STU FEI

Programovacie techniky

3. Zložitosť, O notácia, bubble sort, insertion sort, quick sort, merge sort

Triedenie

Usporiadanie napr. podľa veľkosti

1, 77, 89, 102, 206, 234

„Martin“ < „Alexander“

alebo

„Martin“ > „Alexander“

(M, r): usporiadaj množinu M vzhľadom na reláciu r

Triedenie

- Jednorázové triedenie veľkého množstva údajov
- Utrieďovanie „položka po položke“, okamžite po vzniku daného údaja

Kritéria

Časová zložitosť (time complexity):

- Počet porovnaní
- Počet výmien (swapov)

Priestorová zložitosť (space complexity):

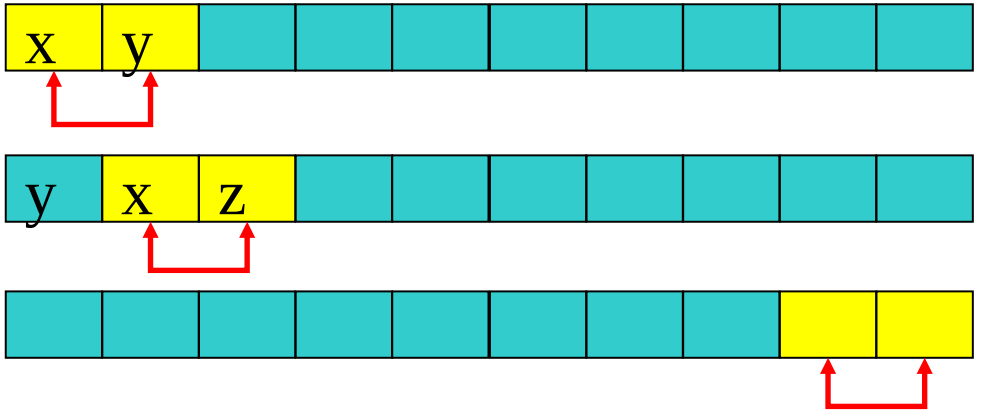
- Koľko pamäte potrebujeme?
- V tom istom poli (in-place), alebo je potrebné ďalšie pomocné pole?

Bubble sort

Idea: pri prechode poľom porovnať dva za sebou idúce prvky poľa. Ak sú v nesprávnom poradí, tak sú vymenené.

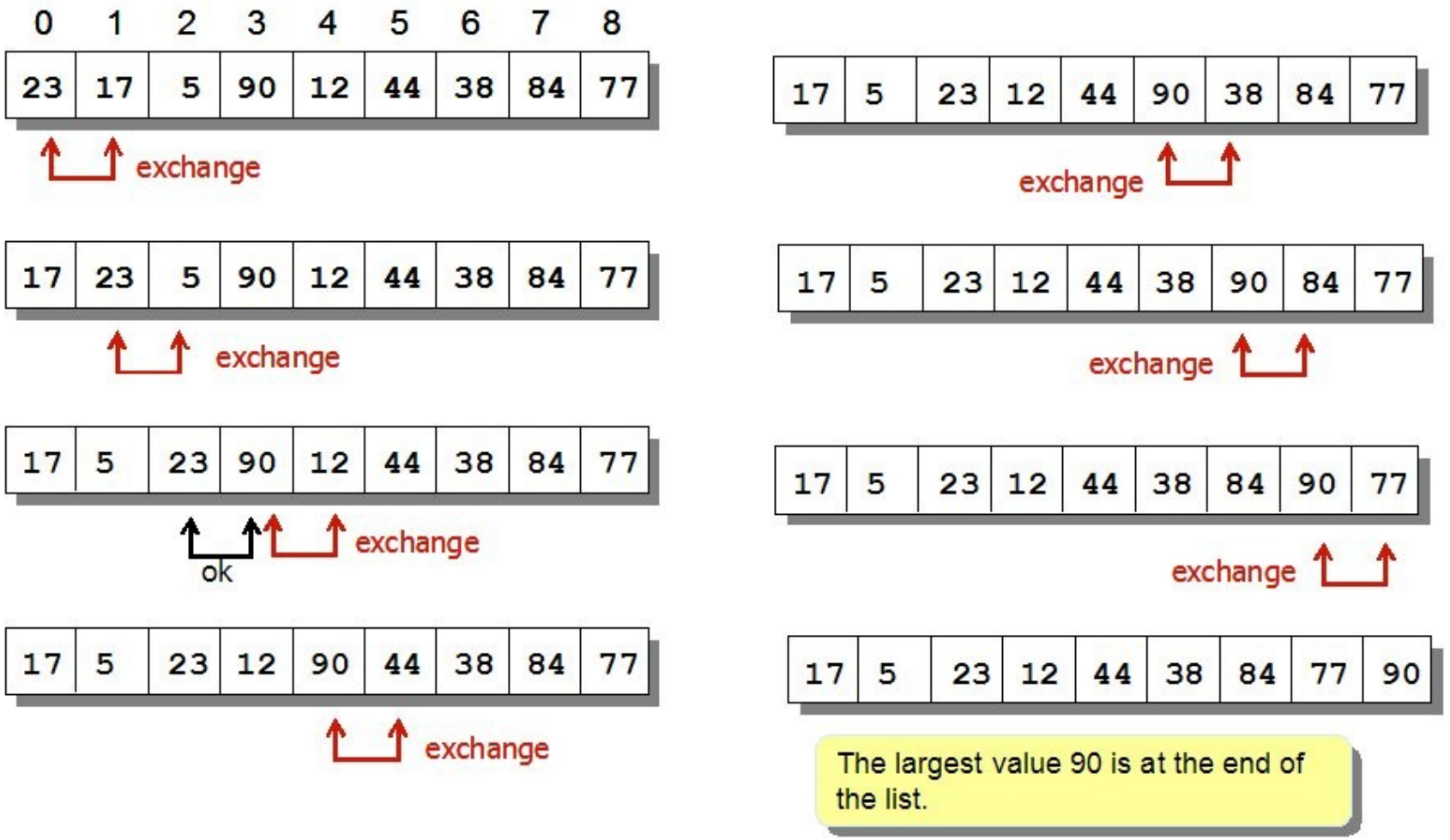
ak x > y

...



Opakovať pokým neutriedime celé pole.

Bubble sort


0 1 2 3 4 5 6 7 8
23 17 5 90 12 44 38 84 77
exchange
17 23 5 90 12 44 38 84 77
exchange
17 5 23 90 12 44 38 84 77
ok
exchange
17 5 23 12 90 44 38 84 77
exchange
17 5 23 12 44 90 38 84 77
exchange
17 5 23 12 44 38 90 84 77
exchange
17 5 23 12 44 38 84 90 77
exchange
17 5 23 12 44 38 84 77 90
The largest value 90 is at the end of the list.

Bubble sort

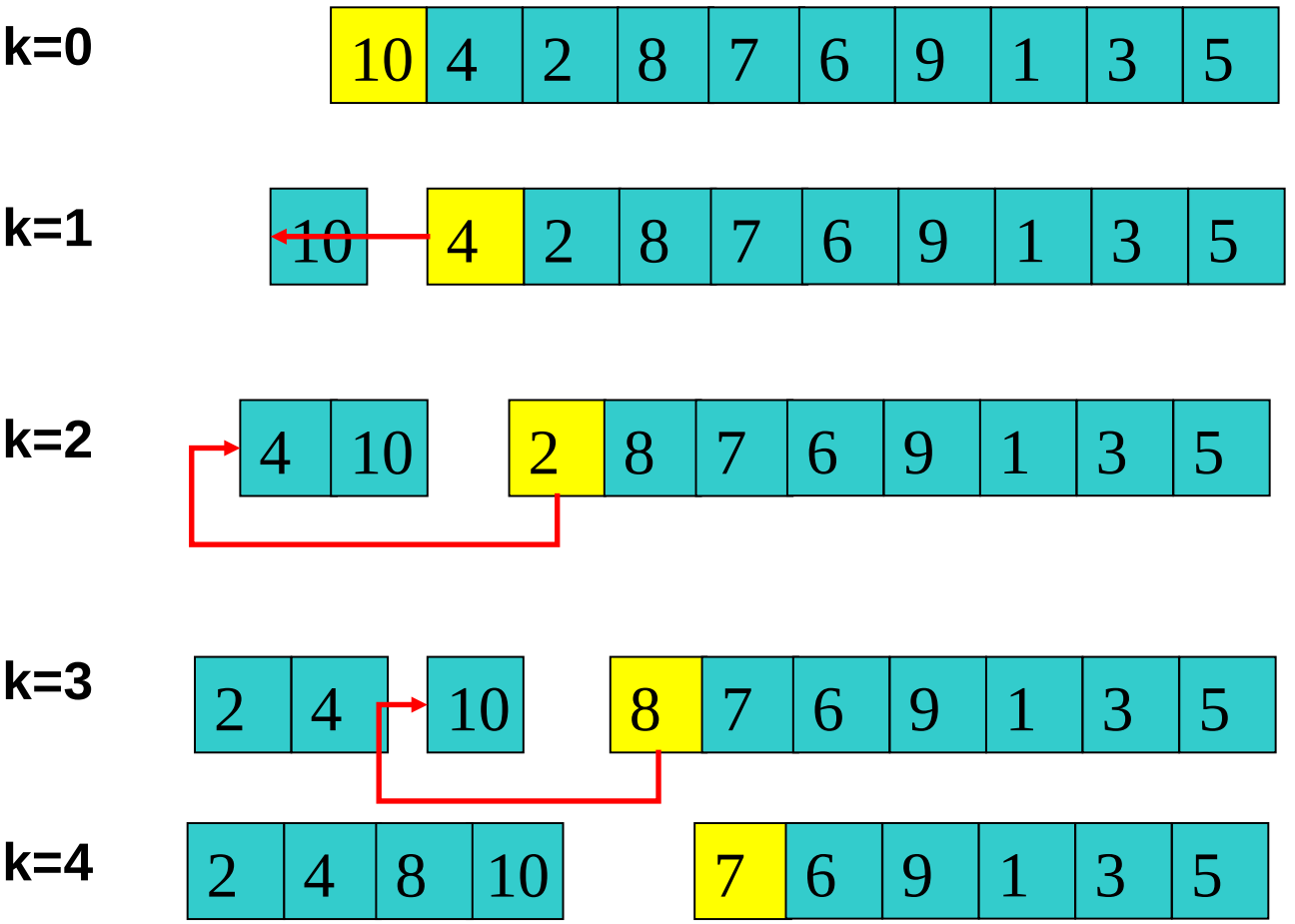
```
void prechod (int *pole, int N) {
    int j;
    for(j=0; j < N-1; j++){              //nie N prečo?
      if(pole[j] < pole[j+1]) {          //porovnanie
        vymen(pole+j, pole+j+1); //výmena
      }
    }
}
```

