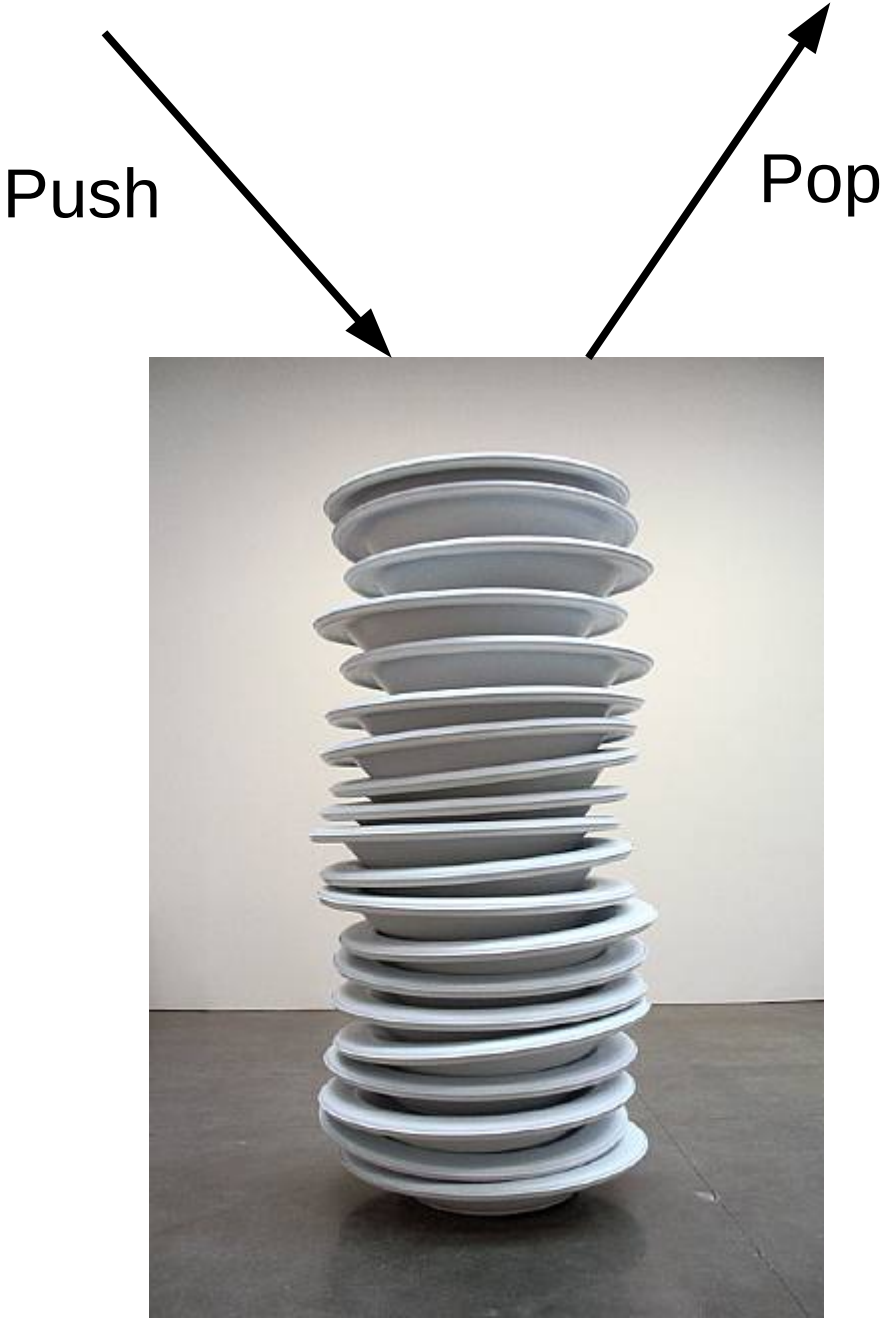
STU FEI

Programovacie techniky

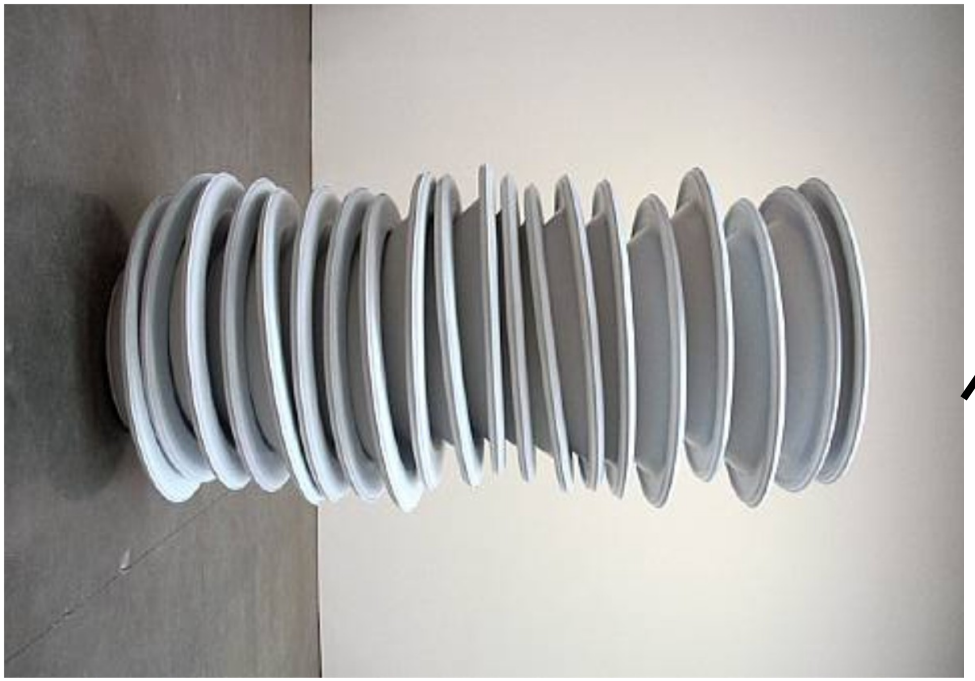
2. Zložitosť, the big O notation, bubble sort, insertion sort

The stack



The queue

