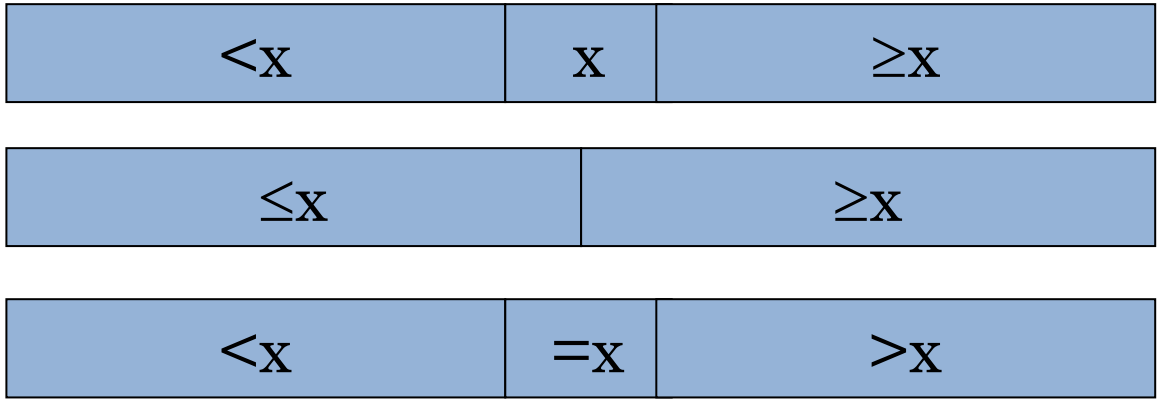
QUICKSORT

- porovnávajú a vymieňajú sa prvky na väčšie vzdialenosti
- pri každej výmene sa odstráni **aspoň jedna** inverzia,
- počet výmen ≤ počet inverzií
- algoritmus **„rozdeľuj a panuj“**

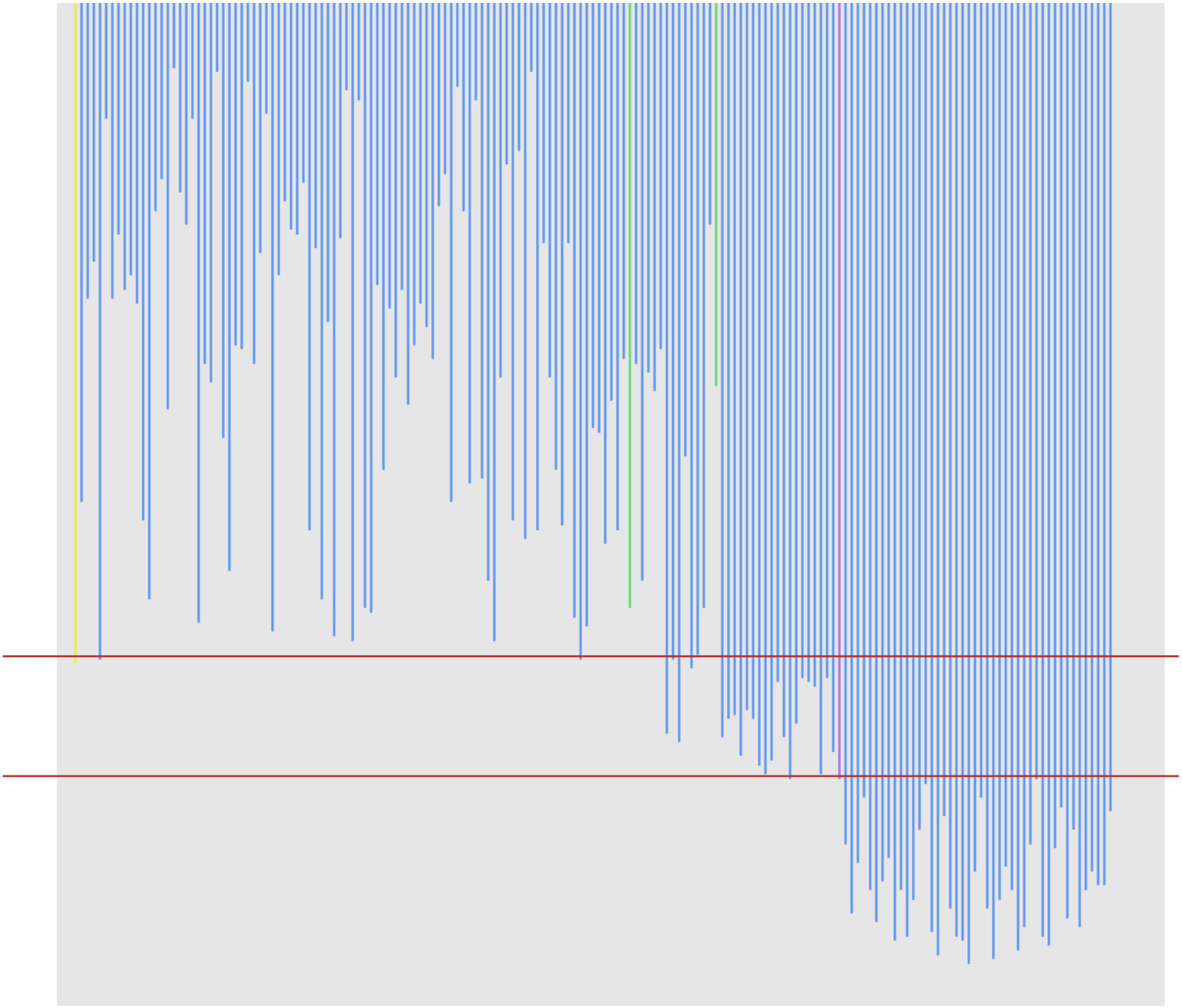
2	12	67	5	9	7	398	16	354	90	34	11	14	43	1
0	0	0	2	2	3	0	2	1	2	4	7	6	4	14

