Různé algoritmy mají různou složitost

Algoritmus a program není totéž

Řazení

Cvičení

Navrhněte a popište algoritmus pro seřazení balíčku karet.

- **Smíte používat jen jednu ruku a v ní smíte držet vždy nanejvýš jednu kartu.**
- **Během řazení můžete (a musíte) odkládat karty do několika pomocných balíčků.**
- **Z každého balíčku je vidět a je známa pouze hodnota vrchní karty.**

Úkolem je sestavit algoritmus tak, aby využíval co nejmenšího počtu pomocných balíčků.

Řazení

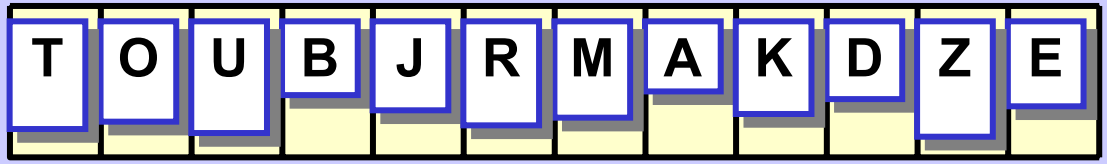
Selection Sort

Řazení výběrem

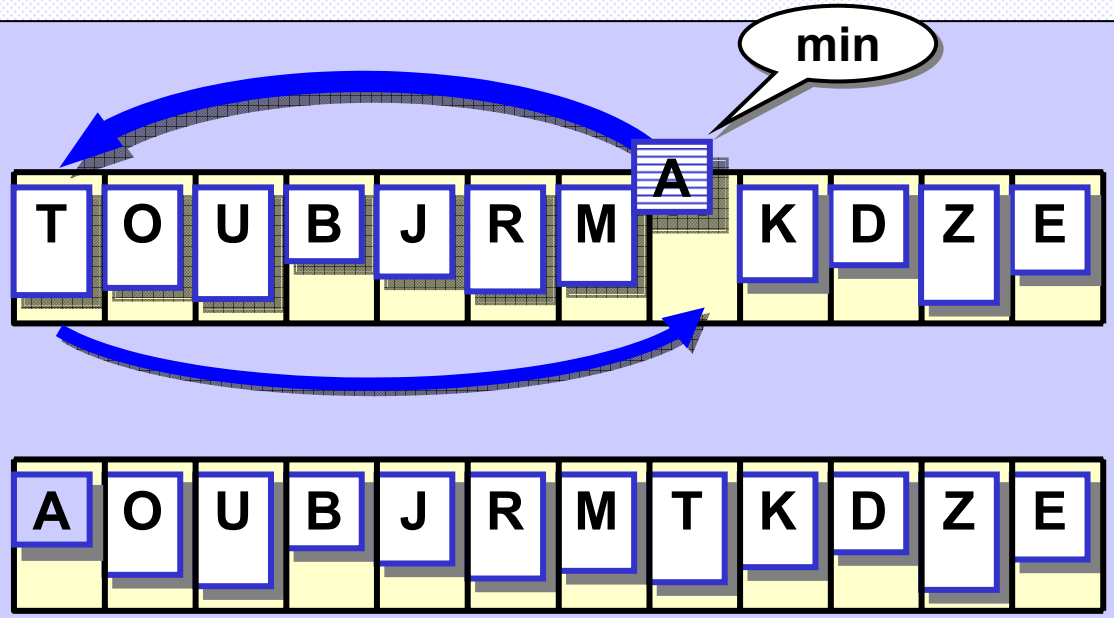
(minima nebo maxima)

Selection Sort

Start

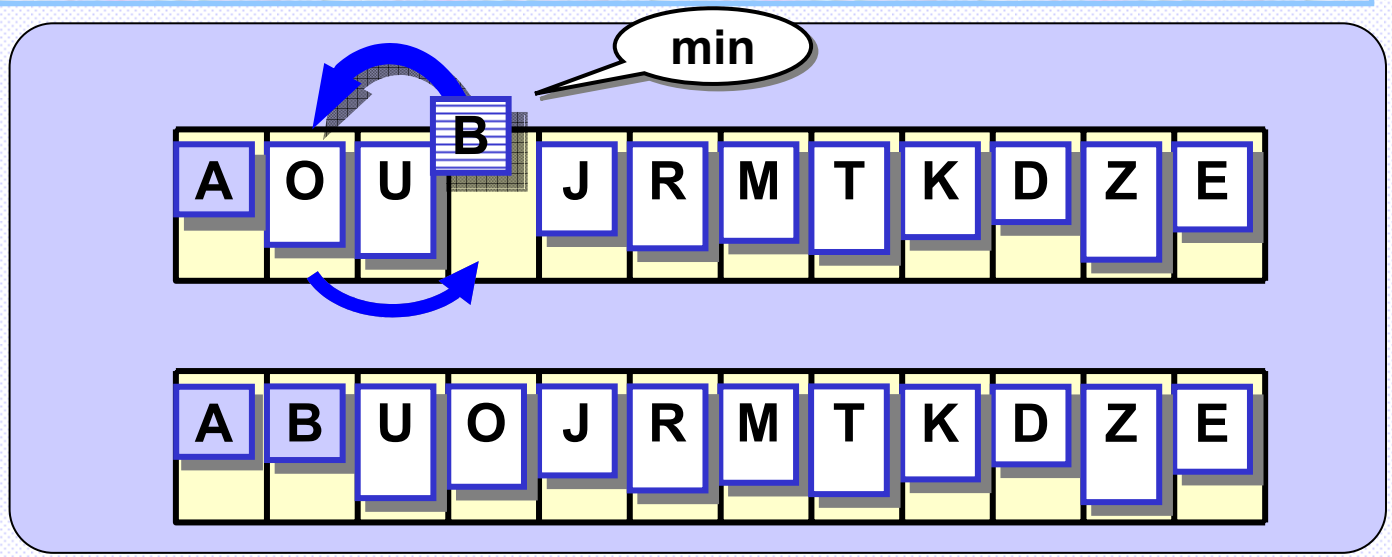


Krok 1

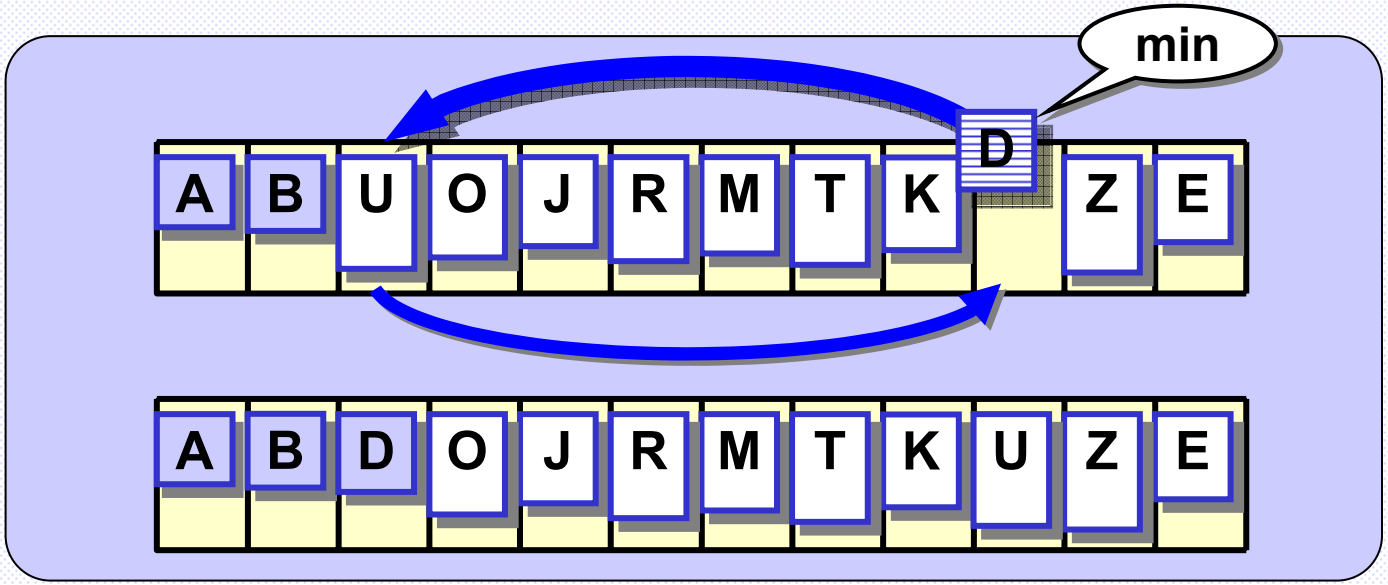


Selection Sort

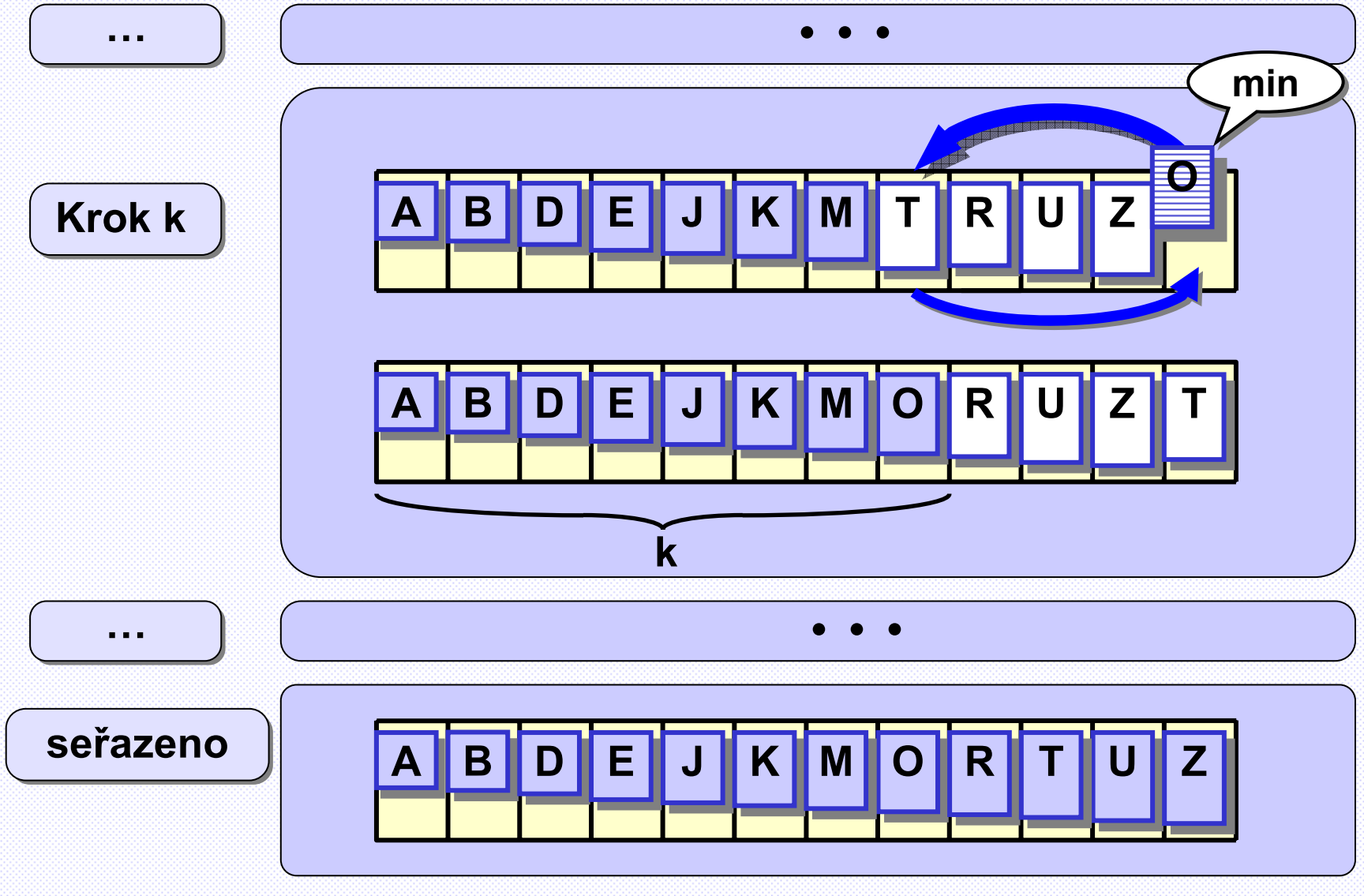
Krok 2



Krok 3



Selection Sort

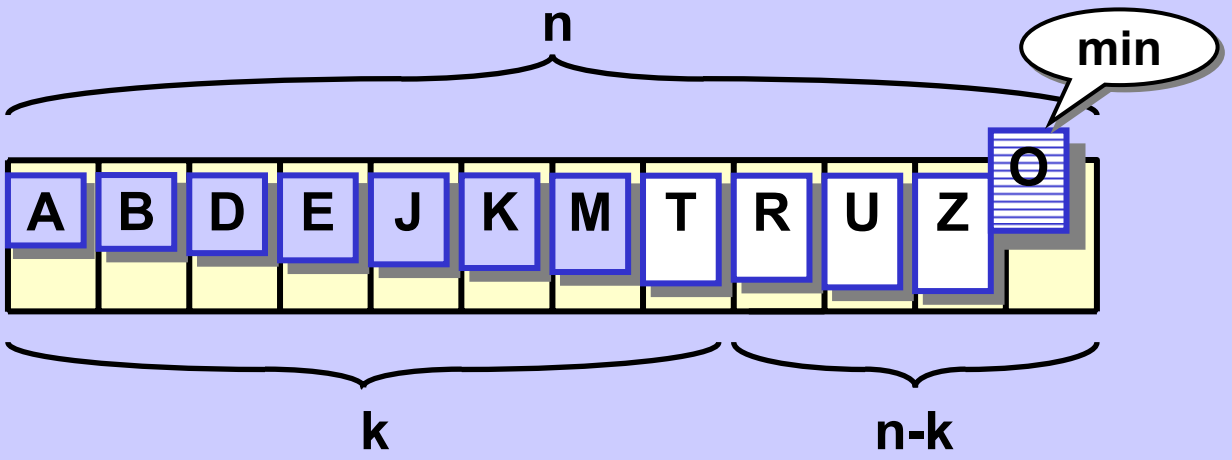


Selection Sort

```
for (i = 0; i < n-1; i++) {
                                    // select min
  jmin = i;
  for (j = i+1; j < n; j++) {
    if (a[j] < a[jmin]) {
      jmin = j;
    }
  }
                                    // put min
  min = a[jmin];
  a[jmin] = a[i];
  a[i] = min;
}
```