Insertion sort

V k-tom kroku vybrať k-ty prvok poľa a uložiť ho na svoju pozíciu medzi prvky s indexmi 0,...,k-1.

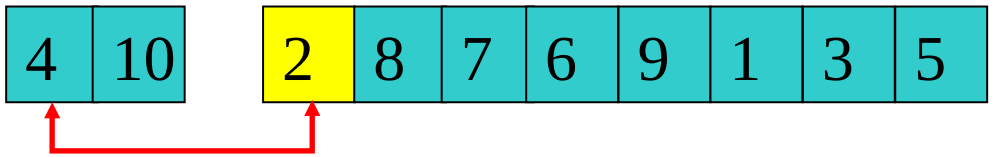
Vkladanie kariet – do už utriedených kariet vlož ďalšiu.

Insertion sort: pole

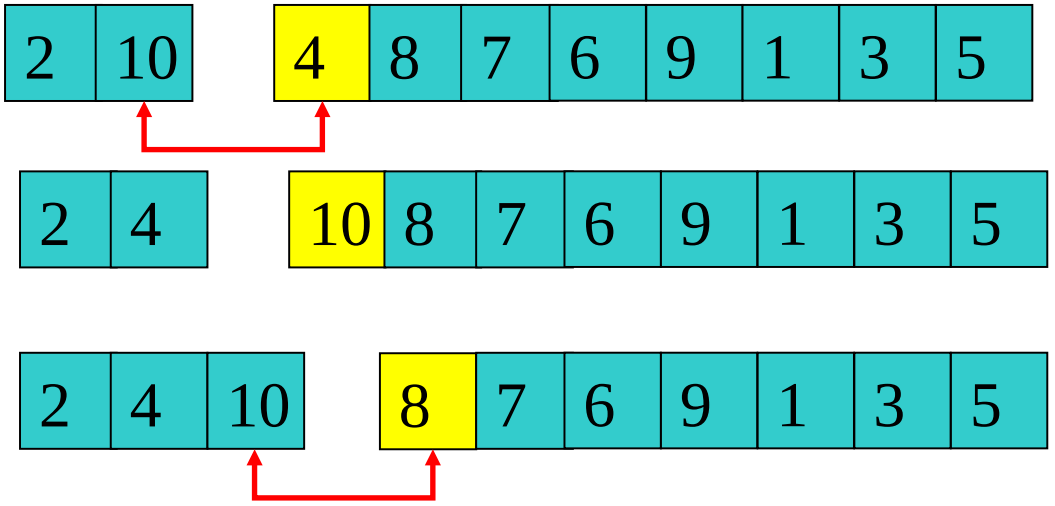


Insertion sort: pomocou výmen

Vloženie na správne miesto možno realizovať postupnými výmenami.



Realizácia umiestenia pomocou výmen:



Insertion sort, bubble sort

Bubble sort: $O(n^2)$

Vonkajší cyklus: (n-1)-krát

Vnútorný cyklus: (n-1) + (n-2) + (n-3) + ... + 1

Insertion sort: $O(n^2)$

Vonkajší cyklus: ?

Vnútorný cyklus: ?

O-notácia zanedbáva konštanty: $3n^2 = O(n^2)$

$n^2 + 100 = O(n^2)$

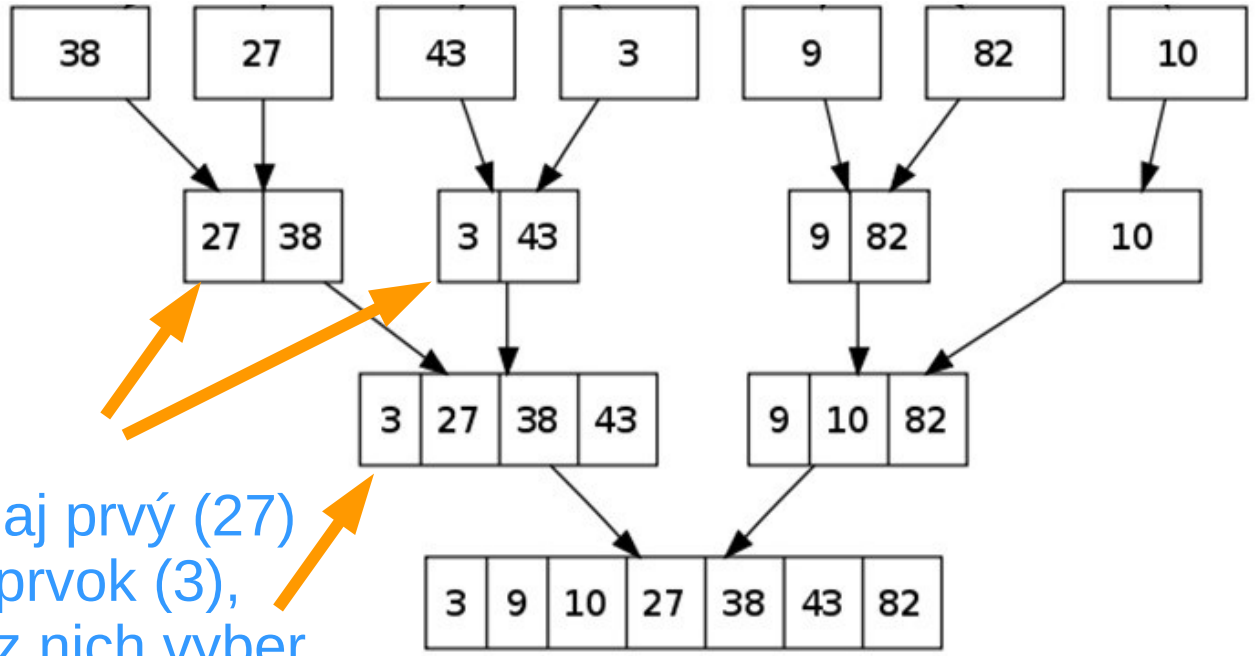
$n^2 / 100 = O(n^2)$

Merge sort

Idea: rozdelím pole na 2 časti. Tie utriedim a spojím.

Ak časť pola neviem roztriediť, pole opätovne rozdelím na 2 časti.

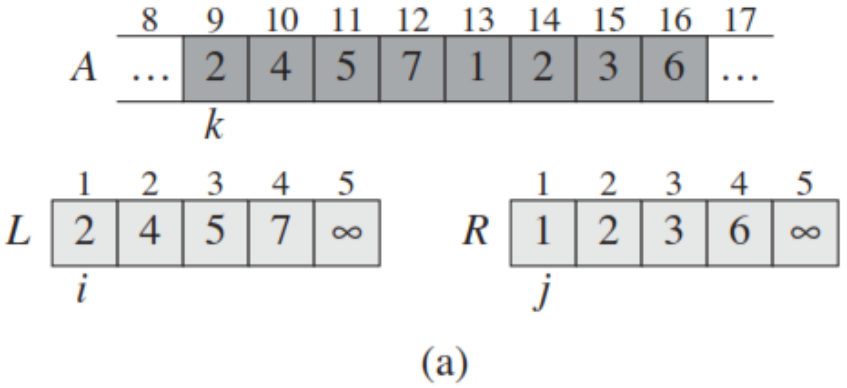
Merge sort: spájanie



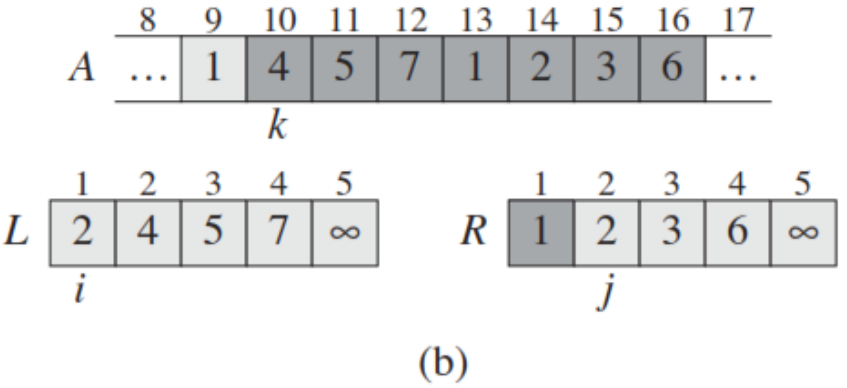
Porovnaj prvý (27) a prvý prvok (3), menší z nich vyber (3) a vlož do spojeného poľa

