

# Программирование на языке Java

- 24. Сортировка массивов
- 25. Поиск в массиве

# **Программирование на языке Java**

## **Тема 24. Сортировка массивов**

# Сортировка

**Сортировка** – это расстановка элементов массива в заданном порядке (по возрастанию, убыванию, последней цифре, сумме делителей, ...).

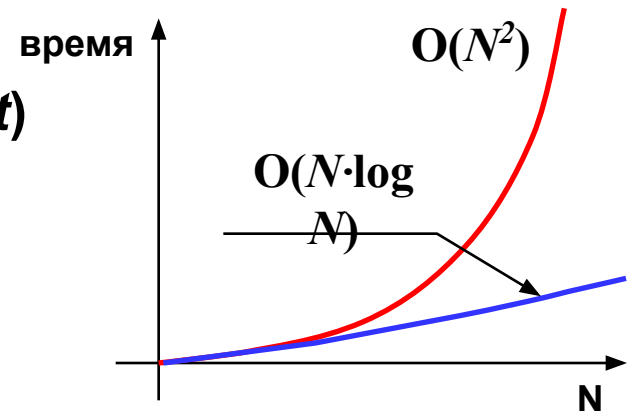
**Задача:** переставить элементы массива в порядке возрастания.

## Алгоритмы:

- простые и понятные, но неэффективные для больших массивов
  - **метод пузырька**
  - **метод выбора**
- сложные, но эффективные
  - **«быстрая сортировка» (*Quick Sort*)**
  - **сортировка «кучей» (*Heap Sort*)**
  - **сортировка слиянием**
  - **пирамидальная сортировка**

сложность  $O(N^2)$

сложность  $O(N \cdot \log N)$



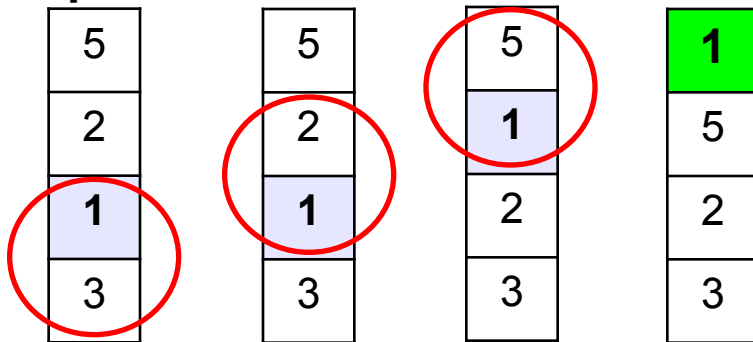
# Метод пузырька

**Идея** – пузырек воздуха в стакане воды поднимается со дна вверх.

**Для массивов** – самый маленький («легкий») элемент перемещается вверх («всплывает»).

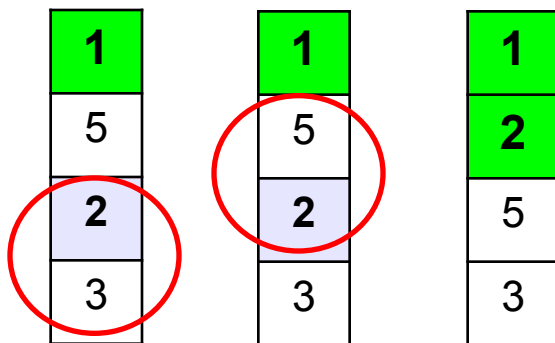
**1-ый**

**проход**

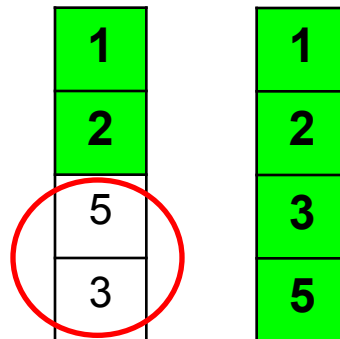


- начиная снизу, сравниваем два соседних элемента; если они стоят «неправильно», меняем их местами
- за 1 проход по массиву **один** элемент (самый маленький) становится на свое место

**2-ой проход**



**3-ий проход**



Для сортировки массива из  $N$  элементов нужен  $N-1$  проход (достаточно поставить на свои места  $N-1$  элементов).