Selection Sort

Krok k



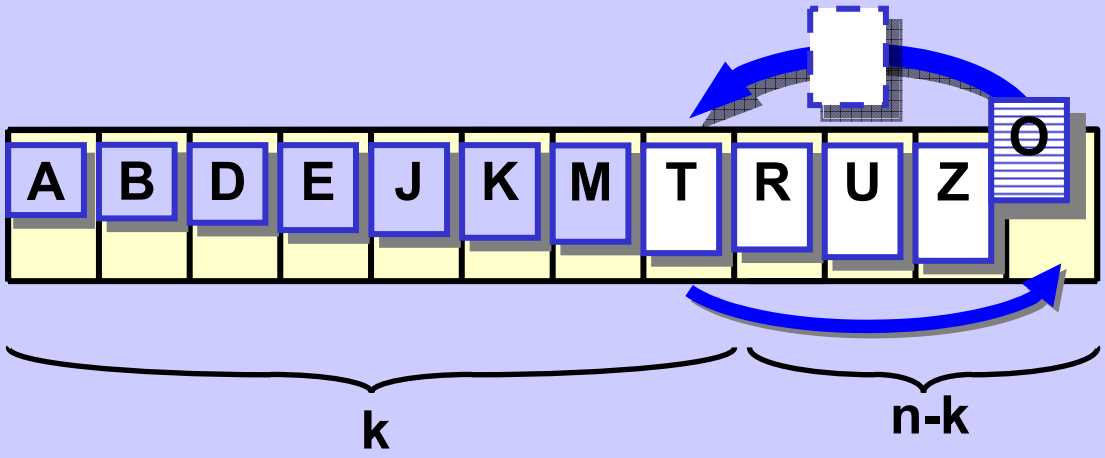
Výběr minima ……. (n-k) testů

Celkem testů

$$\sum_{k=1}^{n-1}(n-k)=\sum_{k=1}^{n-1}n-\sum_{k=1}^{n-1}k=n(n-1)-\frac{n(n-1)}{2}=\frac{1}{2}(n^2-n)$$

Selection Sort

Krok k



přesuny ……. 3

Celkem přesunů

$$\sum_{k=1}^{n-1} 3 = 3(n-1)$$

Selection Sort

Shrnutí

Celkem testů	$\frac{1}{2}(n^2 - n) = \Theta(n^2)$
Celkem přesunů	$3(n-1) = \Theta(n)$
Celkem operací	$\frac{1}{2}(n^2 - n) + 3(n-1) = \Theta(n^2)$

Asymptotická složitost Selection Sortu je $\Theta(n^2)$

Řazení

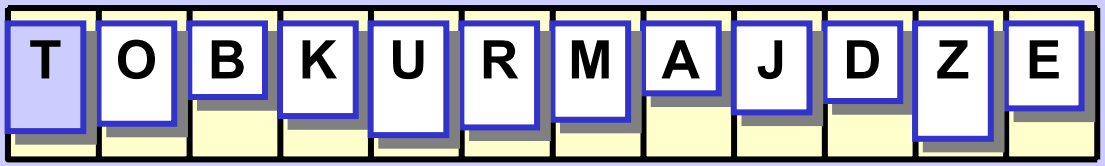
Insertion Sort

Řazení vkládáním

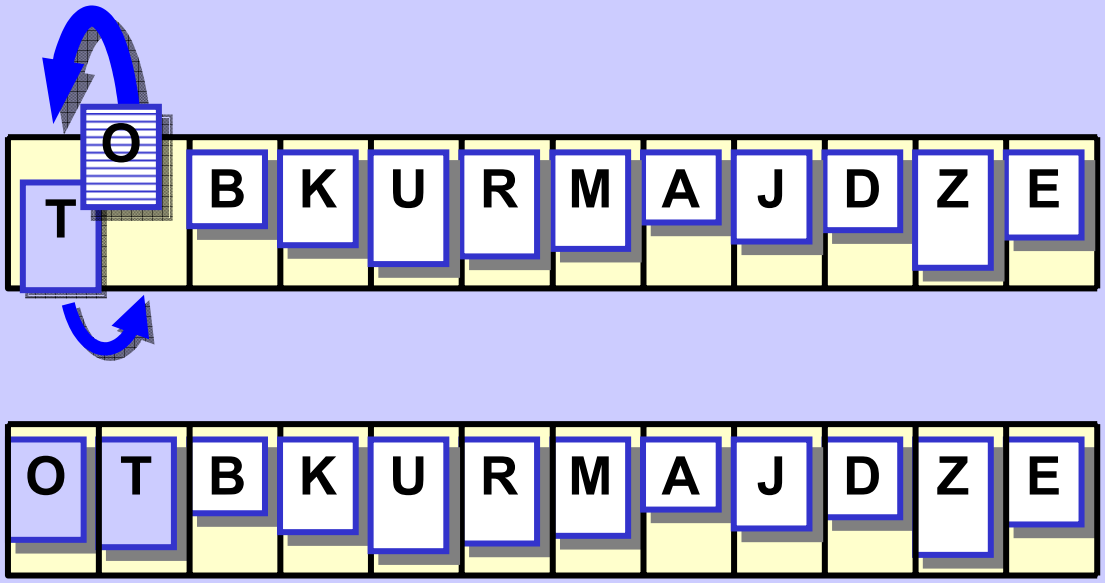
(na adekvátní pozici)