Obrázok zdroj: http://anandgrathod.blogspot.sk/2013/02/merge-sort.html

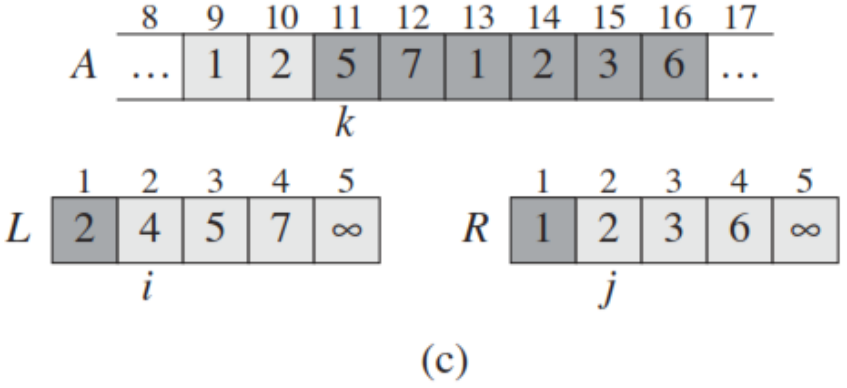
Merge sort: príklad merge



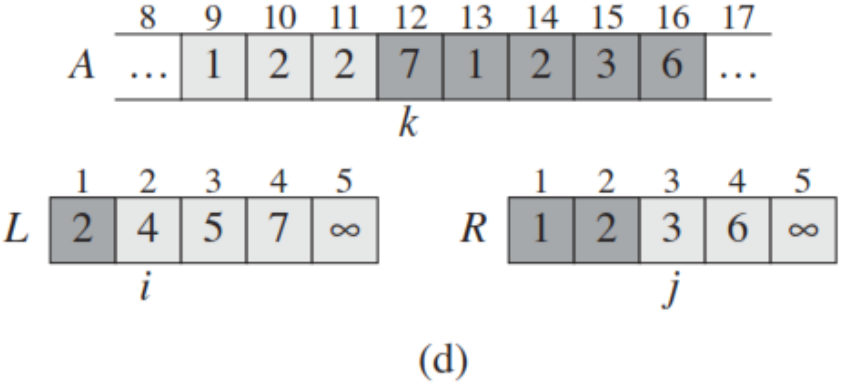
(a)



(b)



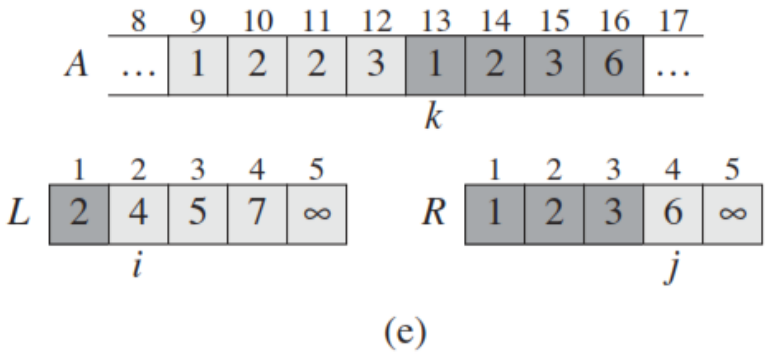
(c)



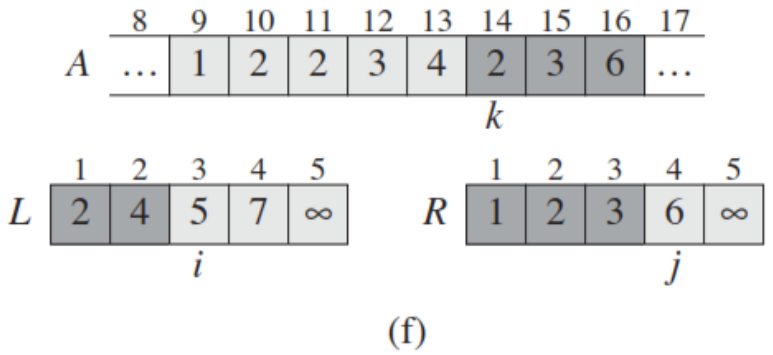
(d)

Merge sort je stabilný triediaci algoritmus

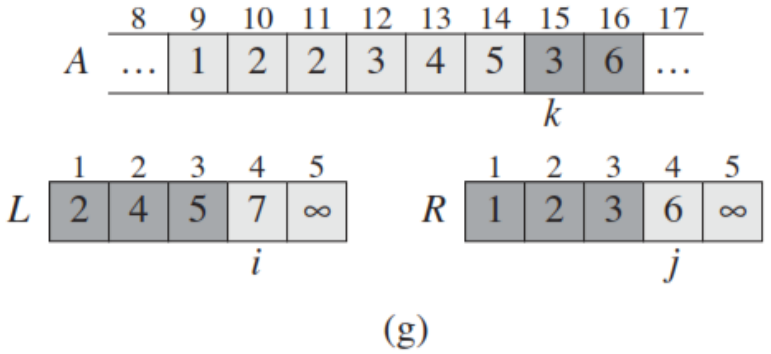
Merge sort: príklad merge



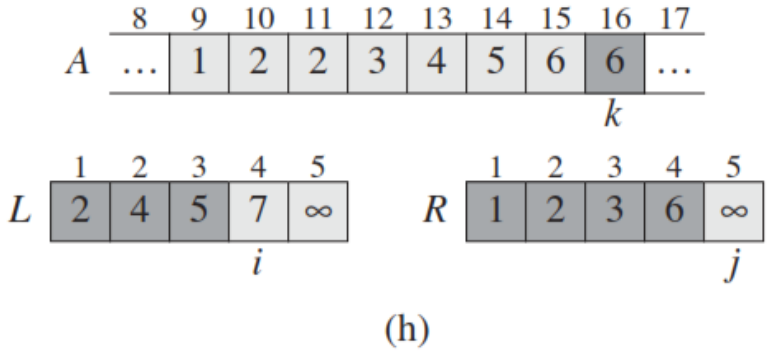
(e)



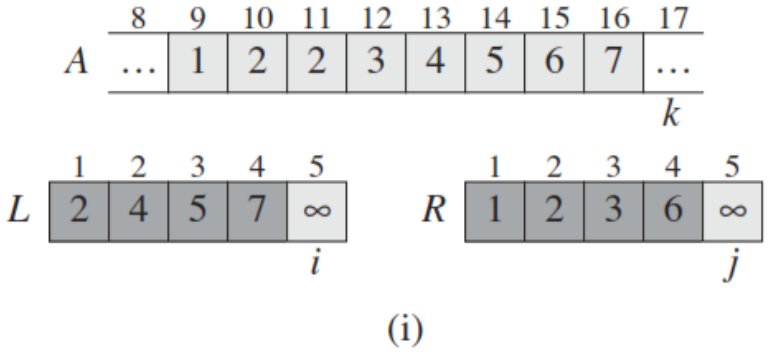
(f)



(g)



(h)



(i)

Merge: pseudokód

A je pole A[p..q..r]

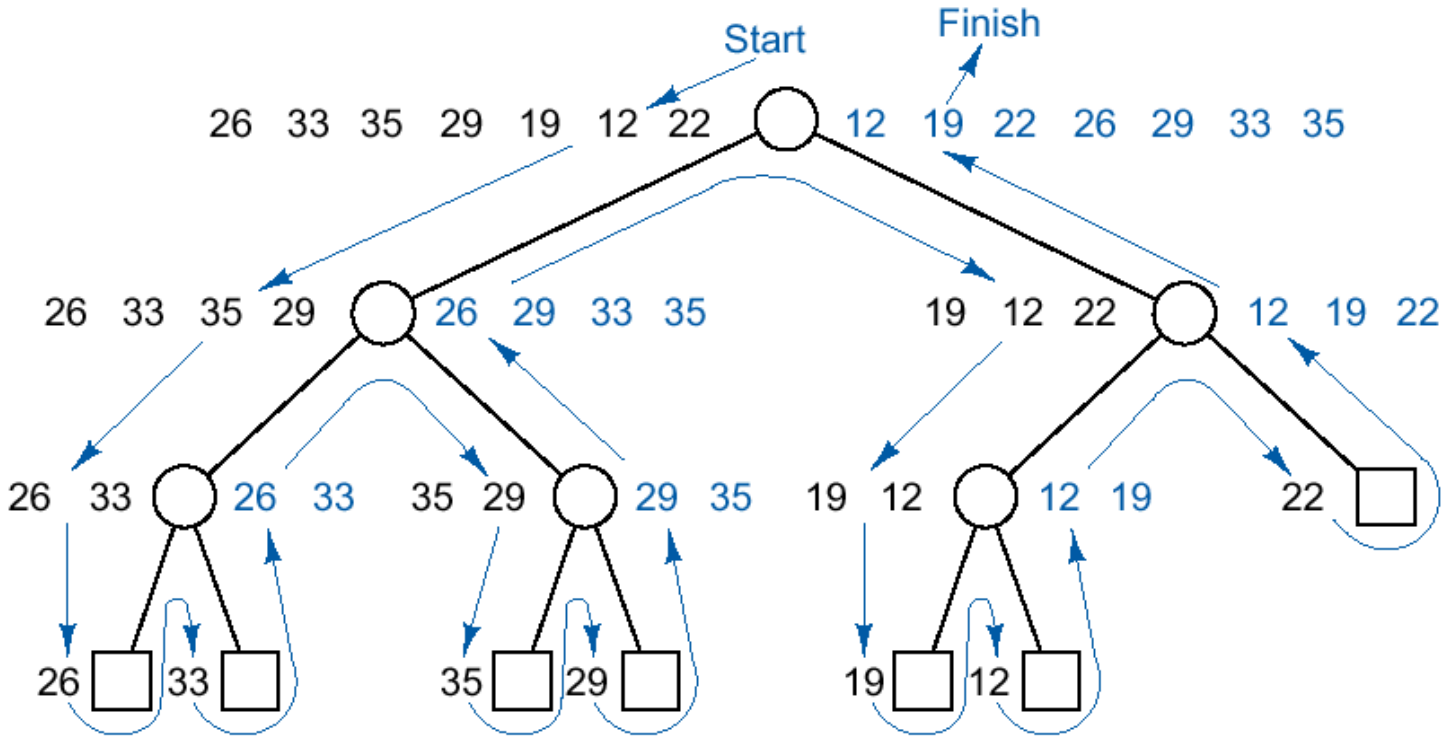
```
MERGE(A, p, q, r)
 1  n1 = q − p + 1                                   Od p po q
 2  n2 = r − q                                       Od q+1 po r
 3  let L[1..n1 + 1] and R[1..n2 + 1] be new arrays  Pomocné polia
 4  for i = 1 to n1
 5      L[i] = A[p + i − 1]                          Prekopíruj do pomocného poľa
 6  for j = 1 to n2
 7      R[j] = A[q + j]                              Prekopíruj do pomocného poľa
 8  L[n1 + 1] = ∞                                    Koniec poľa: nastav na max. celé číslo
 9  R[n2 + 1] = ∞
10  i = 1
11  j = 1
12  for k = p to r
13      if L[i] ≤ R[j]
14          A[k] = L[i]
15          i = i + 1
16      else A[k] = R[j]
17          j = j + 1
```