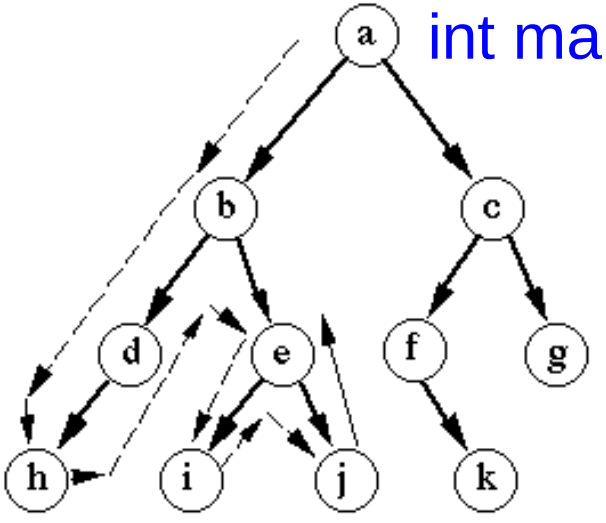
Enqueue



Dequeue

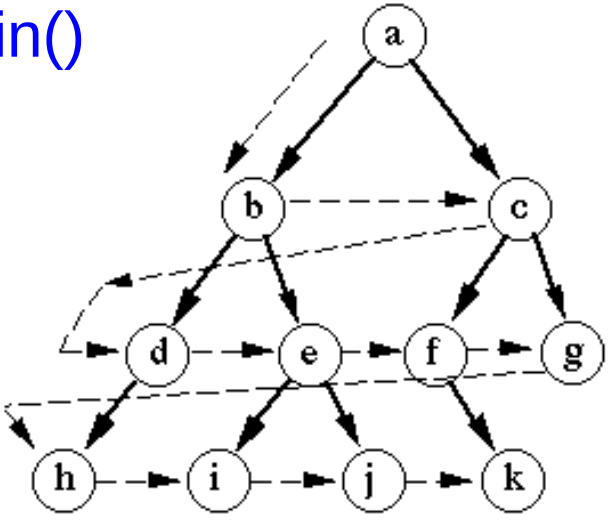
Prehľadávanie do hĺbky/šírky

Zásobník:
a (push a)
ba (push b)
dba (push d)
ba (pop d)
a (pop b)
ba
eba
ba
eba
...