Insertion Sort

Start

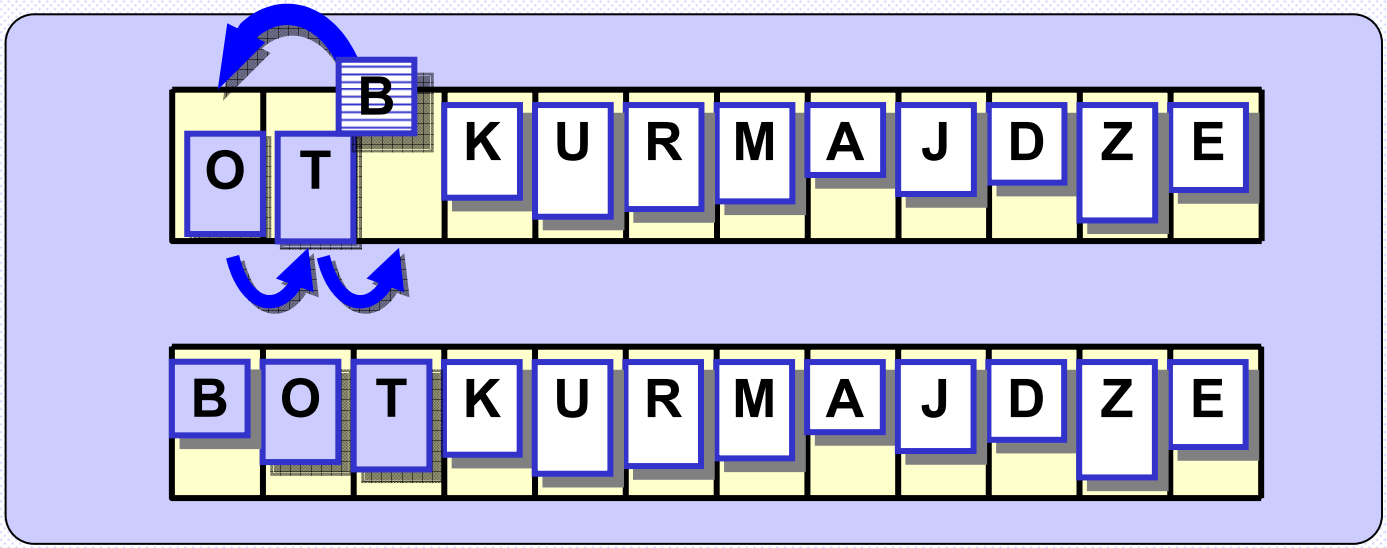


Krok 1

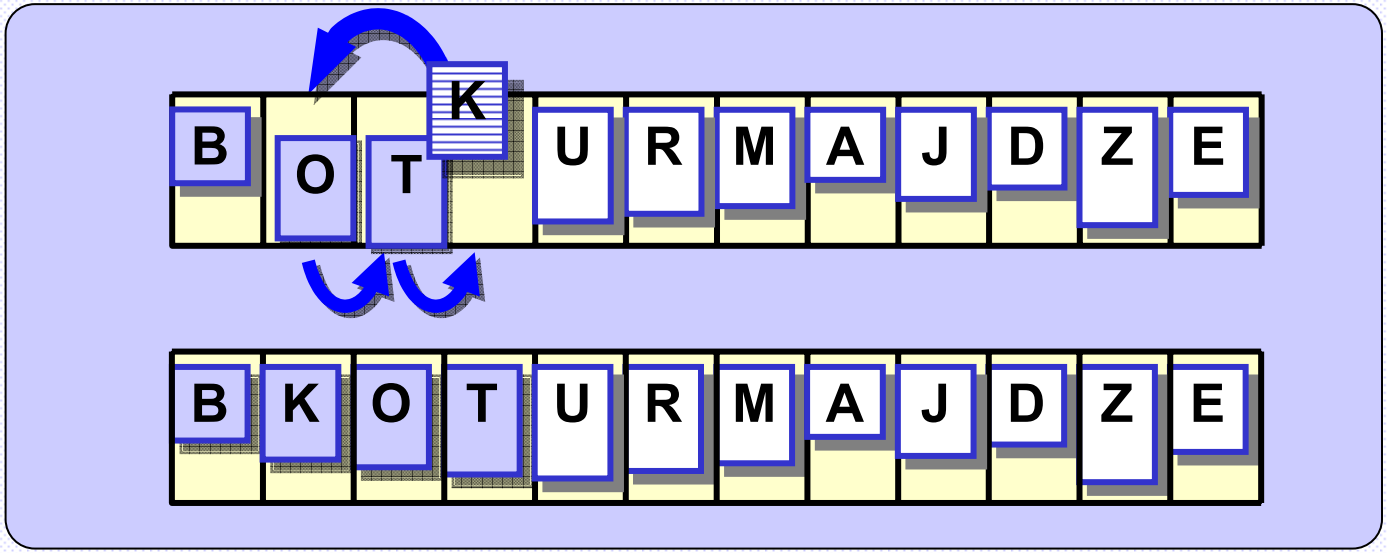


Insertion Sort

Krok 2



Krok 3

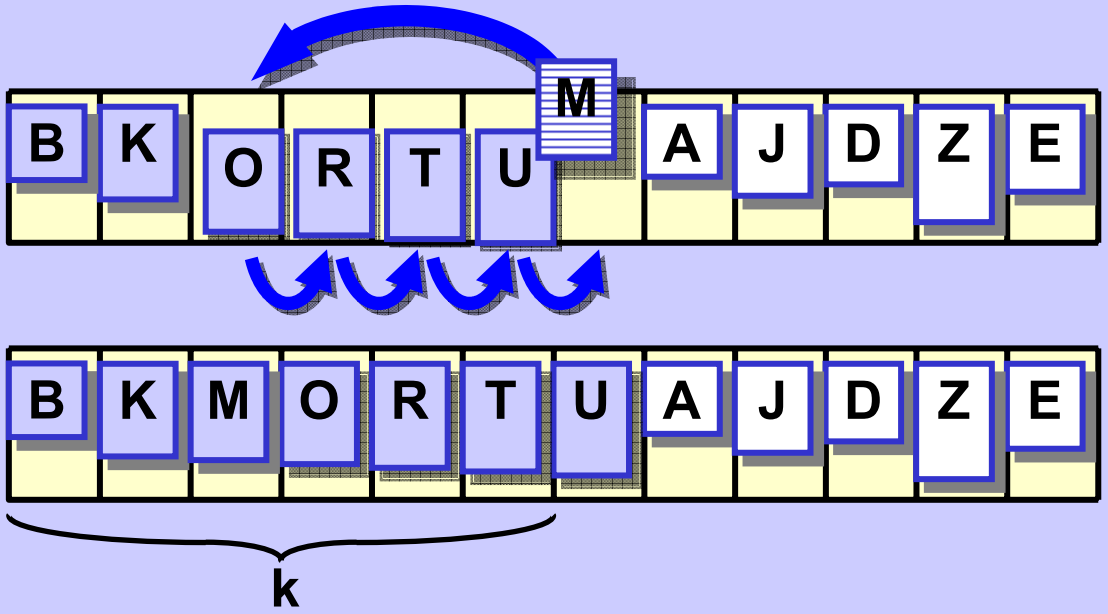


Insertion Sort

…

• • •

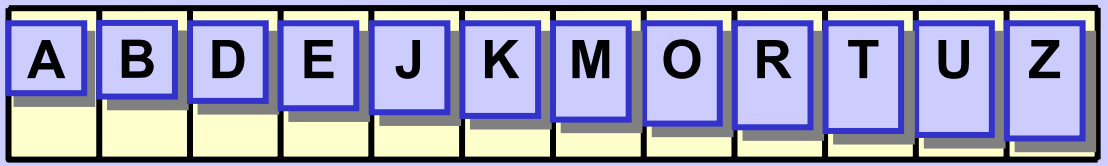
Krok k



…

• • •

seřazeno

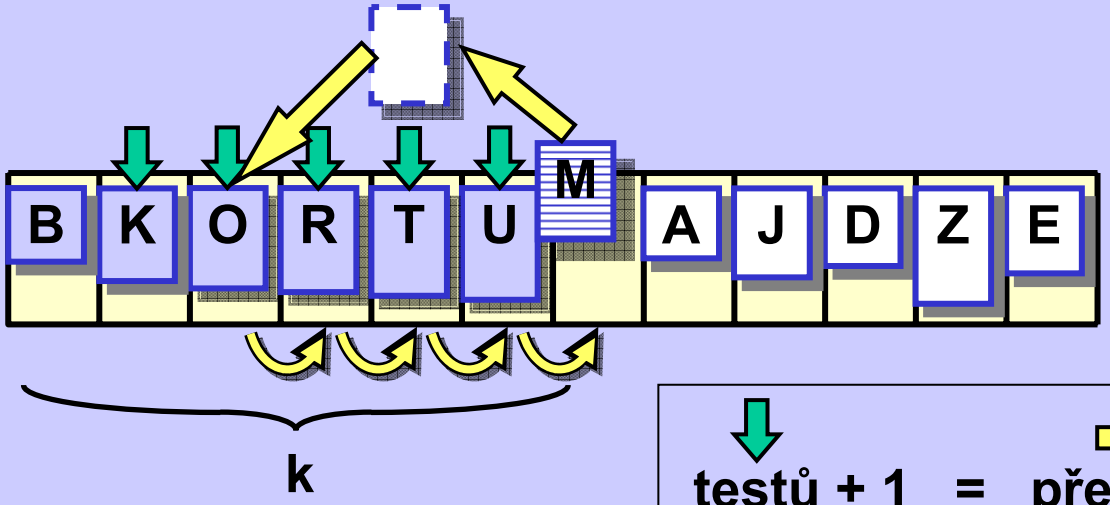


Insertion Sort

```
for (i = 1; i < n; i++) {
                                    // find & make
                                    // place for a[i]
  insVal = a[i];
  j = i-1;
  while ((j >= 0) && (a[j] > insVal)) {
    a[j+1] = a[j];
    j--;
  }
                                    // insert a[i]
  a[j+1] = insVal;
}
```

Insertion Sort

Krok k



testů + 1 = přesunů

testů …….	1	nejlepší případ
	k	nejhorší případ
	(k+1)/2	průměrný případ

přesunů …….	2	nejlepší případ
	k+1	nejhorší případ
	(k+3)/2	průměrný případ

Insertion Sort

Shrnutí

Celkem testů

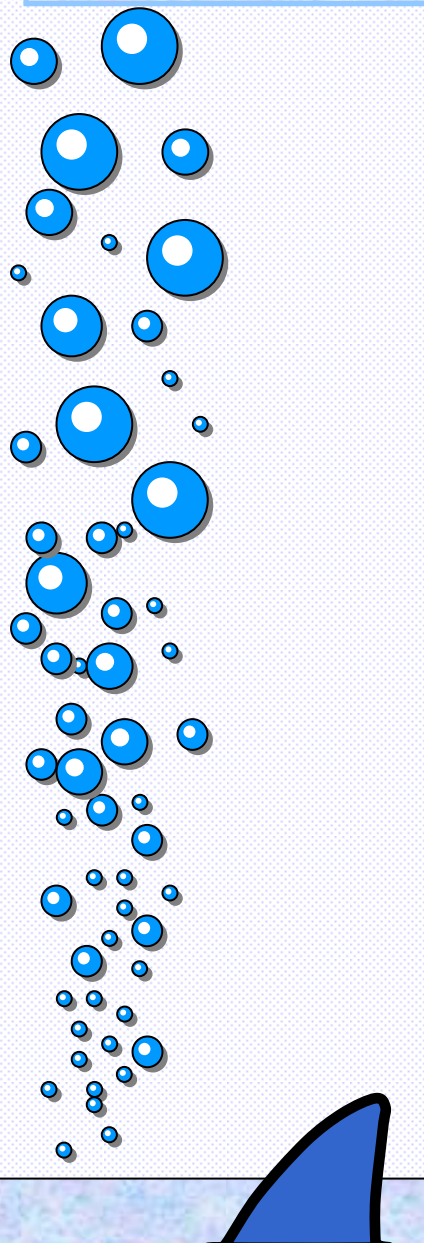
$n - 1$	$= \Theta(n)$	nejlepší případ
$(n^2 - n)/2$	$= \Theta(n^2)$	nejhorší případ
$(n^2 + n - 2)/4$	$= \Theta(n^2)$	průměrný případ

Celkem přesunů

$2n - 2$	$= \Theta(n)$	nejlepší případ
$(n^2 + n - 2)/2$	$= \Theta(n^2)$	nejhorší případ
$(n^2 + 5n - 6)/4$	$= \Theta(n^2)$	průměrný případ

Asymptotická složitost Insertion Sortu je $O(n^2)$ (!!)