Pseudokód

Zápis algoritmu pomocou:

for / end for
for each / end for
while / do while / end while
if … else / end if
<, >, = a všetky ostatné matematické symboly
operácie: napr. vymeň(x, y)
input / return

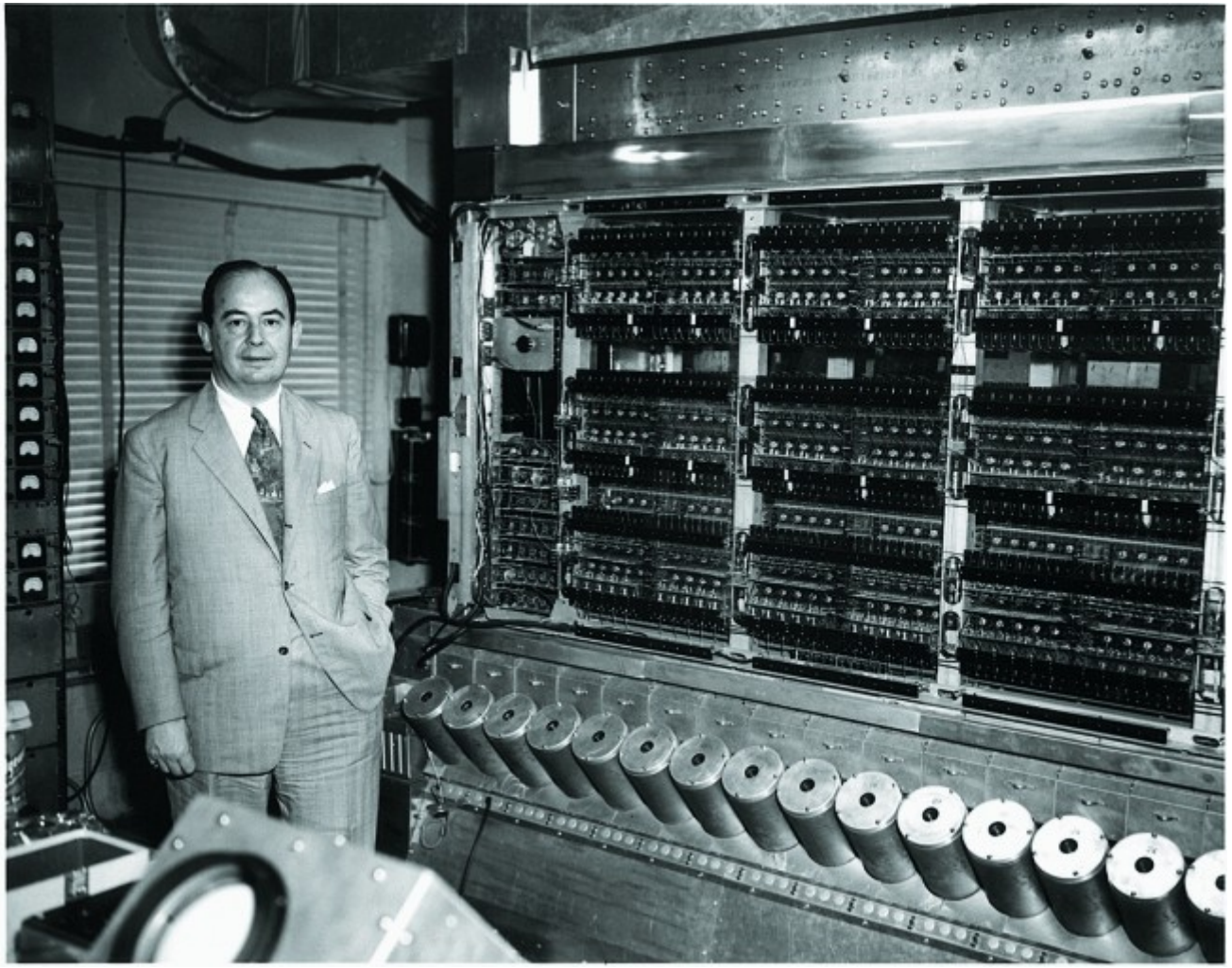
Merge sort



Prehľadávanie do hĺbky

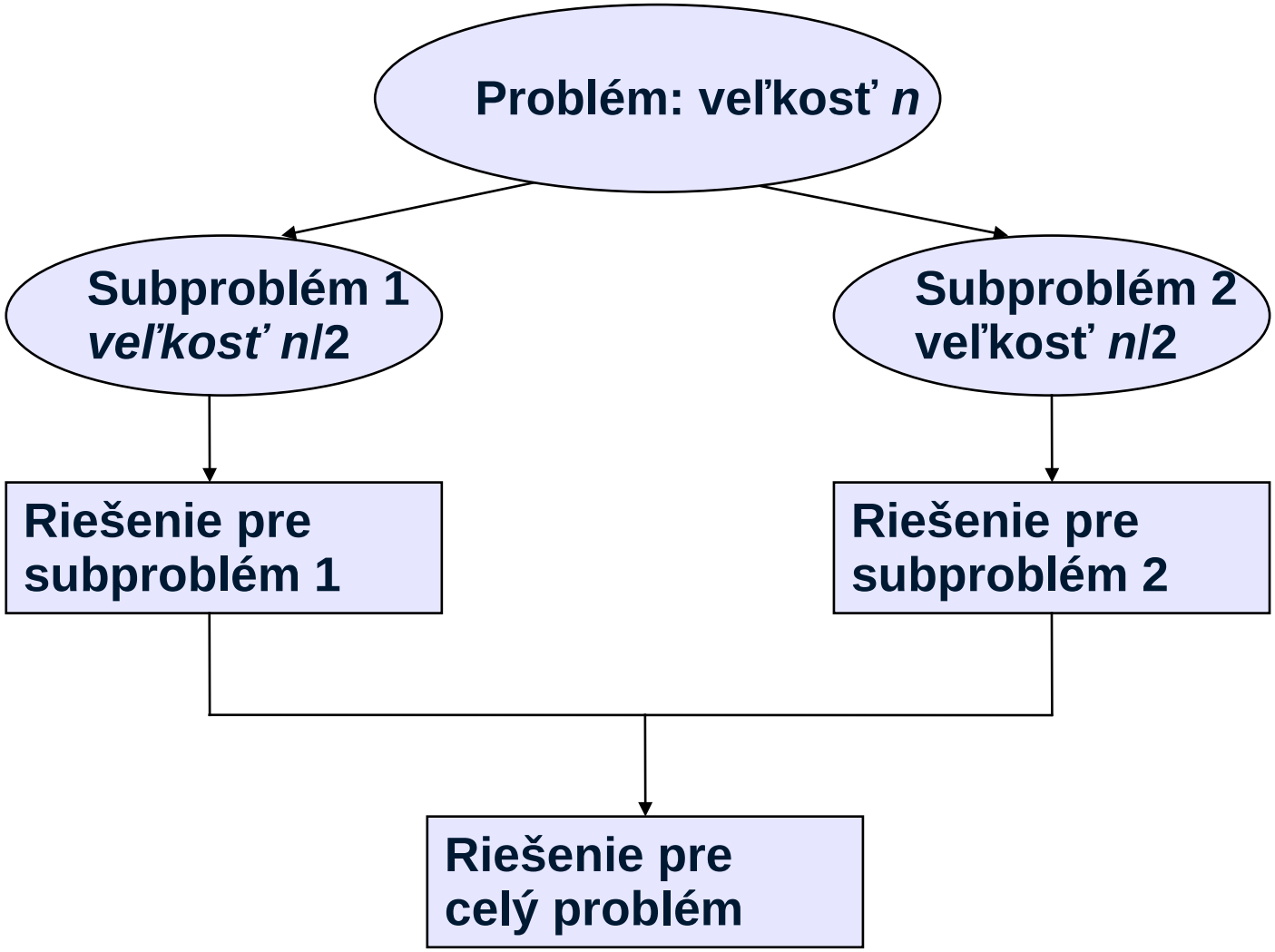
Rekurentná implementácia roztriedi pravý podstrom a potom pravý podstrom.