QUICKSORT – RÔZNE VARIANTY

1. výber pivota x
   - *prvý prvok*
   - *prostredný prvok*
   - *náhodný prvok*
2. rozdelenie poľa podľa pivota x



3. rekurzívne triedenie rozdelených častí poľa



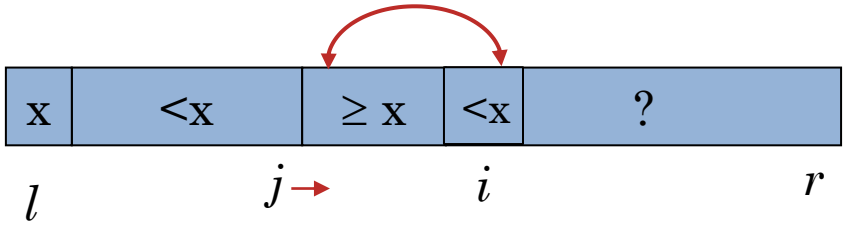
QUICKSORT – ROZDELENIE (BENTLEY)

vstup

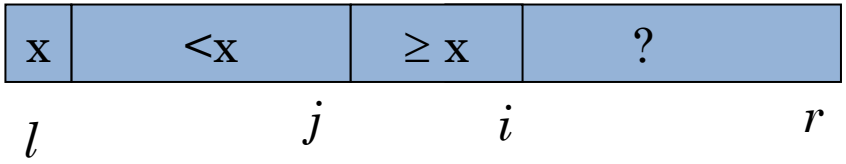
l=j

r

```
x = a[l];
j = l;
for (i = l+1; i <= r; i++) {
   if (a[i] < x) {
      j++;
      výmena a[i], a[j];
   }
}
výmena a[l], a[j];
```



invariant



výstup

l

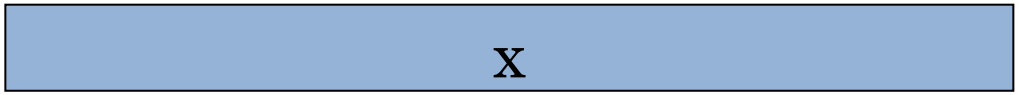
j

r=i

QUICKSORT – TRIEDENIE

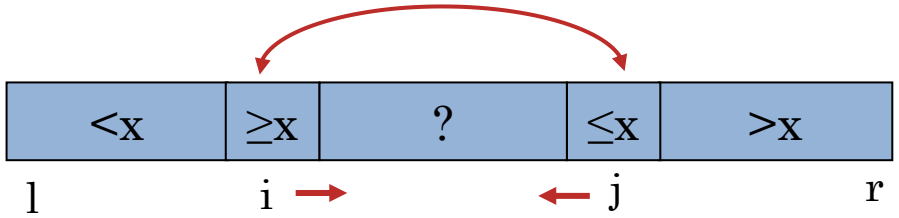
```
void quicksort(int a[], int l, int r)
{
  if  (l >= r) return;

  //rozdelenie

  int i, j;
  x = a[l];
  j = l;
  for (i = l+1; i <= r; i++) {
      if (a[i] < x) {
         j++;
         výmena(i,j);
      }
  }
  výmena(l,j);

  quicksort(l,j-1);
  quicksort(j+1,r);
}
```

QUICKSORT – ROZDELENIE 2 (WIRTH)