Řazení

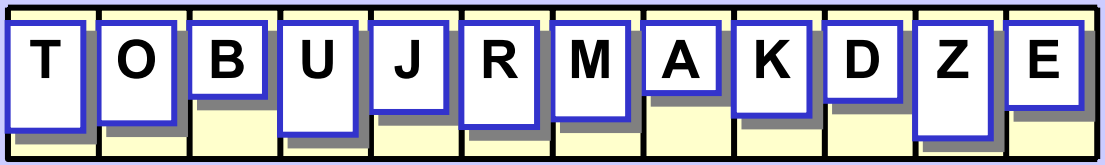


Bubble Sort

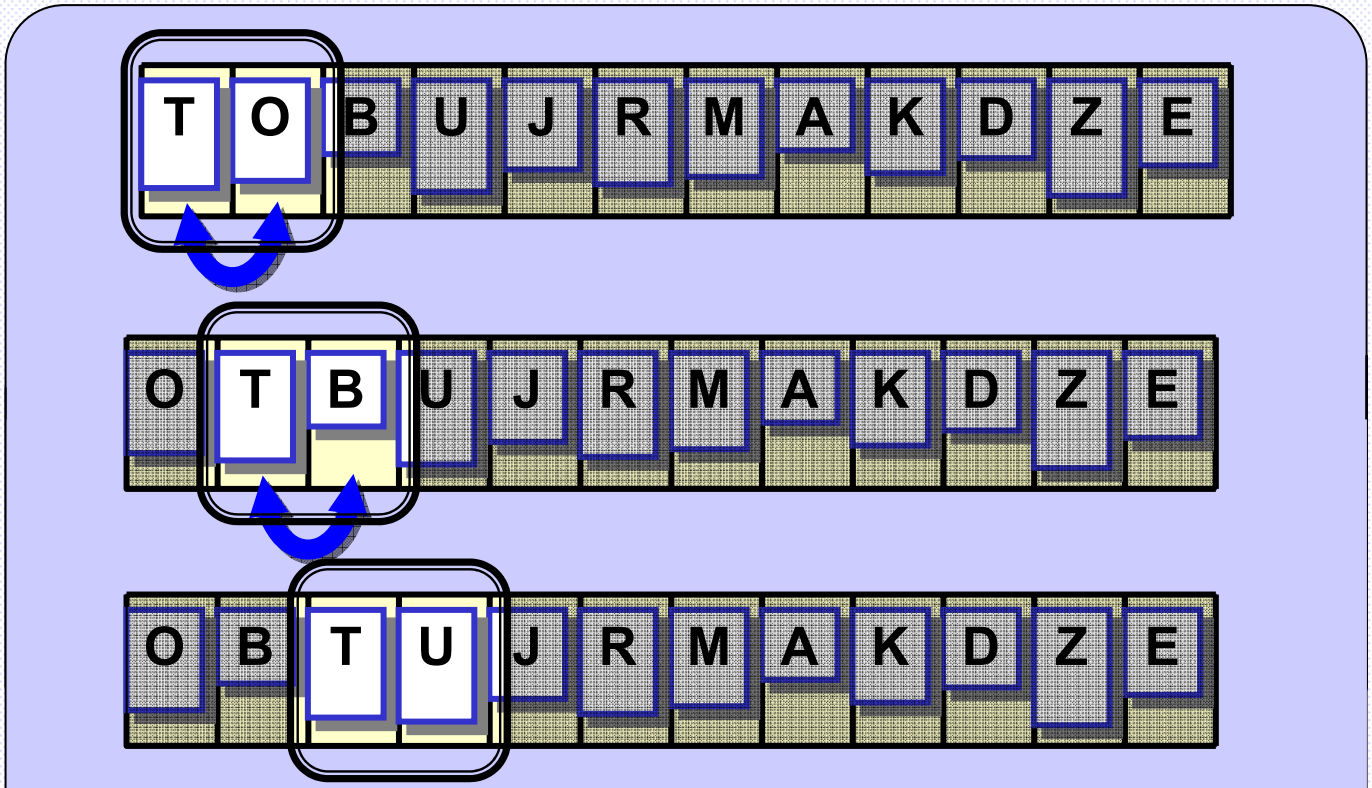
Bublinkové řazení

Bubble Sort

Start

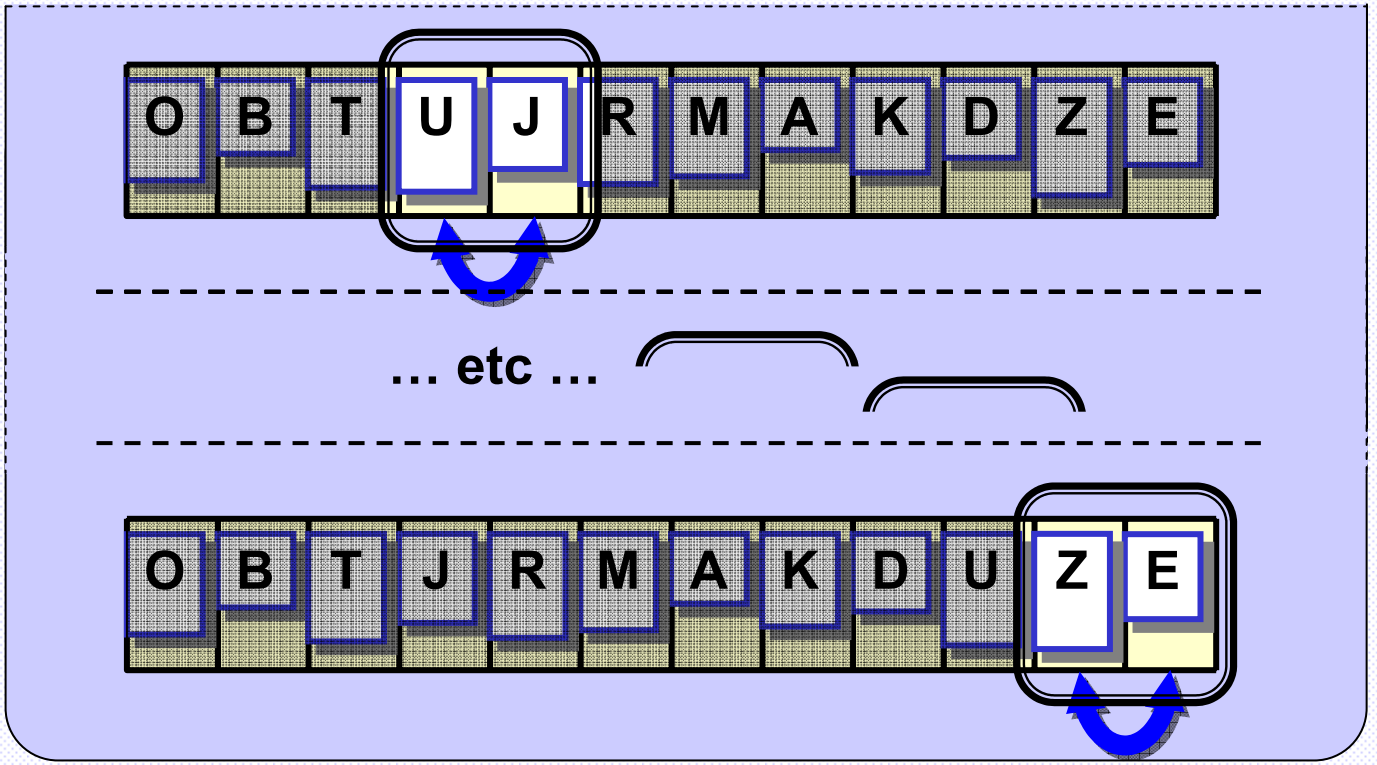


Fáze 1

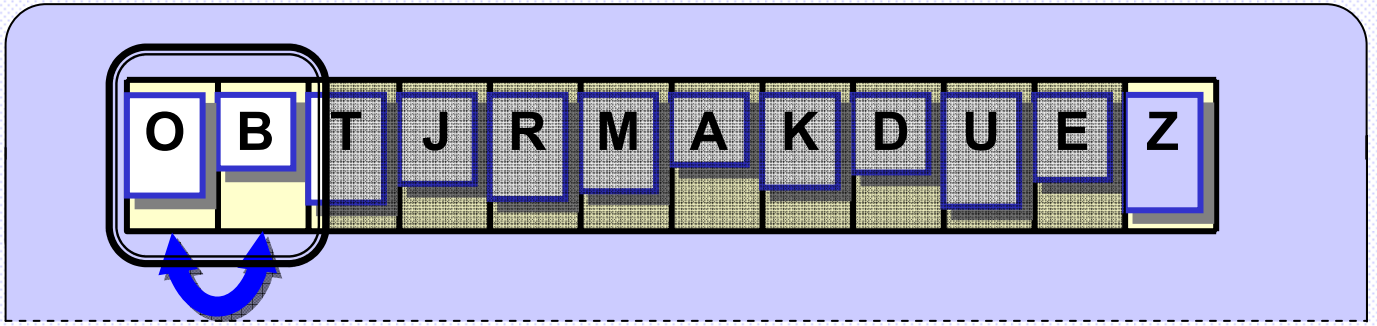


Bubble Sort

Fáze 1

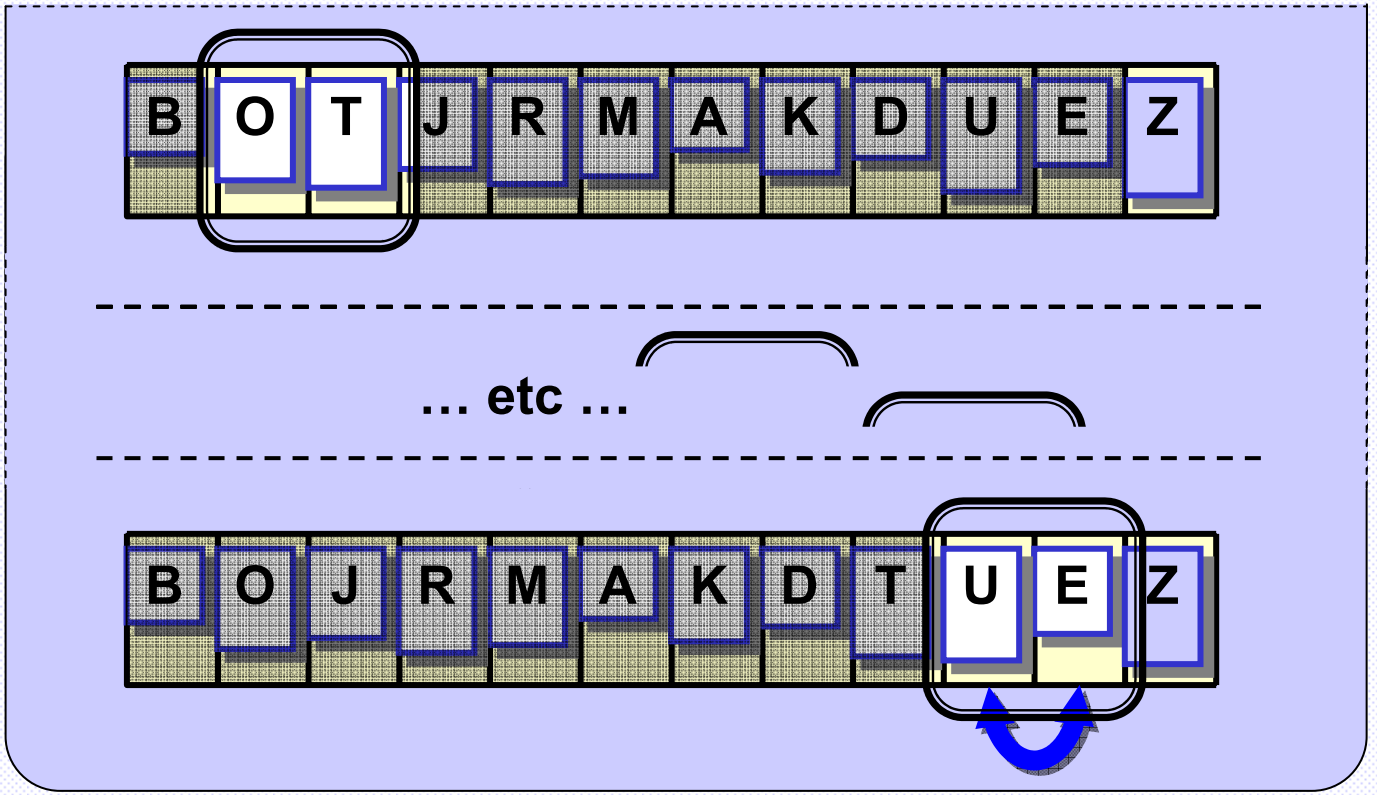


Fáze 2

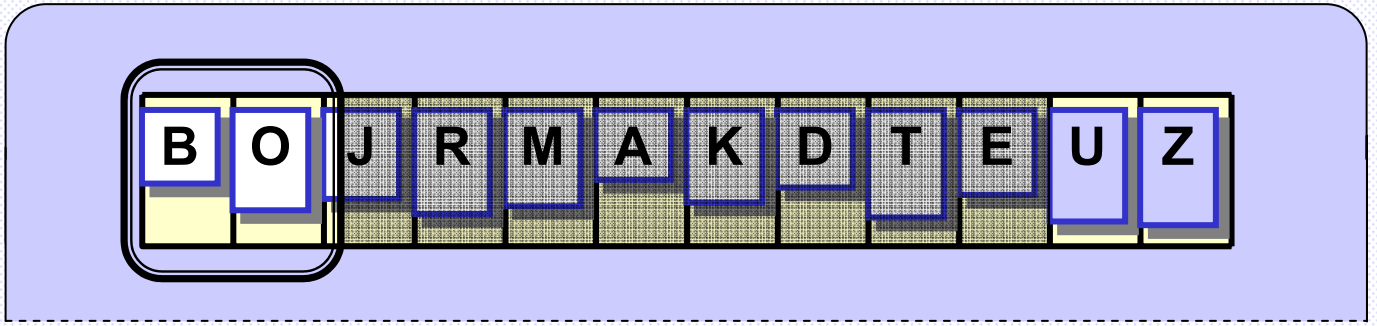


Bubble Sort

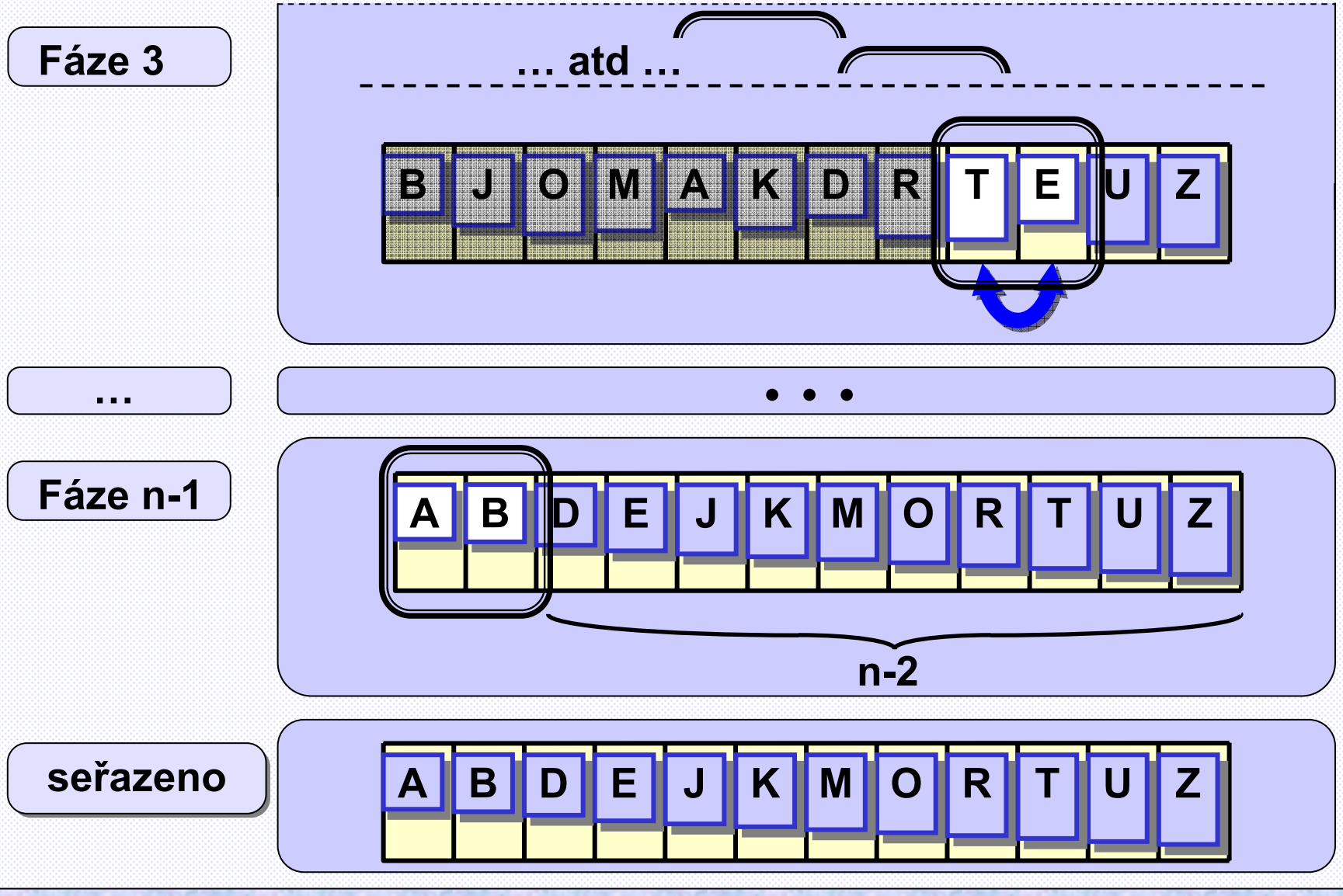
Fáze 2



Fáze 3



Bubble Sort


Fáze 3
… atd …
B J O M A K D R T E U Z
…
• • •
Fáze n-1
A B D E J K M O R T U Z
n-2
seřazeno
A B D E J K M O R T U Z

Bubble Sort

```
for (lastPos = n-1; lastPos > 0; lastPos--)
  for (j = 0; j < lastPos; j++)
    if (a[j] > a[j+1]) swap(a, j, j+1);
```

Shrnutí

Celkem testů

$$(n-1) + (n-2) + \ldots + 2 + 1 = \frac{1}{2}(n^2 - n) = \Theta(n^2)$$

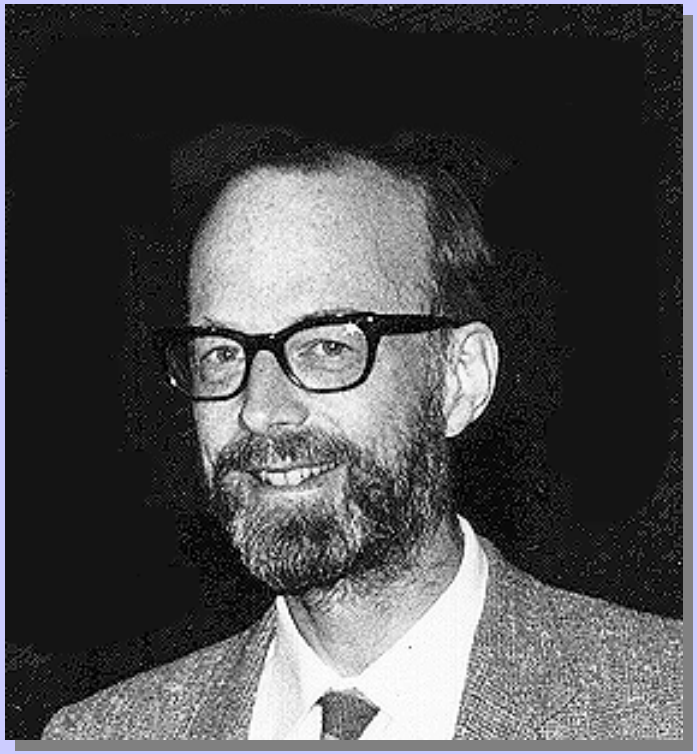
Celkem přesunů

$0 = \Theta(1)$ nejlepší případ

$\frac{1}{2}(n^2 - n) = \Theta(n^2)$ nejhorší případ

Asymptotická složitost Bubble Sortu je $\Theta(n^2)$

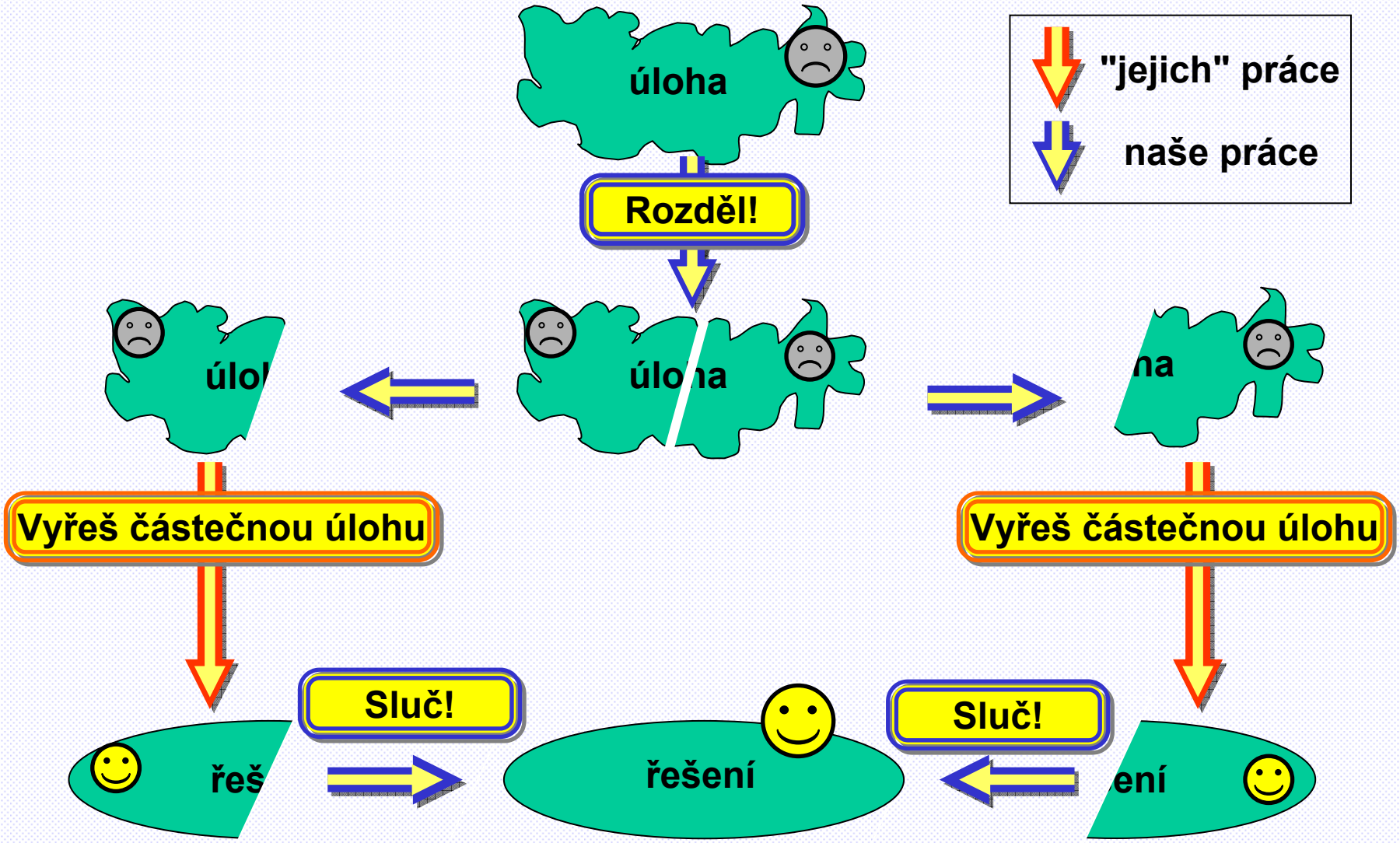
QuickSort



Sir Charles Antony Richard Hoare

C. A. R. Hoare: Quicksort. Computer Journal, Vol. 5, 1, 10-15 (1962)

Rozděl a Panuj! Divide & Conquer! Divide et Impera!