# Программа (1-ый проход)

|     |     |
|-----|-----|
| 0   | 5   |
| 1   | 2   |
| ... | ... |
| N-2 | 6   |
| N-1 | 3   |

сравниваются пары

$A[N-2]$  и  $A[N-1]$ ,

$A[N-3]$  и  $A[N-2]$

...

$A[0]$  и  $A[1]$

$A[j]$  и  $A[j+1]$

```
for ( j = N-2; j >= 0 ; j-- )
    if ( A[j] > A[j+1] ) {
        c = A[j];
        A[j] = A[j+1];
        A[j+1] = c;
    }
```

# Программа (следующие проходы)

2-ой  
проход

|     |     |
|-----|-----|
| 0   | 1   |
| 1   | 5   |
| ... | ... |
| N-2 | 3   |
| N-1 | 6   |

(i+1)-ый  
проход



**A[0] уже на своем месте!**

```
for ( j = N-2; j >= 1; j-- )
    if ( A[j] > A[j+1] ) {
        c = A[j];
        A[j] = A[j+1];
        A[j+1] = c;
    }
```

```
for ( j = N-2; j >= i; j-- )
    ...
```

# Программа

```
public static void main(String[] args)
```

```
{
```

```
    int N = 10;
```

```
    int A[N], i, j, c;
```

```
    // заполнить массив
```

```
    // вывести исходный массив
```

```
    for (i = 0; i < N-1; i++) {
```

```
        for (j = N-2; j >= i; j--)
```

```
            if (A[j] > A[j+1]) {
```

```
                c = A[j];
```

```
                A[j] = A[j+1];
```

```
                A[j+1] = c;
```

```
            }
```

```
        }
```

```
    // вывести полученный массив
```

```
}
```



Почему цикл для  $i < N-1$ ,  
а не  $i < N$ ?

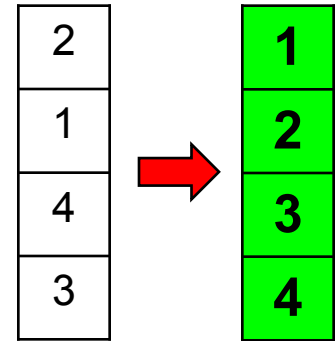
элементы выше  
 $A[i]$  уже  
поставлены

меняем  $A[j]$   
и  $A[j+1]$

# Метод пузырька с флажком

**Идея** – если при выполнении метода пузырька не было обменов, массив уже отсортирован и остальные проходы не нужны.

**Реализация:** переменная-флаг, показывающая, был ли обмен; если она равна **false**, то выход.



```
do {
    boolean flag;
    flag = false; // сбросить флаг
    for (j = N-2; j >= 0; j --)
        if (A[j] > A[j+1]) {
            c = A[j];
            A[j] = A[j+1];
            A[j+1] = c;
            flag = true; // поднять флаг
        }
    }
    while (flag); // выход при flag = false
```



Как улучшить?



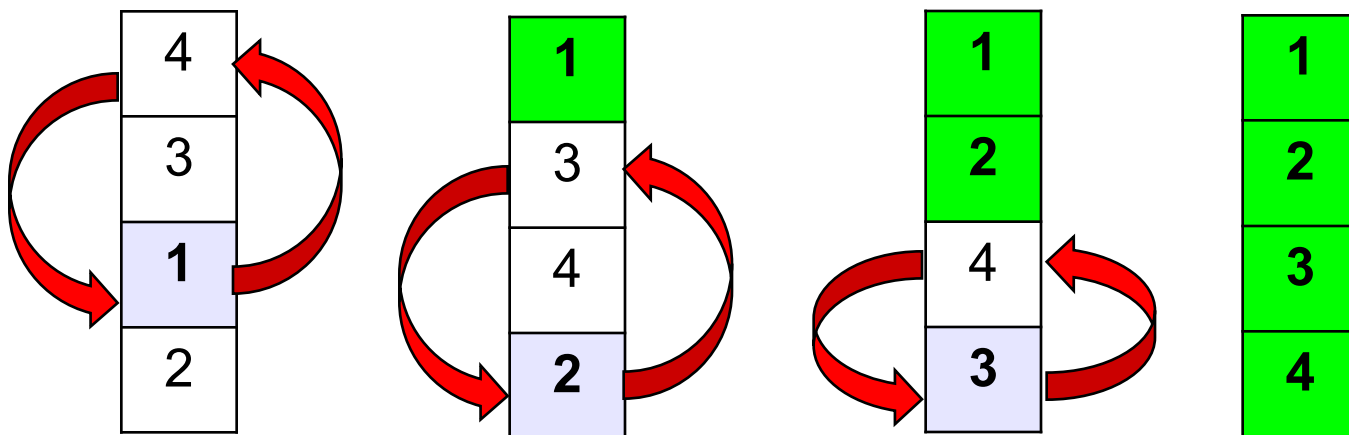
# Метод пузырька с флажком

```
i = 0;  
do {  
    flag = false; // сбросить флаг  
    for ( j = N-2; j >= i; j -- )  
        if ( A[j] > A[j+1] ) {  
            c = A[j];  
            A[j] = A[j+1];  
            A[j+1] = c;  
            flag = true; // поднять флаг  
        }  
    i ++;  
} while ( flag ); // выход при flag = false
```