int main()

Depth-first search



Breadth-first search

Zásobník
(push, pop)
DFS

Fronta
(enqueue,
dequeue)
BFS

Operácie

insert: vložiť

delete/erase: zmazať

merge: spojiť

index: prístup ku k-temu prvku/položke

find: násť prvok/položku

Zložitosť operácií: n položiek

	insert	erase	index	merge	find
Zreťazený zoznam	n vlož na ľub. pozíciu	n zmaž ľub. položku	n	1	n
Zásobník	1 push	1 pop	n	1	n
Fronta	1 enqueue	1 dequeue	n	1	n
Pole	?	?	1	?	n
Usporiadané pole	?	?	1	?	log n

Index

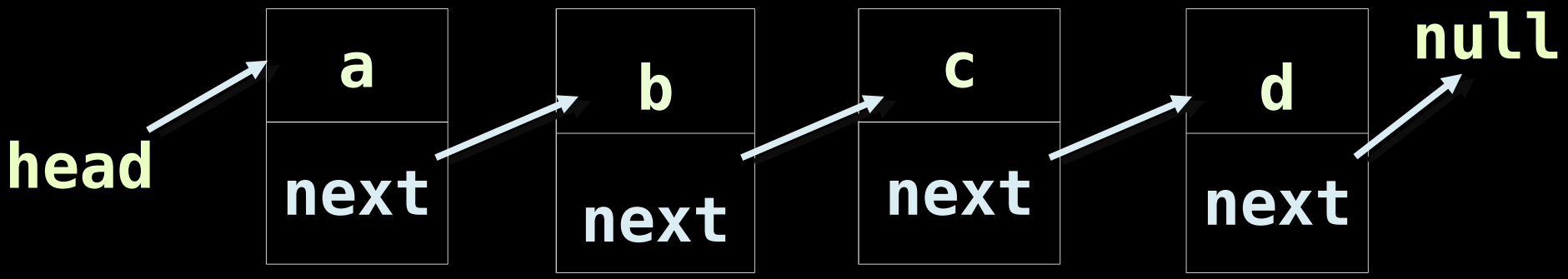
```
int pole[10];

pole[7] = 67; //zložitosť??

*(pole+7) = 67; //ako zložitá je operácia + ??
```

Čo ak nevieme efektívne sčítavať?