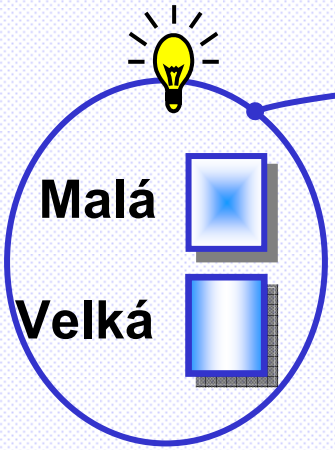


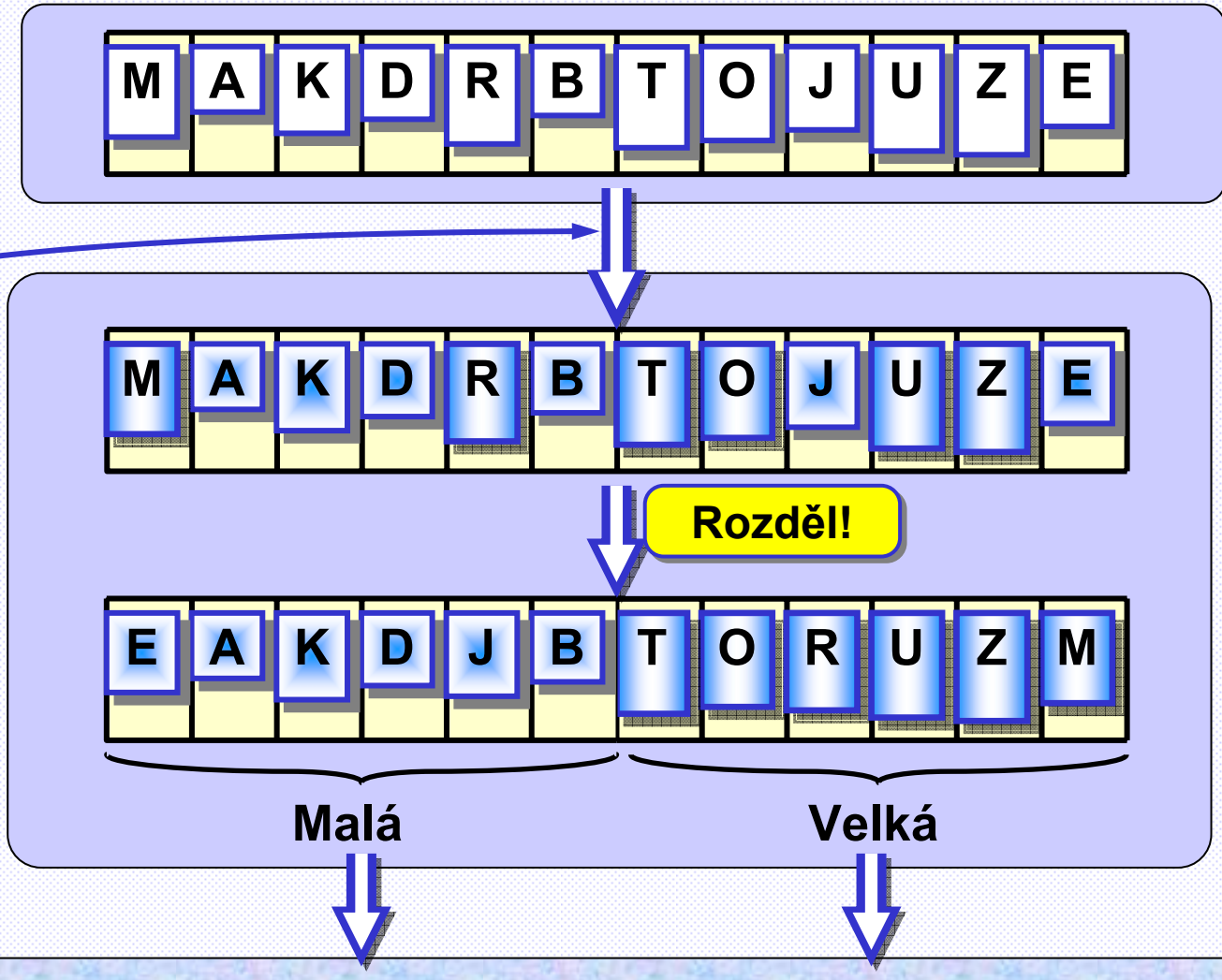
QuickSort

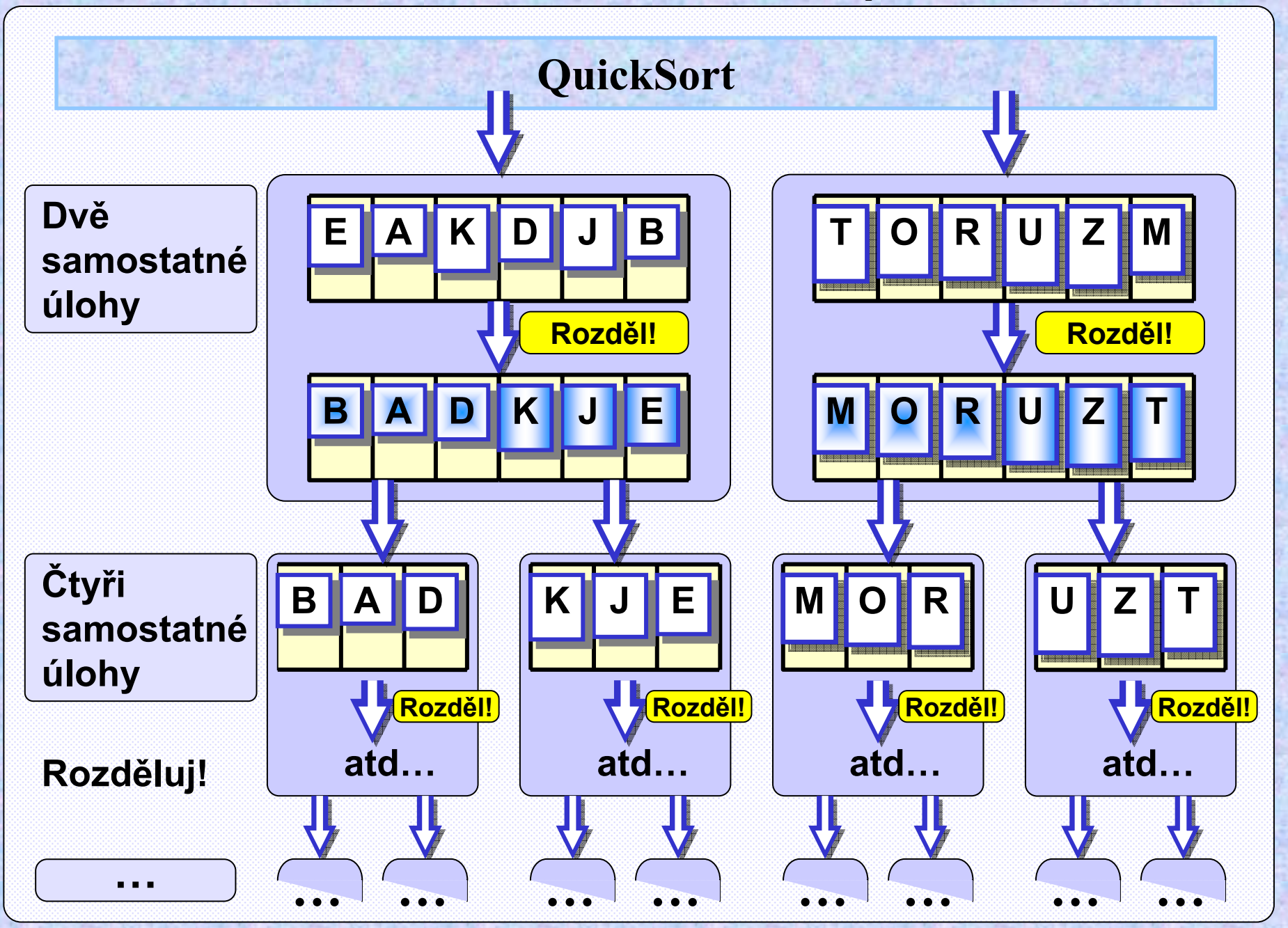
Myšlenka

Divide & Conquer!

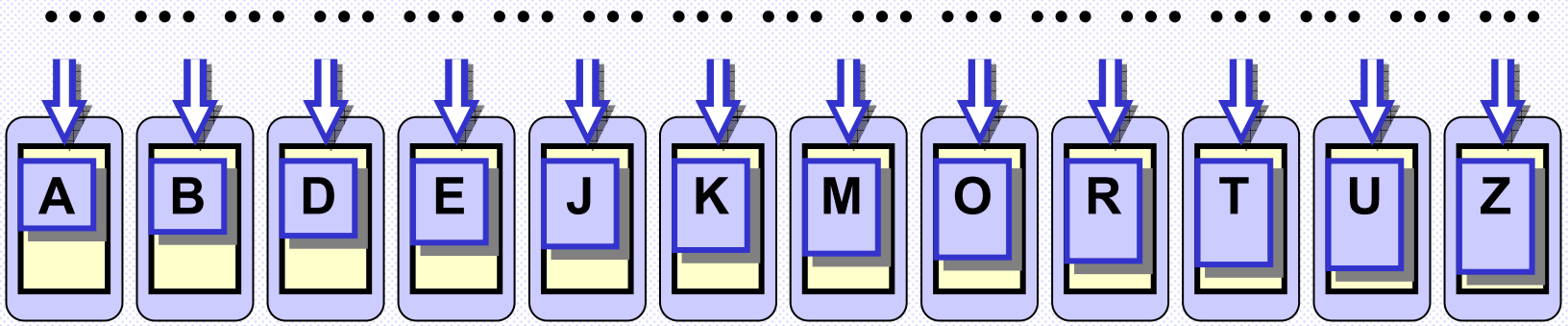
Start






QuickSort
Dvě samostatné úlohy
E A K D J B
Rozděl!
B A D K J E
T O R U Z M
Rozděl!
M O R U Z T
Čtyři samostatné úlohy
B A D
K J E
M O R
U Z T
Rozděl!
atd…
Rozděluj!
…

QuickSort

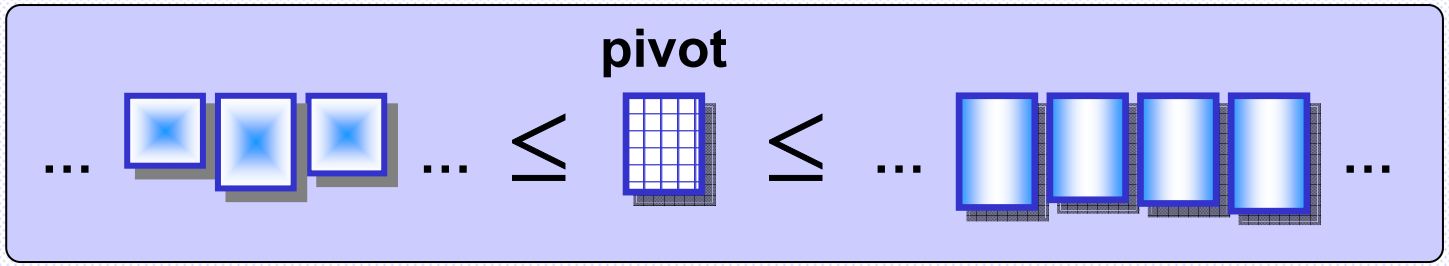


Opanováno!