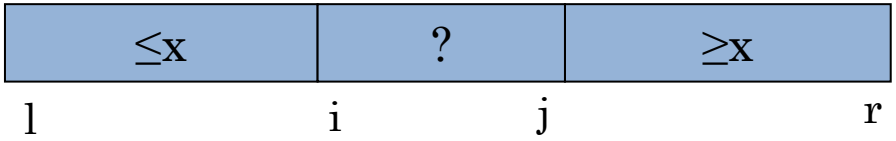


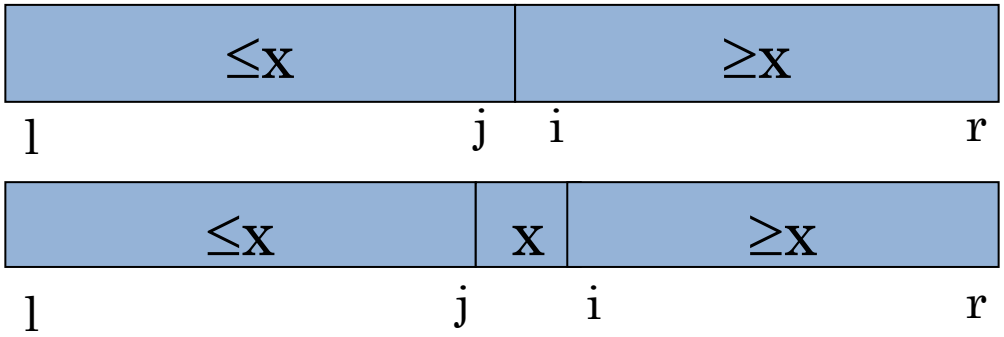
l=i r=j

```
int i = l;
int j = r;
int x = a[(l+r) /2];
do {
    while (a[i] < x) { i++; }
    while (x < a[j]) { j--; }
    if (i ≤ j) {
        výmena(a[i], a[j]);
        i++; j--;
} while (i <= j);
```



invariant:





alebo

QUICKSORT – TRIEDENIE

```
void quicksort(int a[], int l, int r)
{
  if  (l >= r) return;

  //rozdelenie

  int i = l;
  int j = r;
  int x = a[(l+r)/2];
  do {
     while (a[i] < x) { i++; }
     while (x < a[j]) { j--; }
     if (i ≤ j) {
        výmena(a[i], a[j]);
        i++; j--;
  } while (i <= j);


  quicksort(l,j);
  quicksort(i,r);
}
```

Príklad výpočtu podľa Wirthovho algoritmu

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
3	**1**	**4**	**5**	**9**	**10**	**11**	**6**	**2**	**14**	**15**	**7**	**8**	**12**	**13**
3	1	4	5	2	6	11	10	9	14	15	7	8	12	13
3	1	2	5	4	6	11	10	9	14	13	7	8	12	15
1	3	2	4	5	6	11	10	9	12	8	7	13	14	15
1	2	3	4	5	6	11	10	9	7	8	12	13	14	15
1	2	3	4	5	6	8	7	9	10	11	12	13	14	15
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

- počet výmen je 17

Špeciálne prípady vstupov – najhorší prípad

1	4	6	8	10	12	14	15	2	9	5	11	3	13	7
1	4	6	8	10	12	14	7	2	9	5	11	3	13	15
1	4	6	8	10	12	13	7	2	9	5	11	3	14	15
1	4	6	8	10	12	3	7	2	9	5	11	13	14	15
1	4	6	8	10	11	3	7	2	9	5	12	13	14	15
1	4	6	8	10	5	3	7	2	9	11	12	13	14	15
1	4	6	8	9	5	3	7	2	10	11	12	13	14	15
1	4	6	8	2	5	3	7	9	10	11	12	13	14	15
1	4	6	7	2	5	3	8	9	10	11	12	13	14	15
1	4	6	3	2	5	7	8	9	10	11	12	13	14	15
1	4	5	3	2	6	7	8	9	10	11	12	13	14	15
1	4	2	3	5	6	7	8	9	10	11	12	13	14	15
1	3	2	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Špeciálne prípady vstupov – najlepší prípad

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

- počet porovnaní je $O(n \cdot \log_2 n)$

Výpočtová zložitosť Quicksortu

- najhorší prípad – keď sa za pivota vyberie v každom rozdeľovaní minimum alebo maximum
  - $T(n) = O(n^2)$
- najlepší prípad – keď sa za pivota vyberie v každom rozdeľovaní medián (stredná hodnota)
  - $T(n) = O(n \log n)$
- priemerný prípad
  - $T(n) = O(n \log n)$
- pamäťová výpočtová zložitosť
  - $S(n) = O(n+\log n)$
  - triedi sa na mieste $O(n)$ a pri rekurzii sa obsadí zásobník veľkosti priemerne $O(\log n)$