Zreťazený zoznam

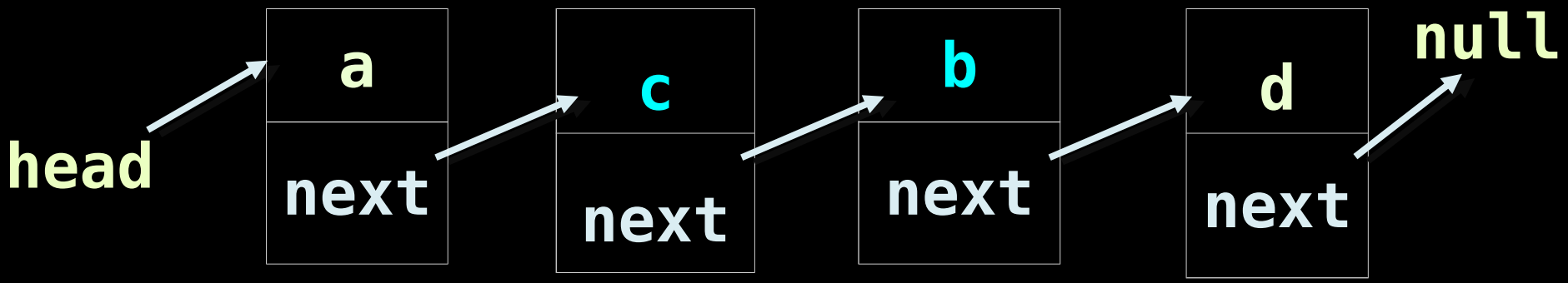


int a, b, c, d; //ak vyžadujeme n premenných, potom je táto možnosť nepraktická

alebo

int a[n]; //pre ľubovolné n

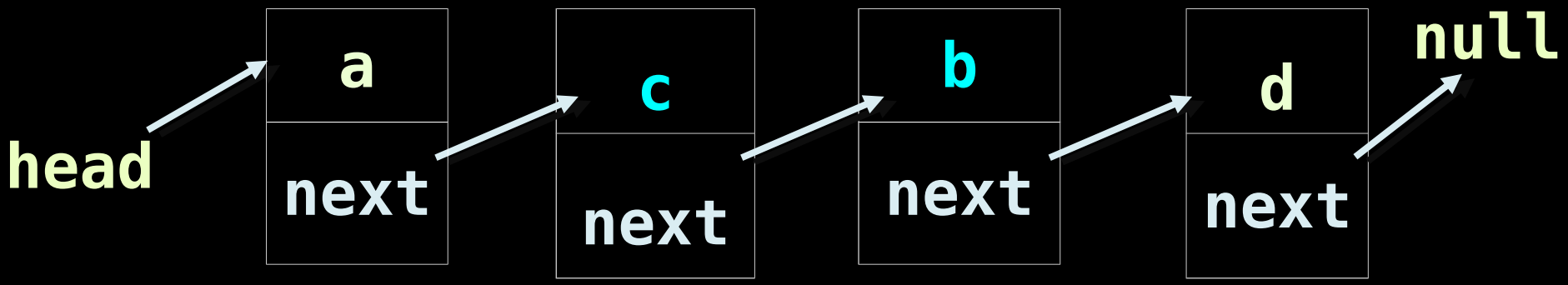
Zreťazený zoznam



int a[n]; //pre ľubovolné n

Ako zložité je vymeniť 2. a 3. položku?

Zreťazený zoznam



int a[n];

Výmena „by reference“ (smerník) a „by value“.

Zložitosť algoritmu

hovorí o efektivite algoritmu

časová – hodnotíme trvanie výpočtu

priestorová – hodnotíme veľkosť pamäte potrebnej na výpočet

Time and space complexity

Obe závisia od počtu n spracovávaných dát!