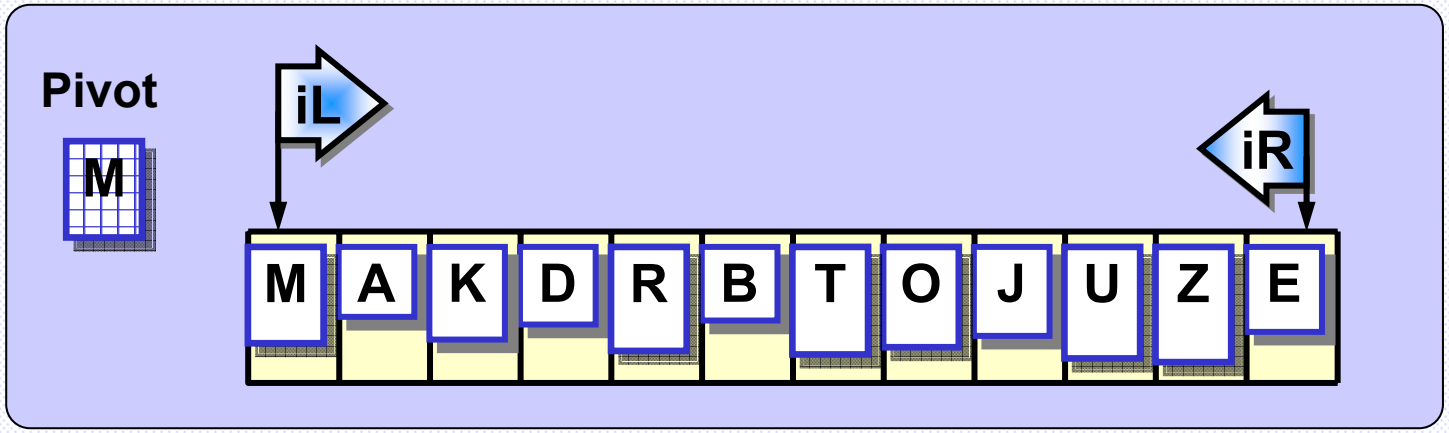
QuickSort

Dělení

pivot



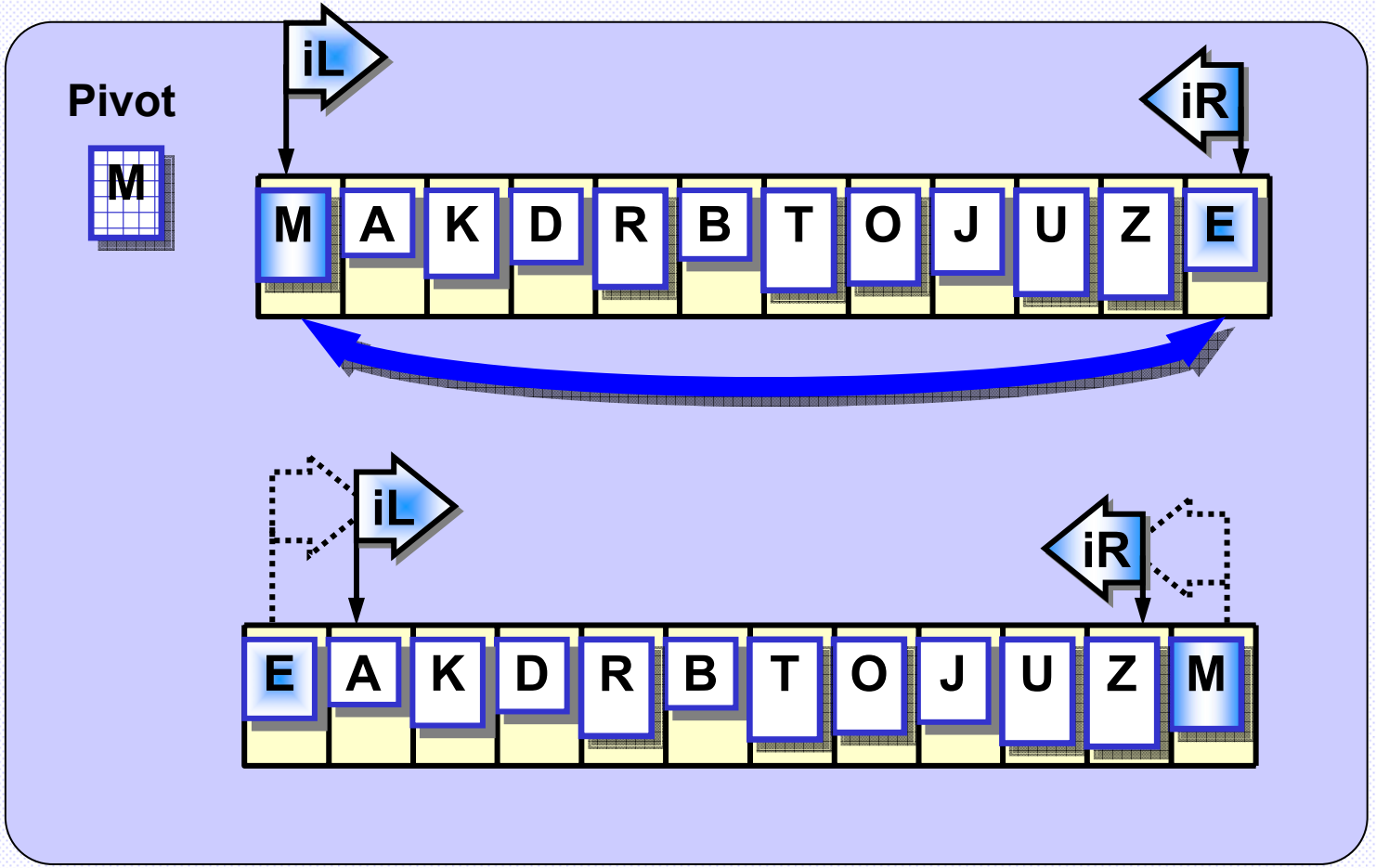
Init



QuickSort

Dělení

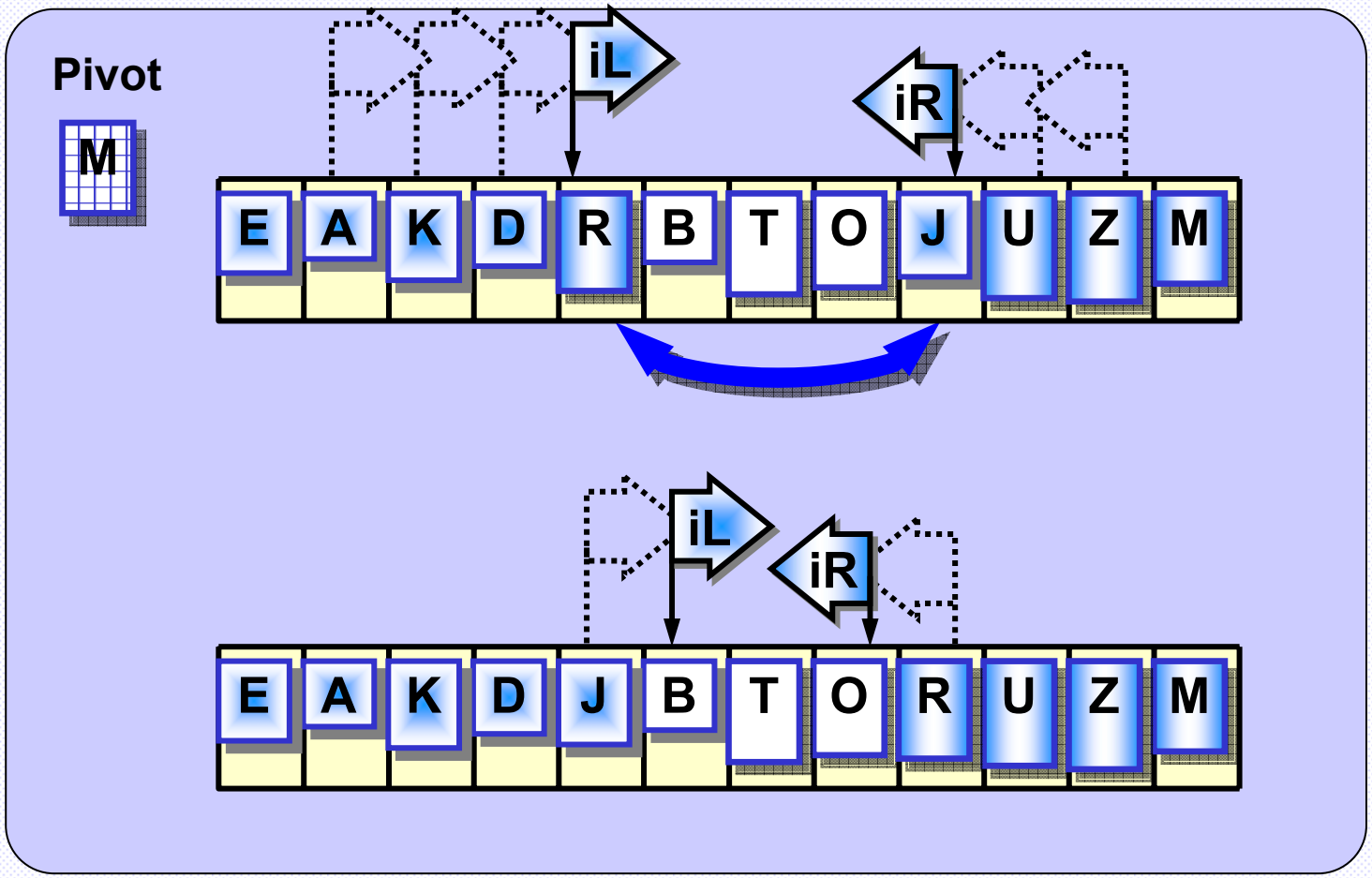
Krok 1



QuickSort

Dělení

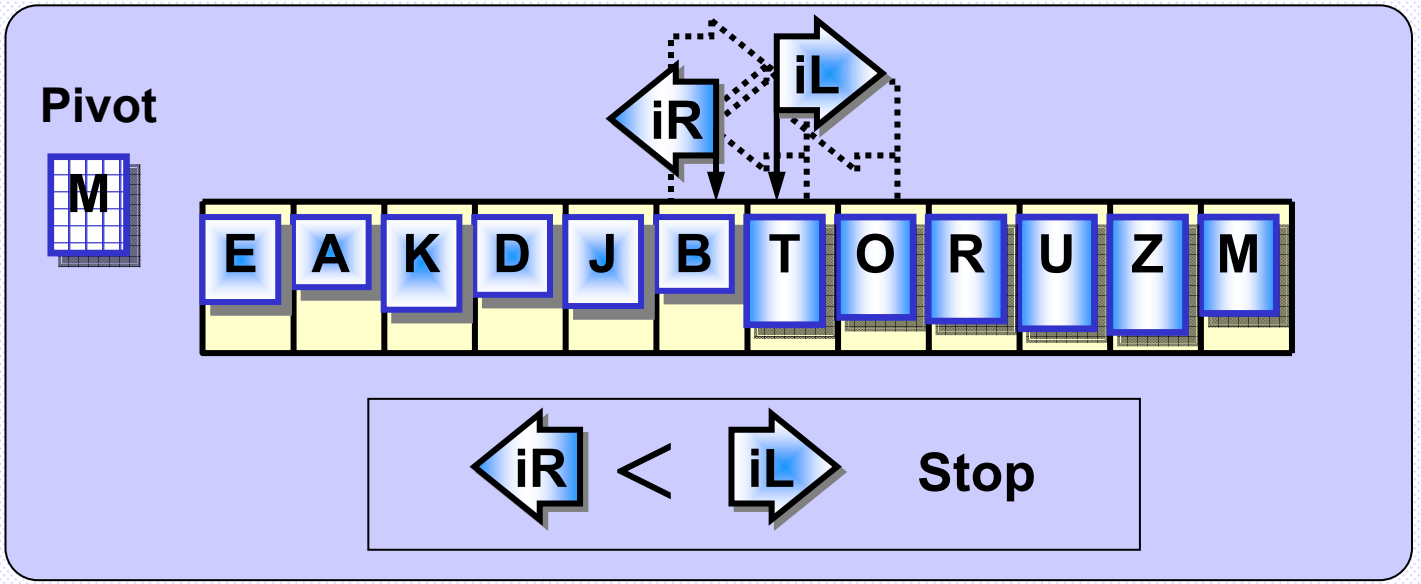
Krok 2



QuickSort

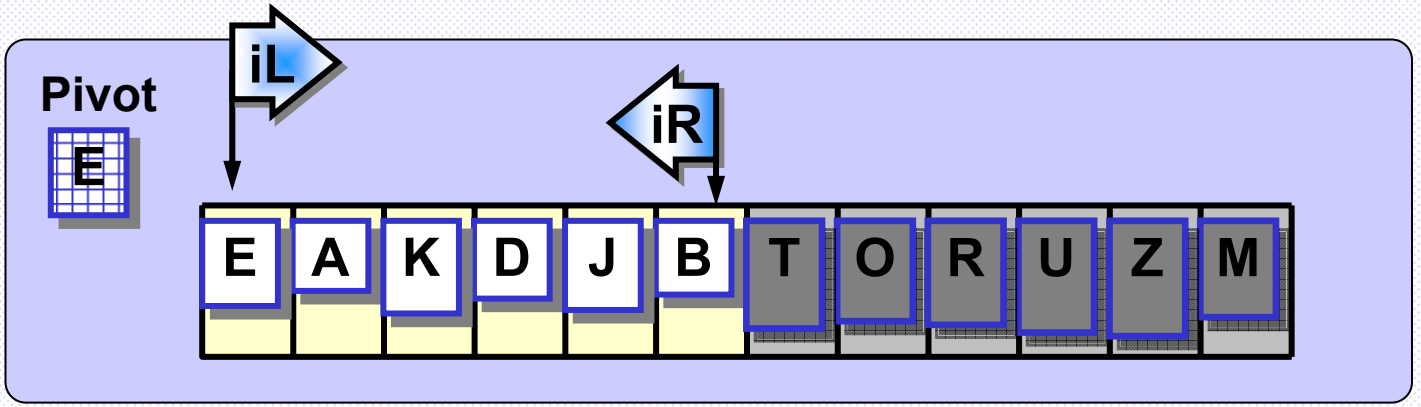
Dělení

Krok 3



Rozděl!