John von Neumann



Neumann János: merge sort 1945

Rozdeľuj a panuj



Quick sort

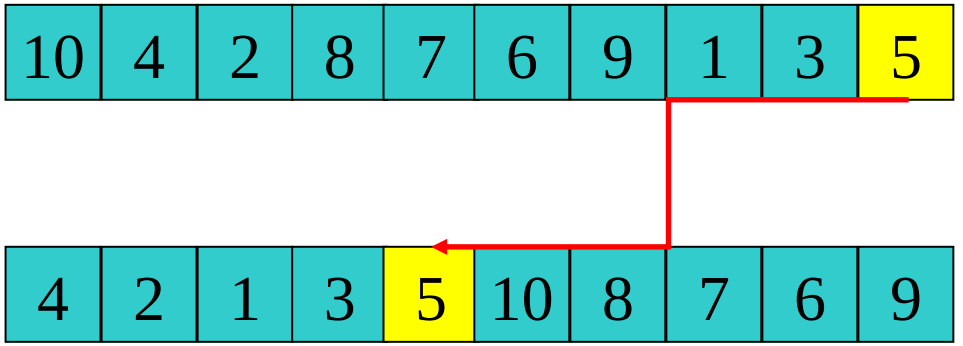
Idea:

Vyberiem z poľa prvok (pivot). Od pivota menšie prvky umiestnim naľavo a väčšie (rovné) napravo.

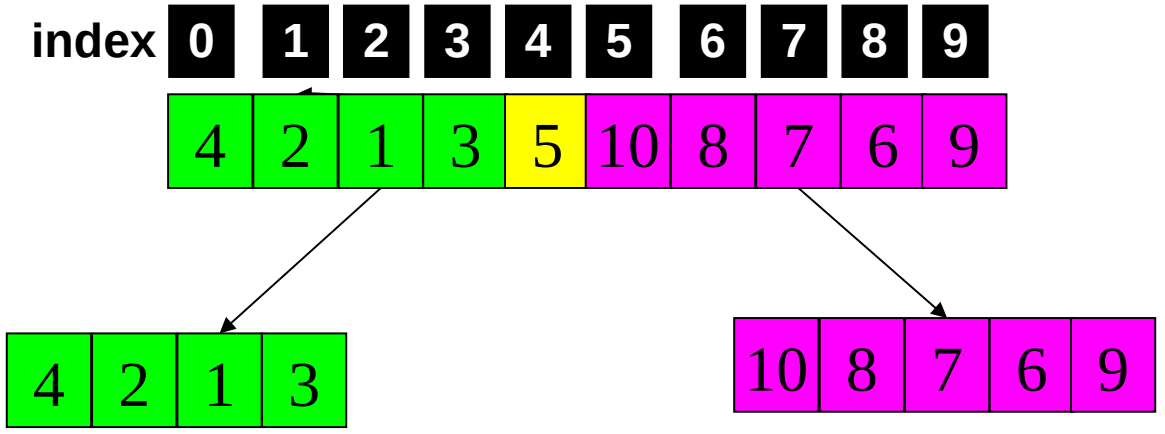
Aplikujem rekurzívne quick_sort zvlášť na menšie prvky a väčšie prvky.

Quick sort

1. quick_sort(0,9)



2. quick_sort(0,3)
   quick_sort(5,9)



Quick sort

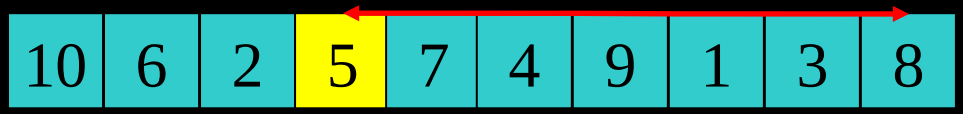
1. Vyberiem náhodne pivota a umiestnim ho na koniec poľa.

2. Prechod celým poľom a ukladanie za sebou na začiatok prvky menšie ako pivot.

3. Umiestnie pivota tesne za menšie prvky.

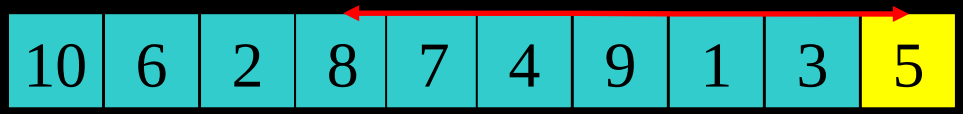
Quick sort

1. Vyberiem náhodne pivota a dám ho na koniec.



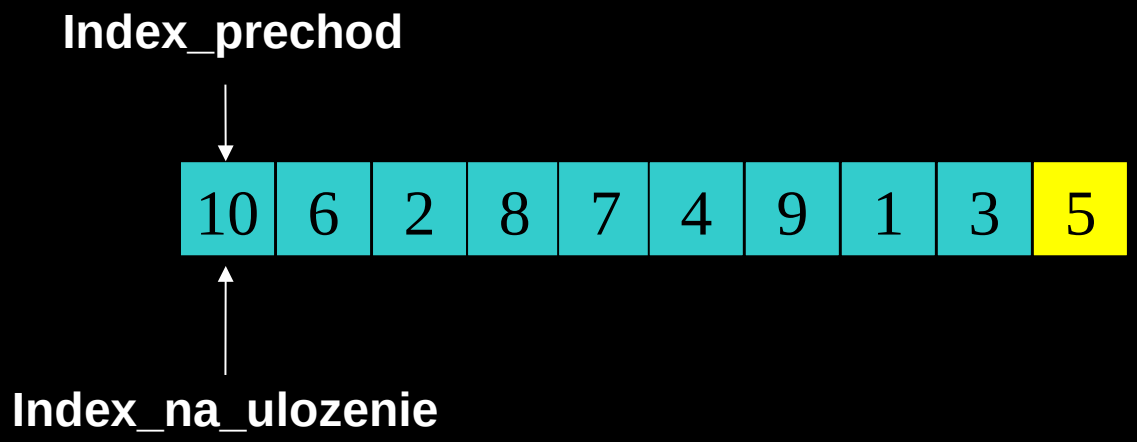
Quick sort

1. Vyberiem náhodne pivota a dám ho na koniec.

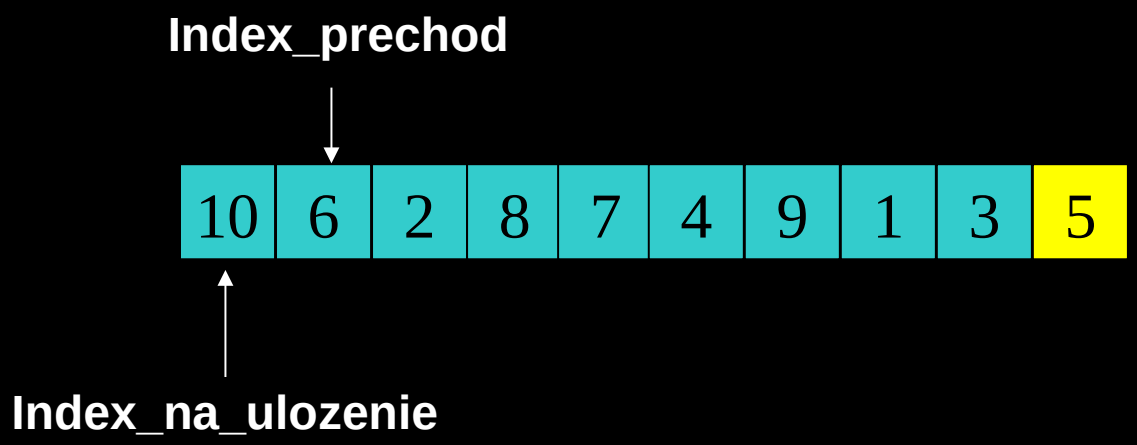


Quick sort

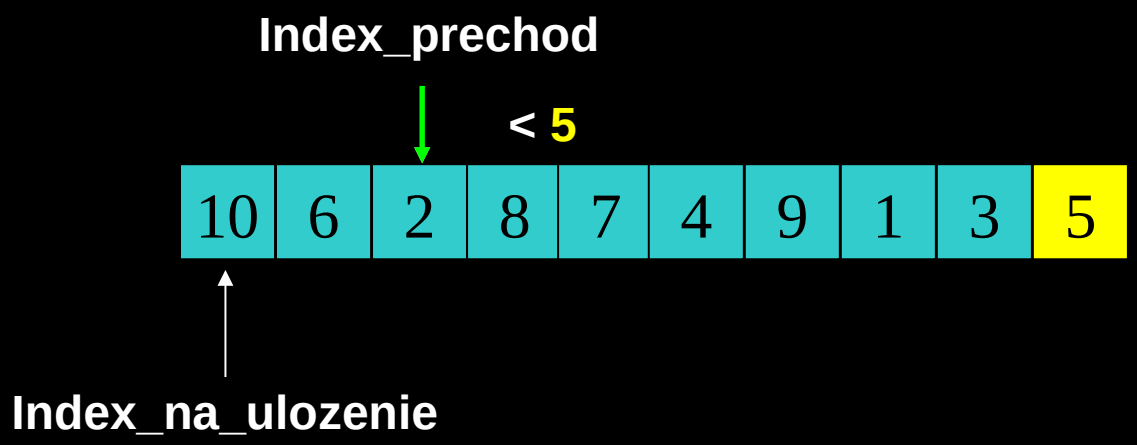
2. Prechod celým poľom a ukladanie za sebou na začiatok prvky menšie ako pivot.