Rôzny pohľad na zložitosť

zložitosť v najhoršom prípade

zložitosť v priemernom prípade

zložitosť v najlepšom prípade

Nájdenie prvku v poli

v najhoršom prípade: n

v priemernom prípade: 0.5n

v najlepšom prípade: 1

Príklad

Nech algoritmus vykoná na množine s n prvkami nasledovné počty operácií

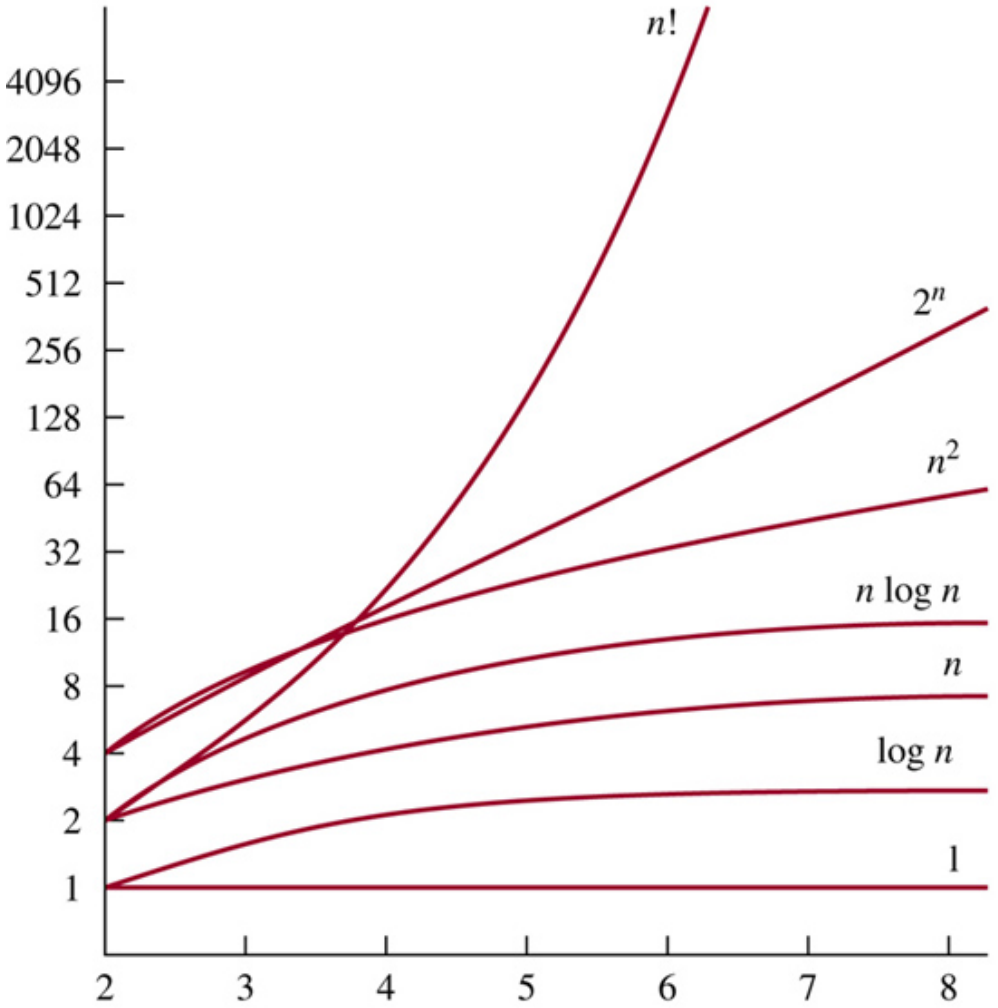
	trvanie operácie		počet
op1	0.2	s	12n
op2	0.001	s	n^2
op3	2	s	log n
op4	1000	s	20

Celkový čas výpočtu: $0.2*12n + 0.001*n^2 + 2*\log n + 1000*20$ sekúnd

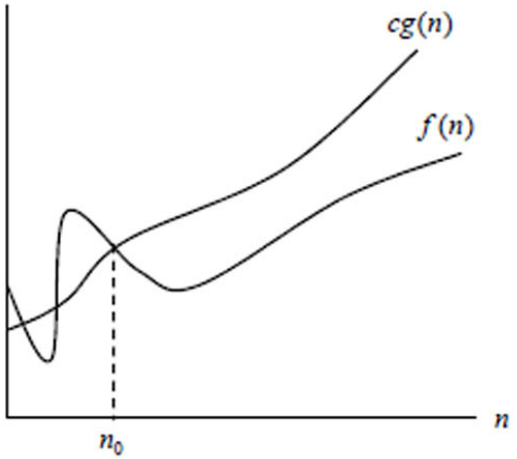
Čo je určujúci člen časovej zložitosti?

- Pre malé n
- Pre veľké n

Zložitosť



The big O notation: O()



$O(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0\}$.

$g(n)$ is an asymptotic upper bound for $f(n)$.

Examples:

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

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

$n^2 + 1000n = O(n^2)$

$5230n^2 + 1000n = O(n^2)$

$n = O(n^2)$

$\frac{n}{1200} = O(n^2)$

$n^{1.99999} = O(n^2)$

$\frac{n^2}{\log n} = O(n^2)$

Note: Since changing the base of a log only changes the function by a constant factor, we usually don't worry about log bases in asymptotic notation.

O(): zložitosť v najhoršom prípade

Zložitosť

O(1): konštantná
O(log n): logaritmická
O(n): lineárna
O(n^2): kvadratická, polynomiálna
O(n^3): kubická, polynomiálna
O(2^n): exponenciálna
O(n!): $n! \underset{+\infty}{\sim} \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$

James Stirling

1730: aproximácia faktoriálu

From Oxford he made his way to Venice, where he occupied himself as a professor of mathematics...Fearing assassination on account of having discovered a trade secret of the glassmakers of Venice, he returned with Newton's help to London about the year 1725. (Wikipedia)

Timeline

1730: aproximácia faktoriálu (Stirling)
1972: C (Dennis Ritchie)
1983: C++ (Stroustrup)
1995: Java (Gosling)