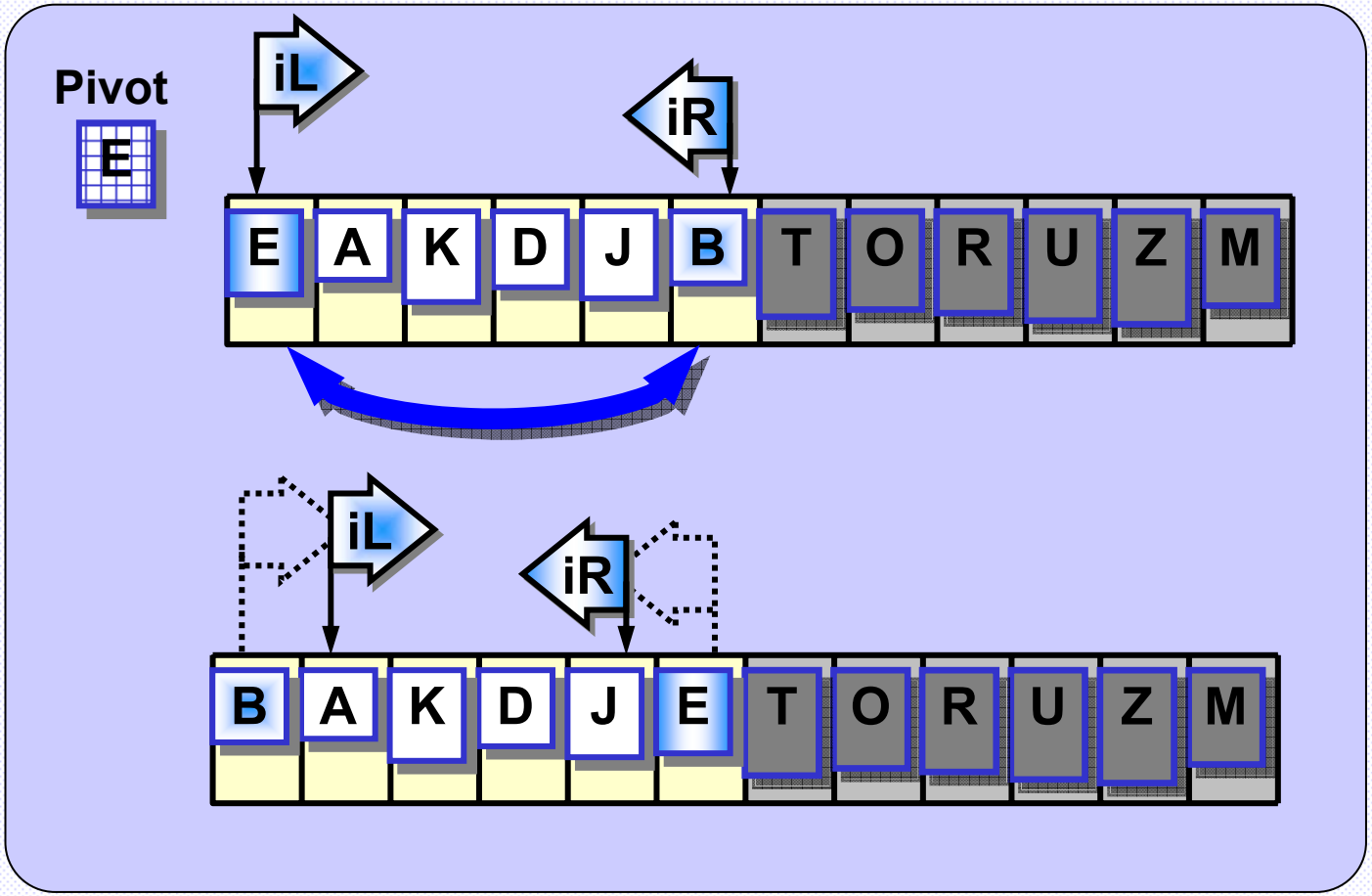
Init



QuickSort

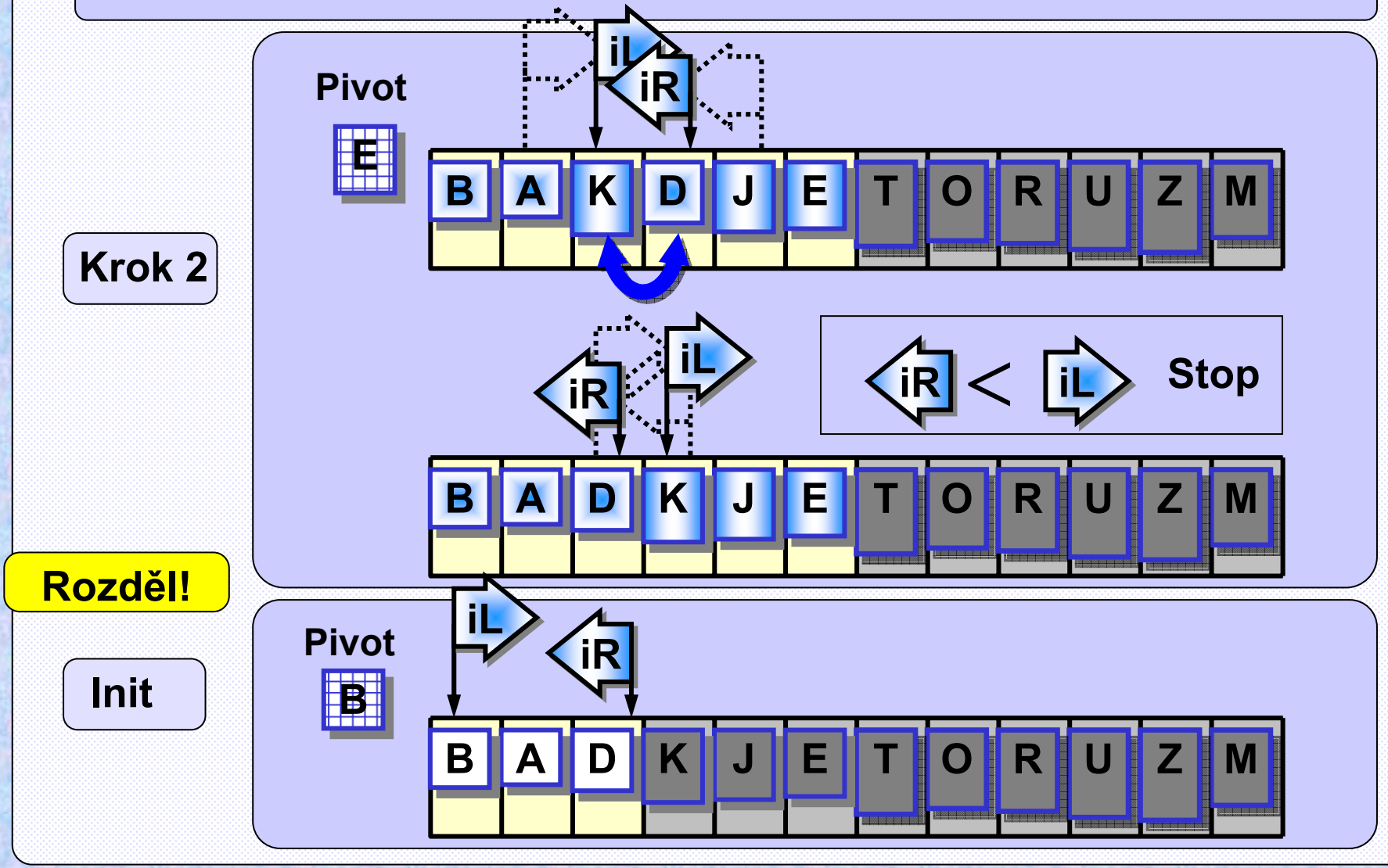
Dělení

Krok 1



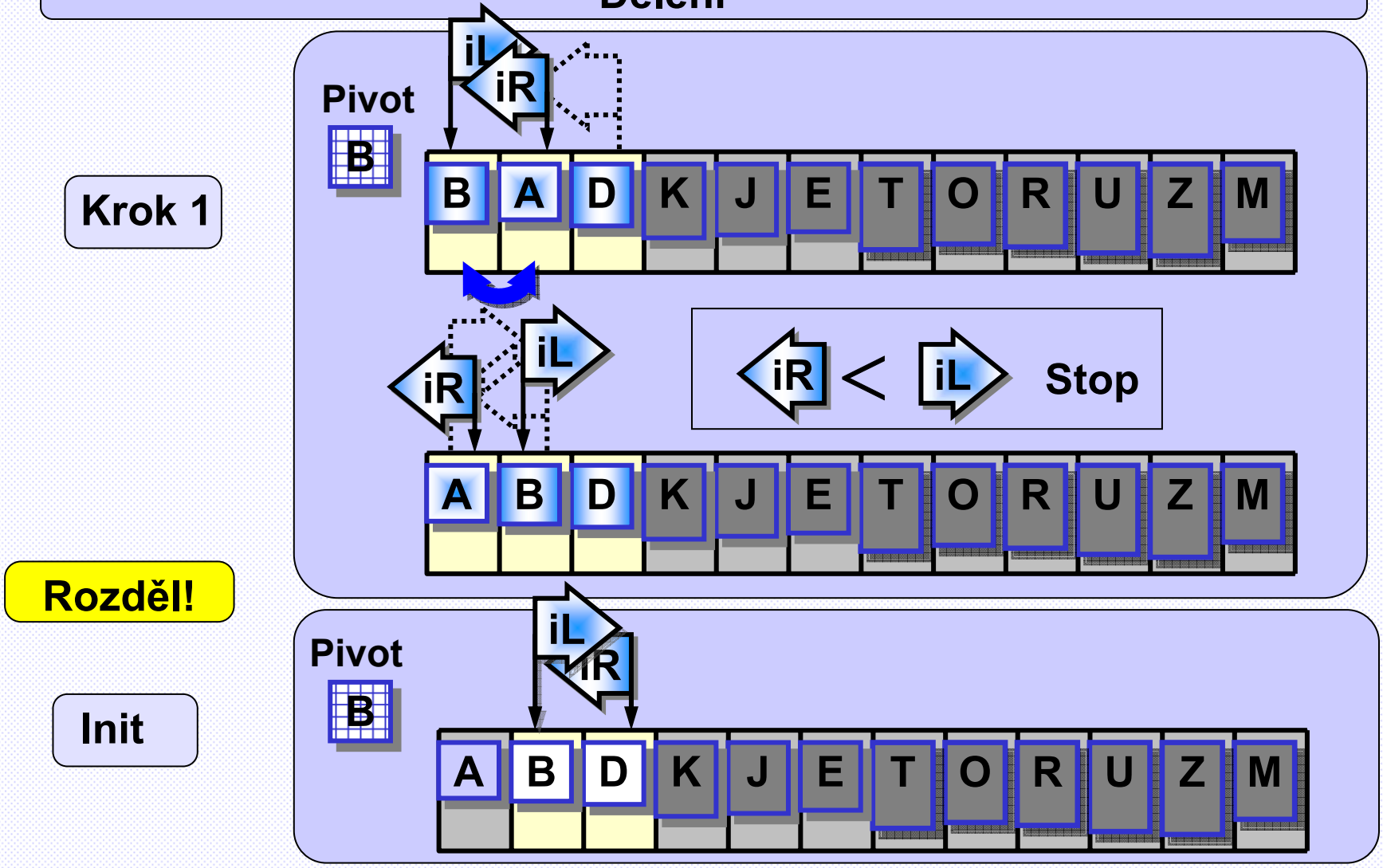
QuickSort

Dělení



QuickSort

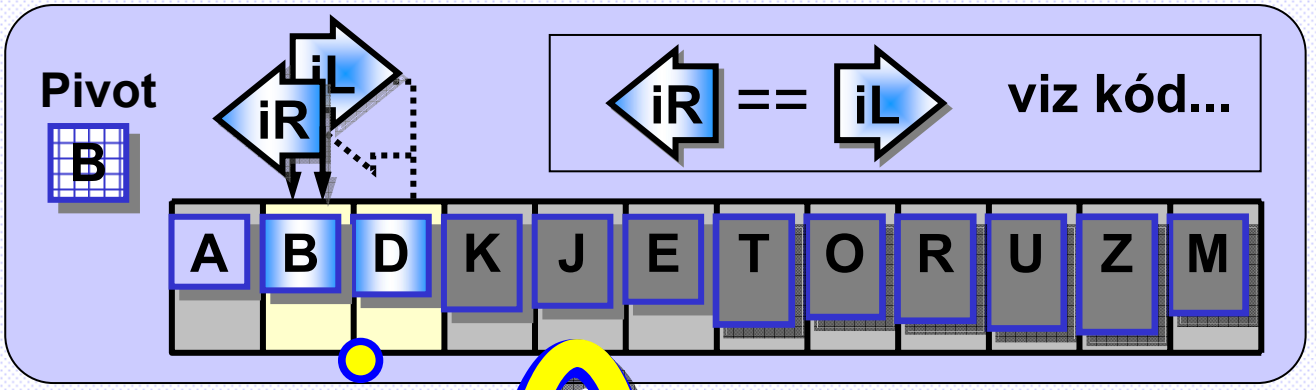
Dělení



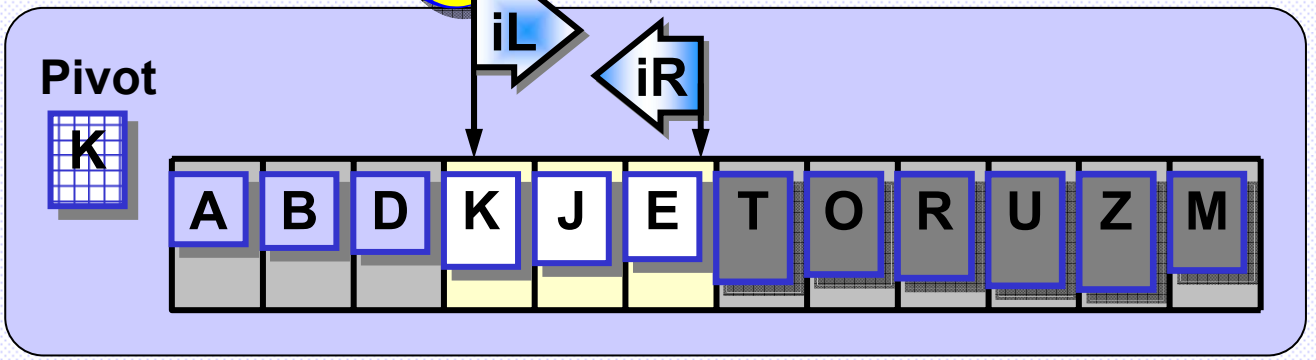
QuickSort

Dělení

Krok 1



Další oddíl



Init

atd…

atd…

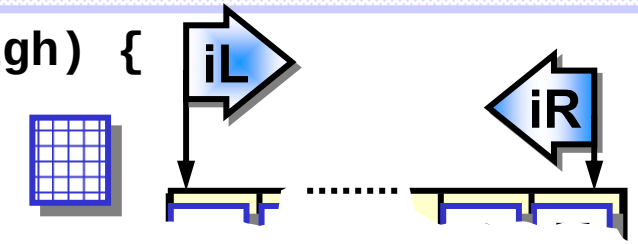
QuickSort

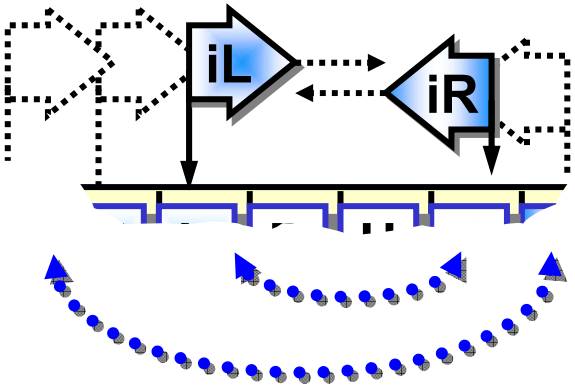
```
void qSort(Item a[], int low, int high) {
  int iL = low, iR = high;
  Item pivot = a[low];

  do {
    while (a[iL] < pivot) iL++;
    while (a[iR] > pivot) iR--;
    if (iL < iR) {
      swap(a,iL, iR);
      iL++; iR--;
    }
    else
      if (iL == iR) { iL++; iR--;}

  } while( iL <= iR);

  if (low < iR)  qSort(a, low, iR);
  if (iL < high) qSort(a, iL, high);
}
```





Rozděl!

QuickSort

Levý index se nastaví na začátek zpracovávaného úseku pole, pravý na jeho konec, zvolí se pivot.

<u>Cyklus (rozdělení na „malé" a „velké") :</u>

Levý index se pohybuje doprava
a zastaví se na prvku vetším nebo rovném pivotovi.
Pravý index se pohybuje doleva
a zastaví se na prvku menším nebo rovném pivotovi.
Pokud je levý index ještě před pravým,
příslušné prvky se prohodí,
a oba indexy se posunou o 1 ve svém směru.
Jinak pokud se indexy rovnají,
jen se oba posunou o 1 ve svém směru.