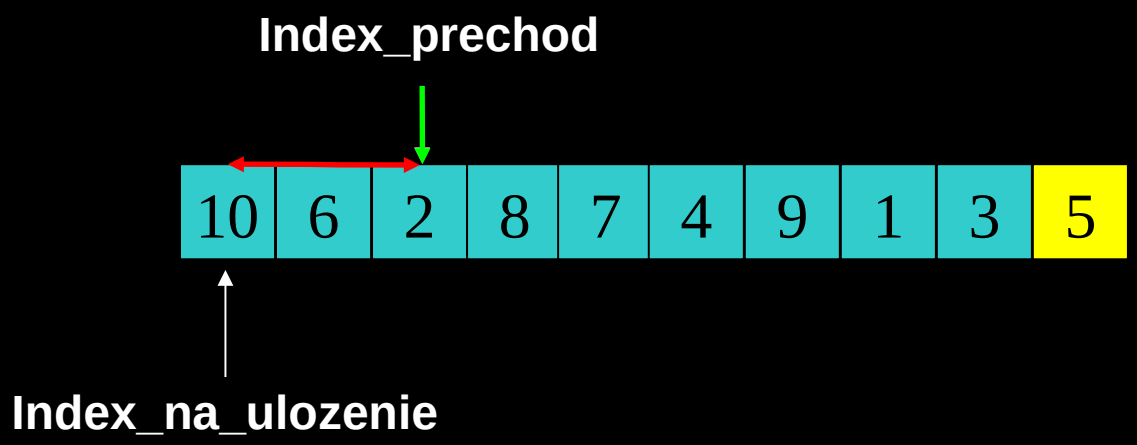
Quick sort



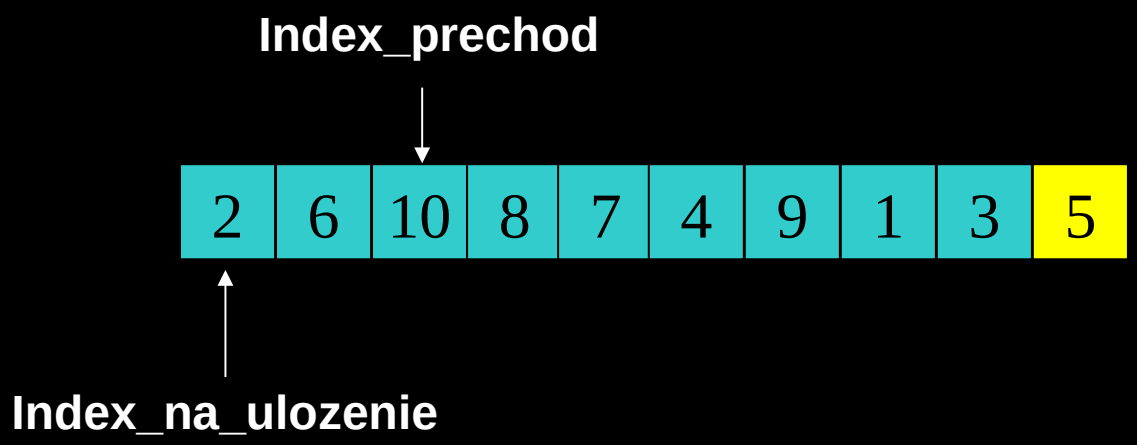
Quick sort



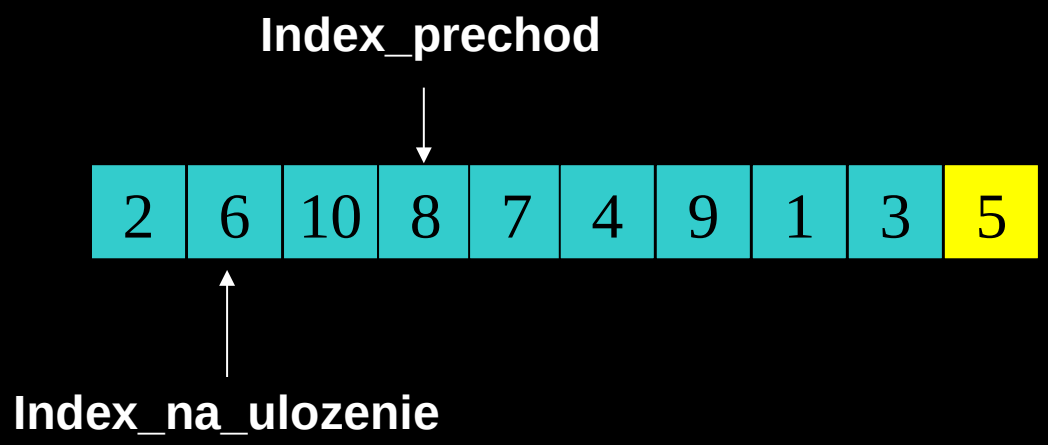
Quick sort



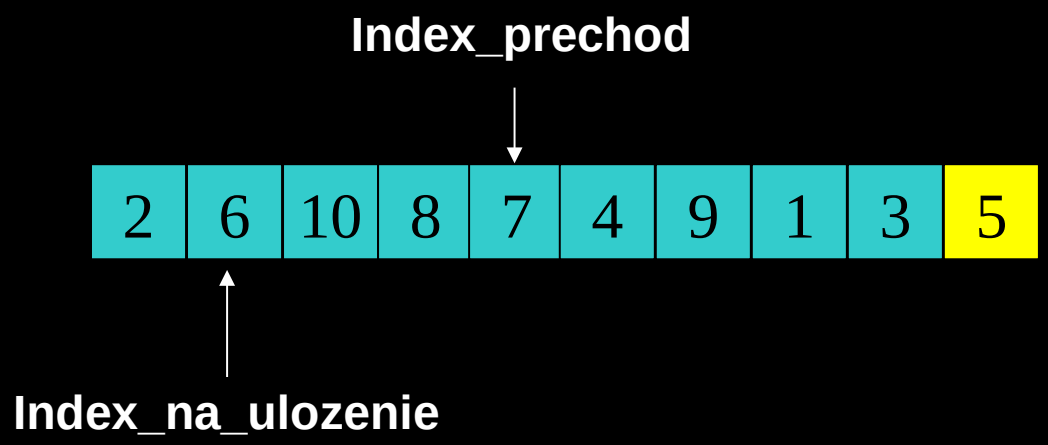
Quick sort



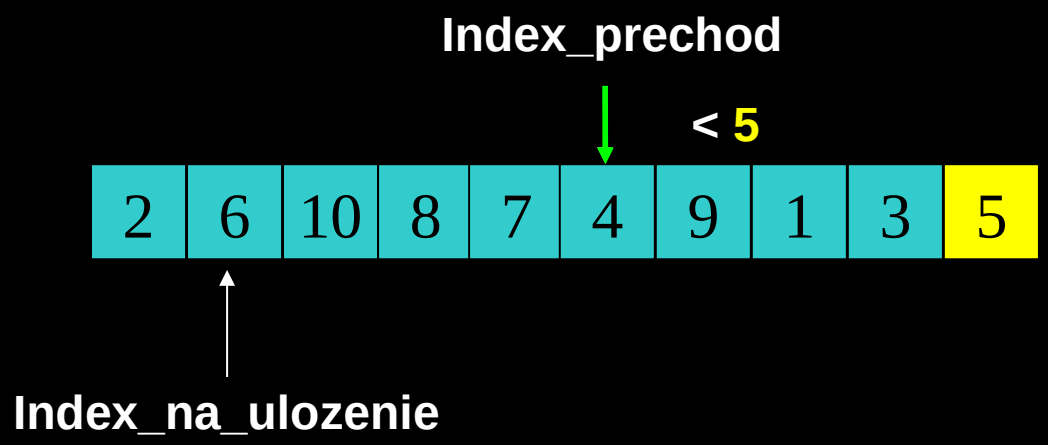
Quick sort



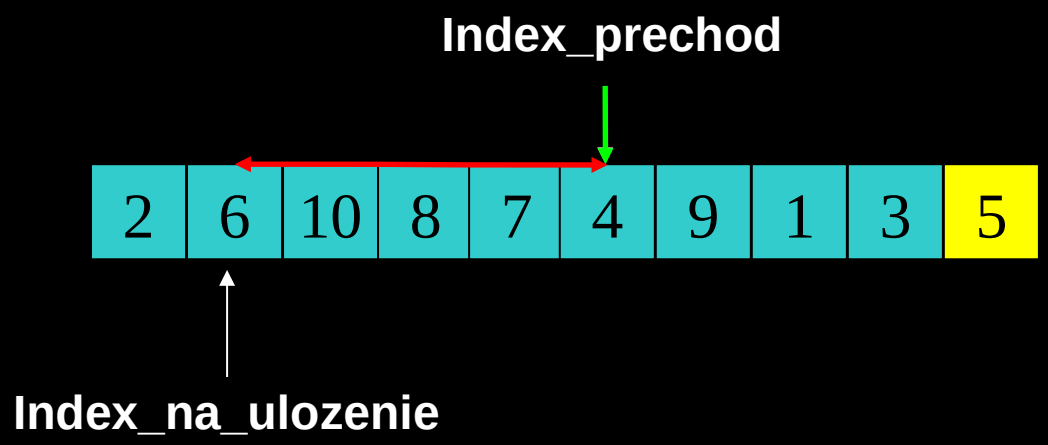
Quick sort



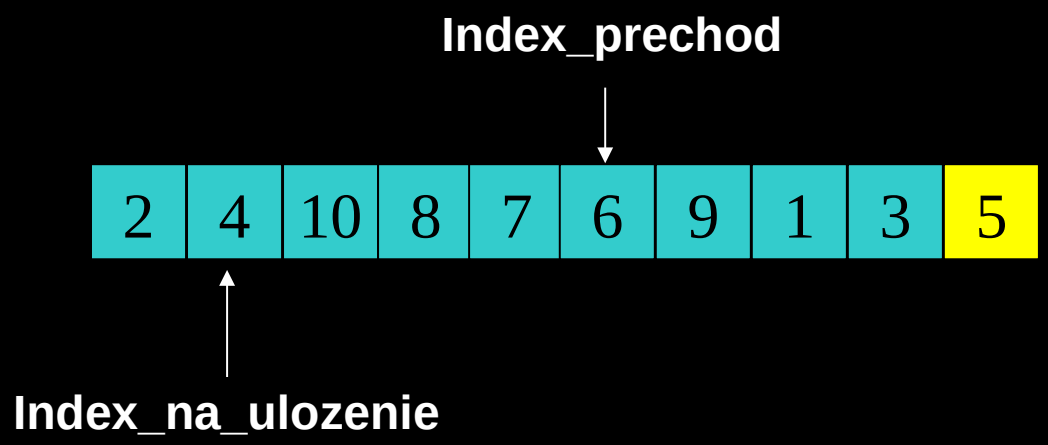
Quick sort



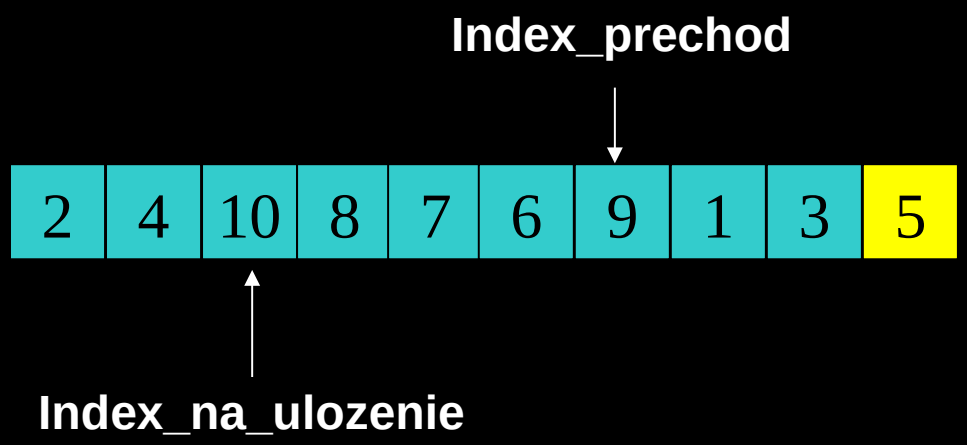
Quick sort



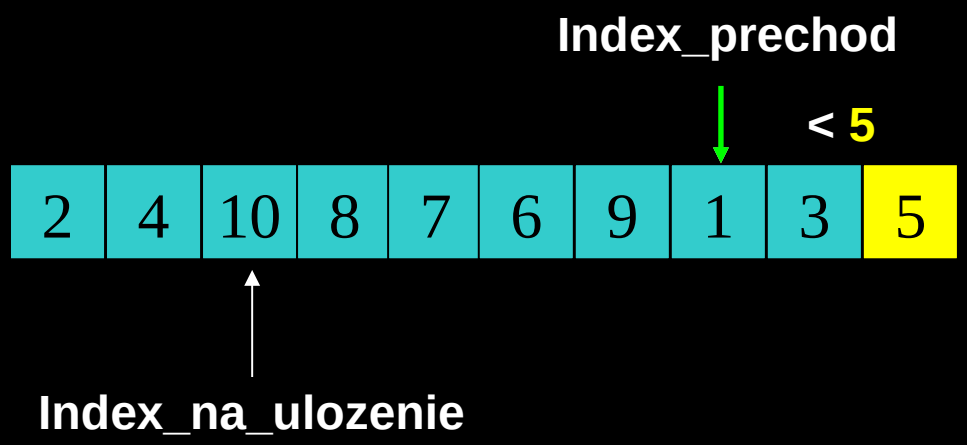
Quick sort



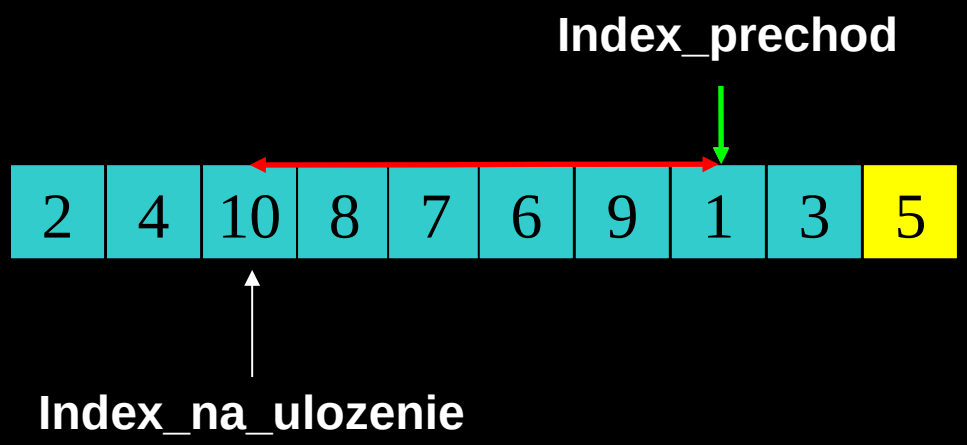
Quick sort



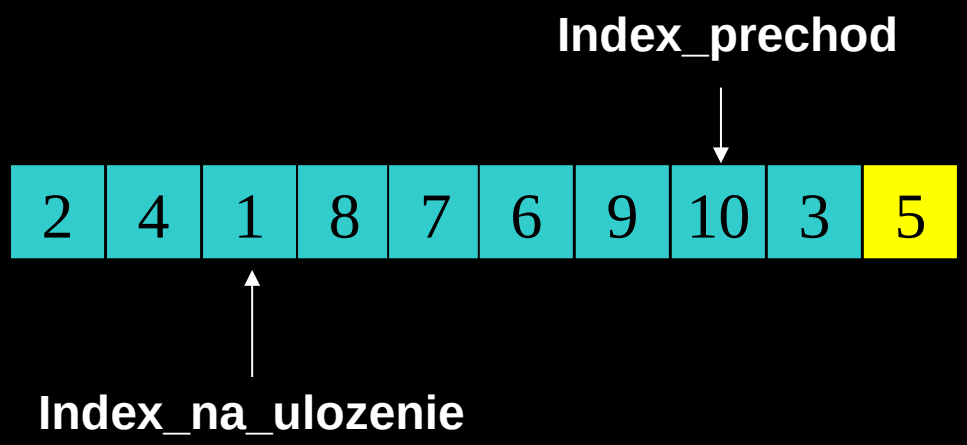