Triedenie

Usporiadanie napr. podľa veľkosti

1, 77, 89, 102, 206, 234

„Martin“ < „Alexander“

alebo

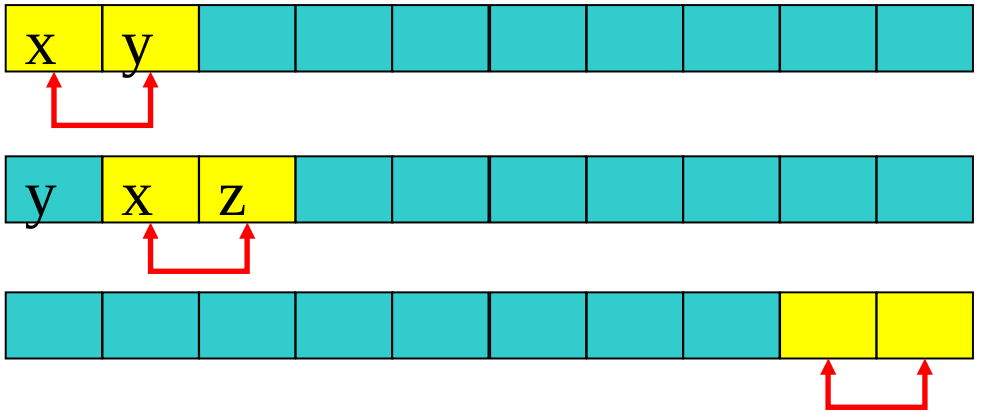
„Martin“ > „Alexander“

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

Bubble sort

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

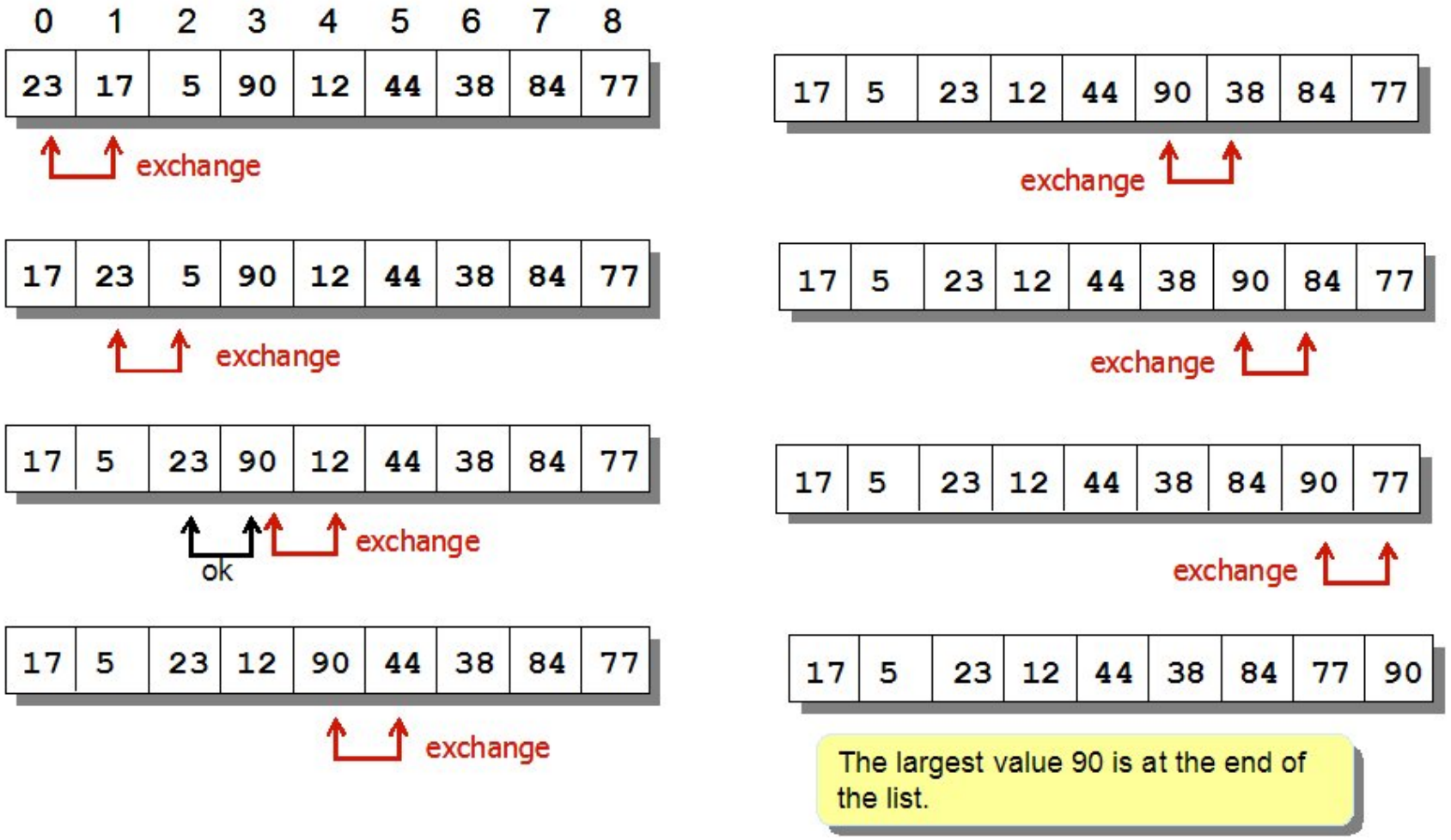
ak x > y



...