Cyklus se opakuje, dokud se indexy neprekříží,
tj. pravý se dostane pred levého.

<u>Následuje rekurzivní volání</u> (zpracování „malých" a „velkých" zvlášť)
na úsek od začátku do pravého(!) indexu včetně
a na úsek od levého(!) indexu včetně až do konce,
má-li príslušný úsek délku větší než 1.

QuickSort

Asymptotická složitost

Celkem přesunů a testů

$\Theta(n \cdot \log_2(n))$	nejlepší případ
$\Theta(n \cdot \log_2(n))$	průměrný případ
$\Theta(n^2)$	nejhorší případ

Asymptotická složitost Quick Sortu je $O(n^2)$, ...

… ale! :

"Očekávaná" složitost Quick Sortu je $\Theta(n \cdot \log_2(n))$ (!!)

Sorting

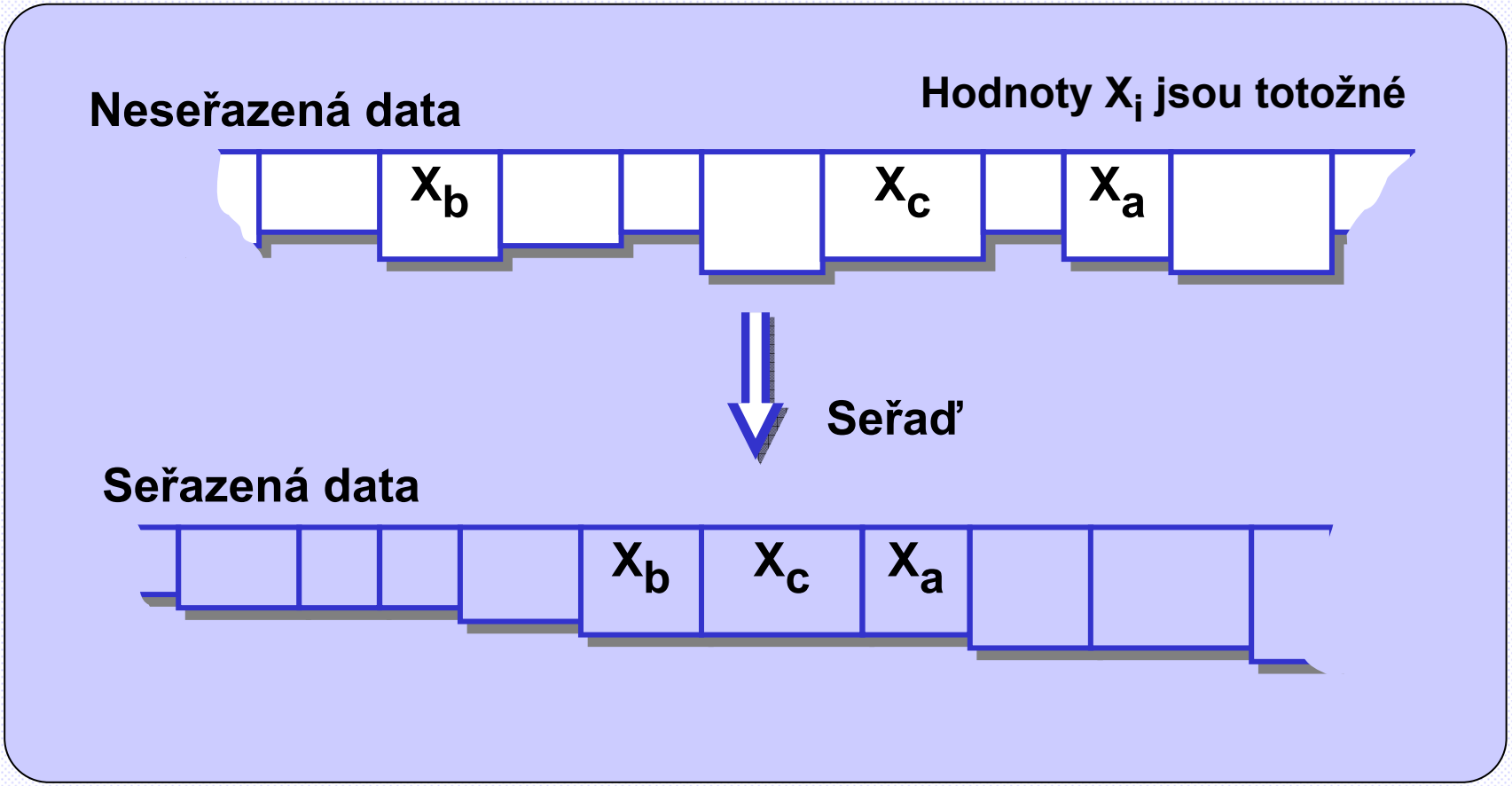
Porovnání efektivity

N	N^2	$N \times \log_2(N)$	$\frac{N^2}{N \times \log_2(N)}$	zpoma-lení (1~1sec)
1	1	0		
10	100	33.2	3.0	3 sec
100	10 000	6 64.4	15.1	15 sec
1 000	1 000 000	9 965.8	100.3	1.5 min
10 000	100 000 000	132 877.1	752.6	13 min
100 000	10 000 000 000	1 660 964.0	6 020.6	1.5 hod
1 000 000	1 000 000 000 000	19 931 568.5	50 171.7	14 hod
10 000 000	100 000 000 000 000	232 534 966.6	430 042.9	5 dnů

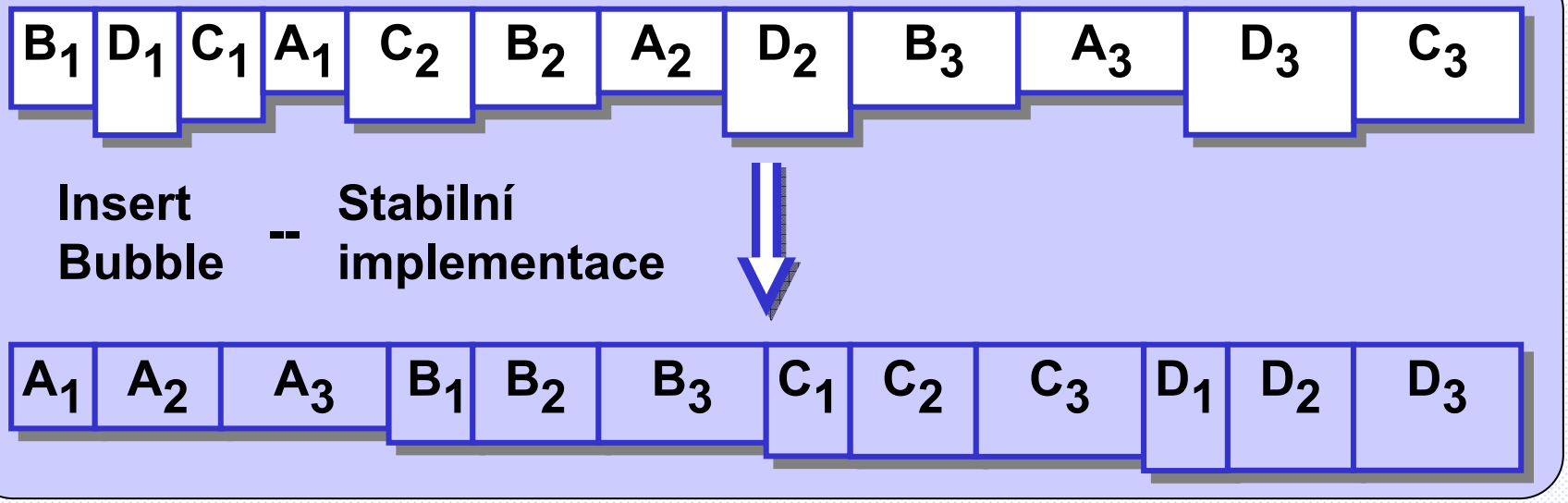
tab. 1

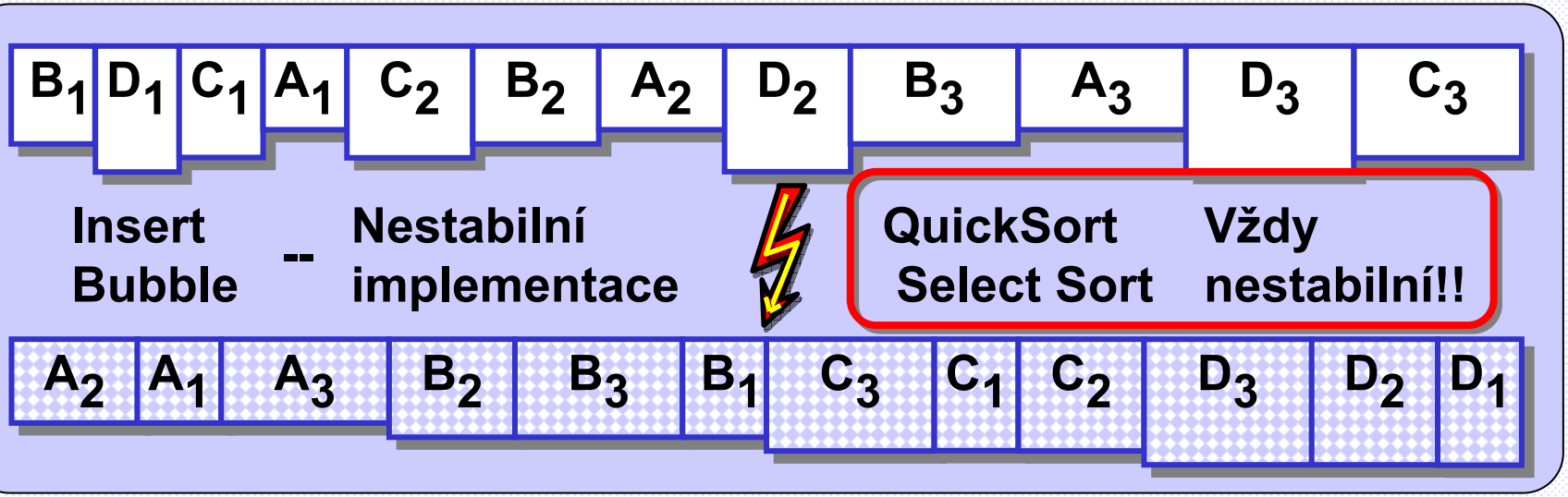
Stabilita řazení

Stabilní řazení nemění pořadí prvků se stejnou hodnotou.



Stabilita řazení





Různé algoritmy mají různou složitost

Algoritmus a program není totéž