Quick sort



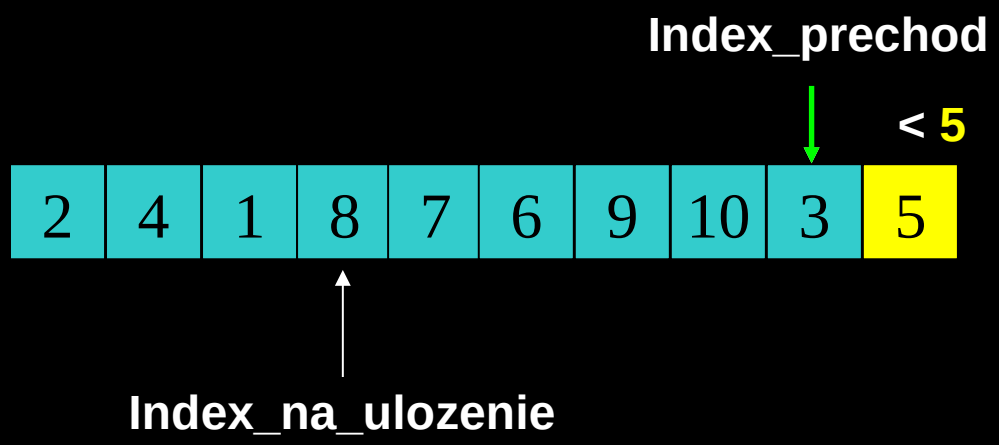
Quick sort



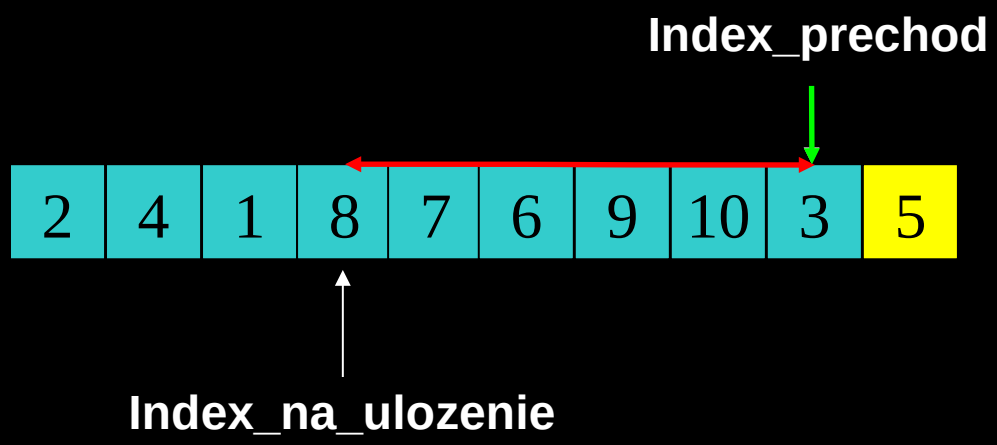
Quick sort



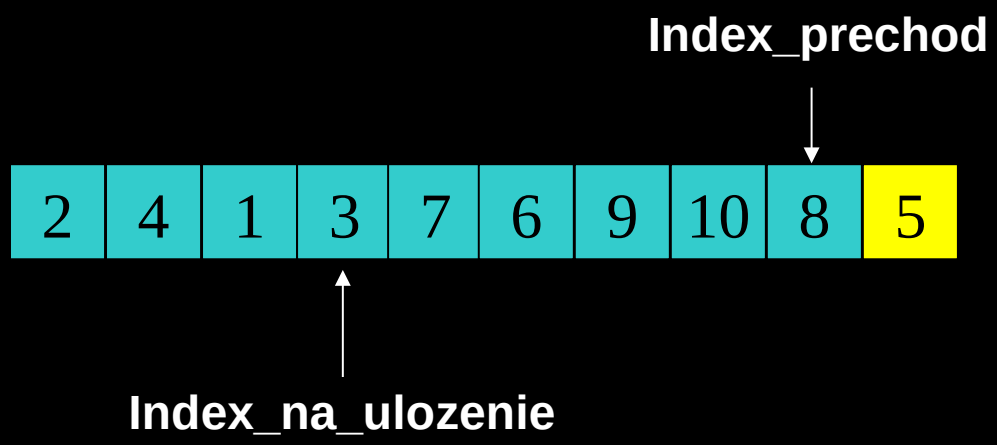
Quick sort



Quick sort



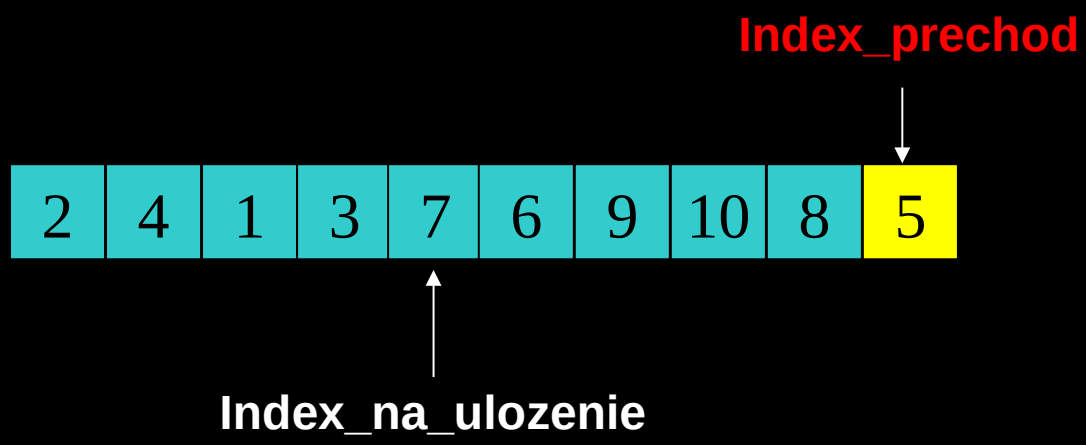
Quick sort



Quick sort

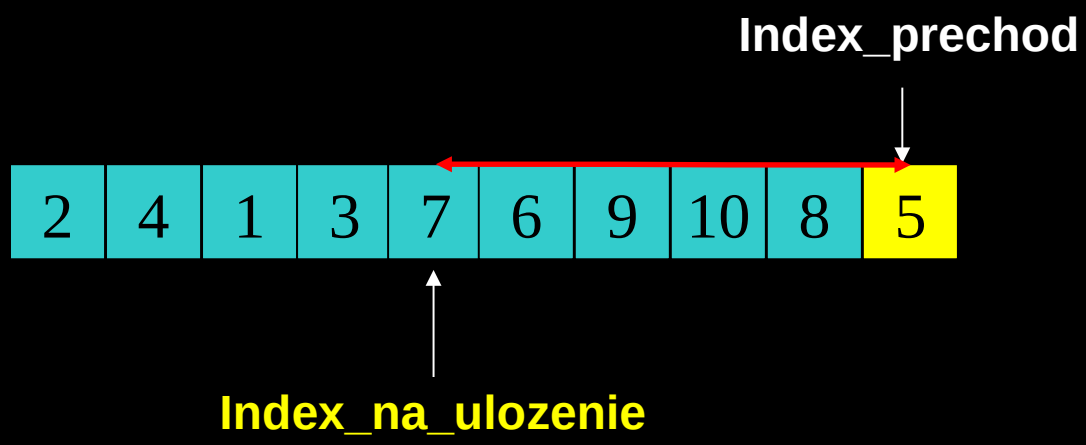
Koniec prechodu:

index_prechod = index pivota



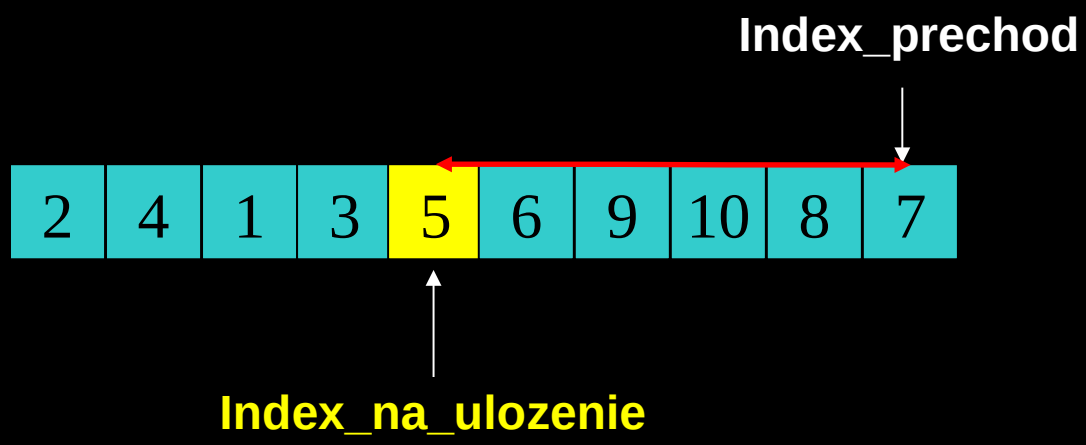
Quick sort

3. Umiestnim pivota na správne miesto index_na_umiestnenie



Quick sort

3. Umiestnim pivota na správne miesto index_na_umiestnenie



Voľba pivota