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])
      vymen(pole+j, pole+j+1);
}
```

Zložitosť??

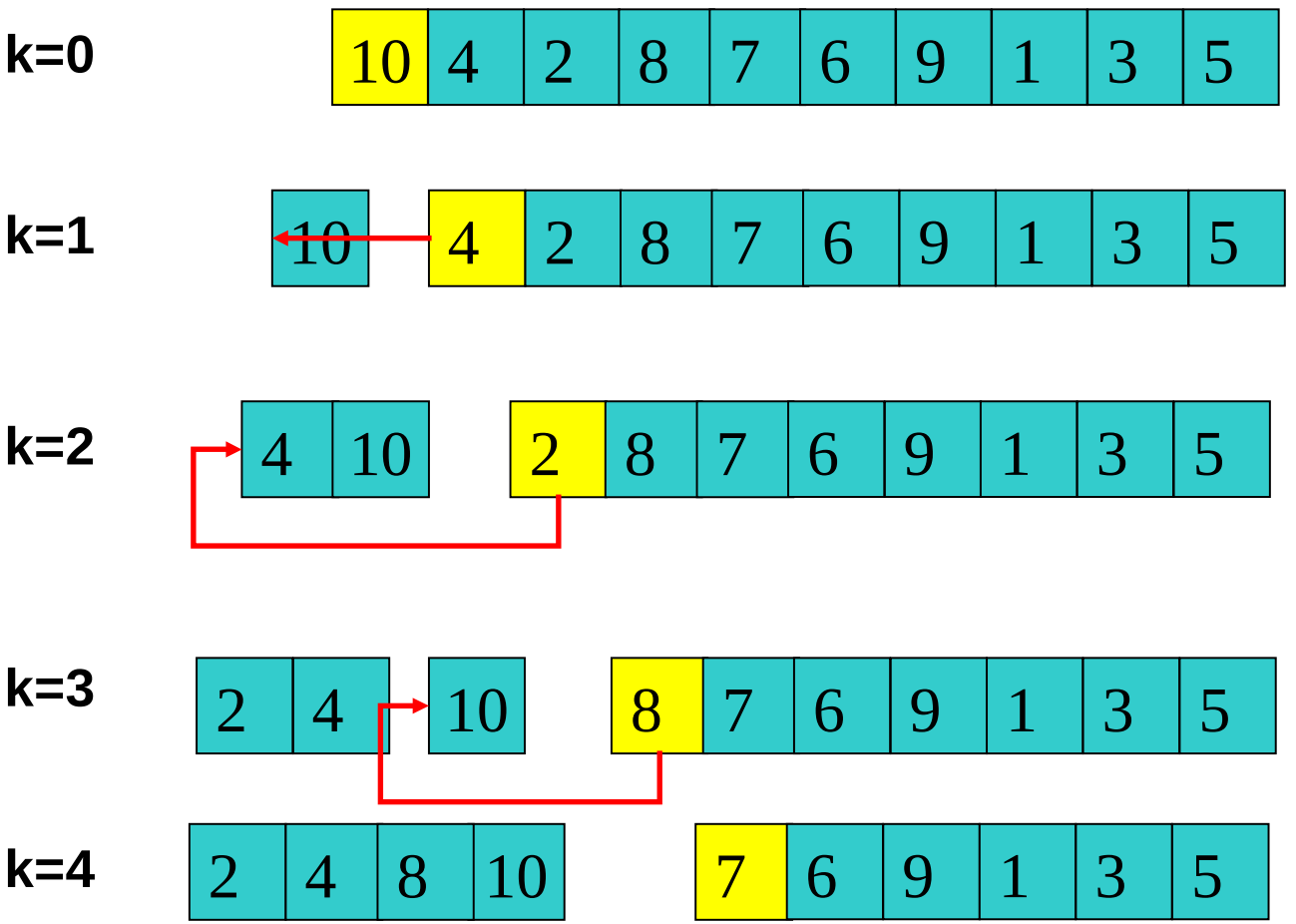
Insertion sort

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

Vkladanie kariet – do už utriedených kariet vlož ďalšiu.

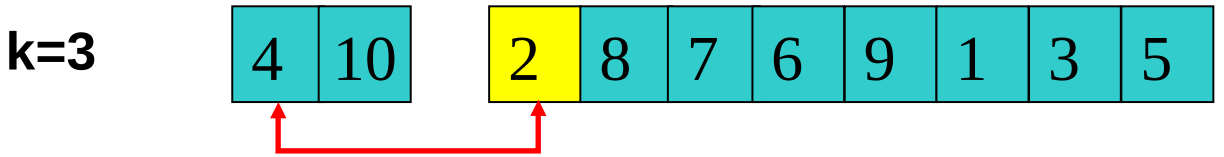
Rôzne implementácie:
1. Na jedinom statickom poli.
2. Na zreťazenom zozname.

Insertion sort: pole

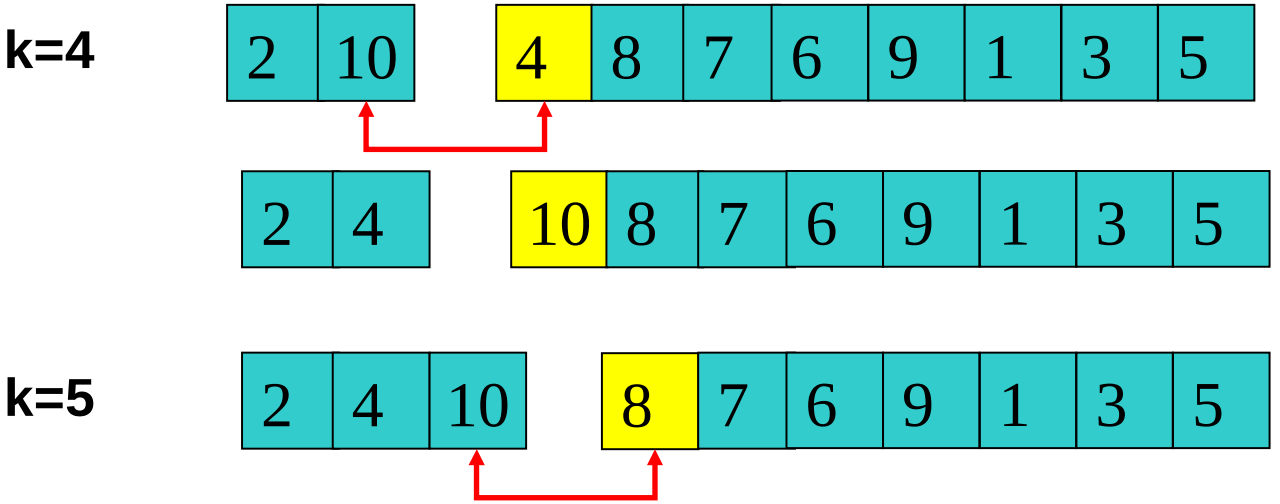


Insertion sort: pomocou výmen

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: ?

Nároky na pamäť??
Space complexity?