Prvok vľavo/vpravo: zlá voľba, ak je pole už usporiadané

Náhodná voľba

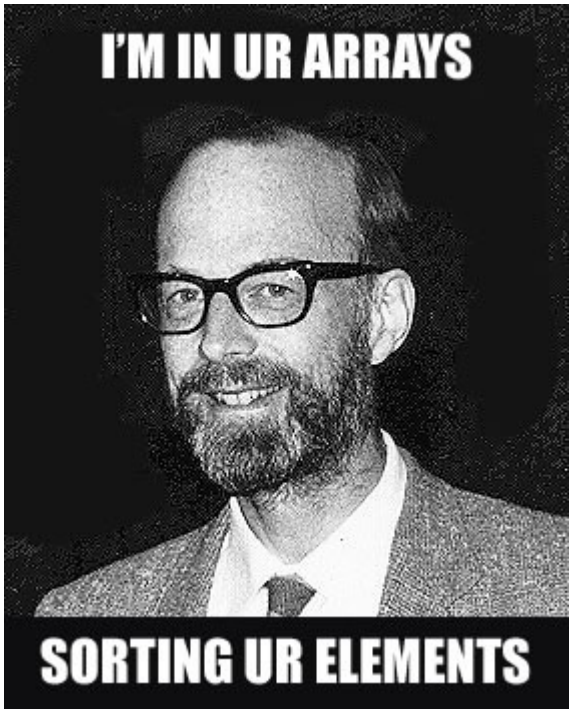
Medián: prvý, posledný a stredný prvok

Quick sort: zložitosť

Najhorší prípad: pivot je najväčší prvok v každom kroku, dve časti o veľkosti n-1 a 0: $O(n^2)$

Najlepší prípad: rozdelenie v každom kroku na dve rovnaké časti: $O(n \log n)$, hĺbka stromu volaní log n

Tony Hoare



Quick sort: 1960 počas študijného pobytu v ZSSR

Timeline

1945: merge sort (von Neumann)
1957: Fortran (John Backus)
1960: quick sort (Hoare)
1972: C (Dennis Ritchie)
1983: C++ (Stroustrup)
1995: Java (Gosling)