# Метод выбора

## Идея:

- найти минимальный элемент и поставить на первое место (поменять местами с  $A[0]$ )
- **из оставшихся** найти минимальный элемент и поставить на второе место (поменять местами с  $A[1]$ ), и т.д.



# Метод выбора

нужно  $N-1$  проходов

```
for ( i = 0; i <  $N-1$ ; i++ ) {  
    nMin = i;  
    for ( j = i+1; j < N; j++ )  
        if ( A[j] < A[nMin] ) nMin = j;  
    if ( nMin != i ) {  
        c = A[i];  
        A[i] = A[nMin];  
        A[nMin] = c;  
    }  
}
```

поиск минимального  
от  $A[i]$  до  $A[N-1]$

если нужно,  
переставляем



Можно ли убрать if?

# Задания

---

**Задача 1:** Заполнить массив из 10 элементов случайными числами в интервале  $[0..100]$  и отсортировать его по последней цифре.

**Пример:**

Исходный массив:

14 25 13 30 76 58 32 11 41 97

Результат:

30 11 41 32 13 14 25 76 97 58

**Задача 2:** Заполнить массив из 10 элементов случайными числами в интервале  $[0..100]$  и отсортировать первую половину по возрастанию, а вторую – по убыванию.

**Пример:**

Исходный массив:

14 25 13 30 76 | 58 32 11 41 97

Результат:

13 14 25 30 76 | 97 58 41 32 11

# Формирование массива по условию

**Задача** – найти в массиве элементы, удовлетворяющие некоторому условию (например, отрицательные), и скопировать их в другой массив.

**Примитивное решение:**

```
int N = 5;  
int A[N], B[N];  
// здесь заполнить массив A  
for( i=0; i<N; i++)  
    if( A[i] < 0 ) B[i] = A[i];
```

|   | А  |   | В  |
|---|----|---|----|
| 0 | 1  |   | 0  |
| 1 | -5 | → | -5 |
| 2 | 3  |   | 0  |
| 3 | -2 | → | -2 |
| 4 | 5  |   | 0  |



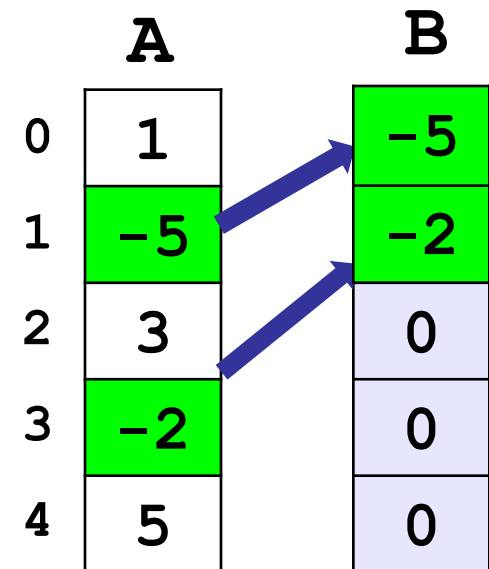
- выбранные элементы не рядом, не в начале массива
- непонятно, как с ними работать

# Формирование массива по условию

**Решение:** ввести счетчик найденных элементов `count`, очередной элемент ставится на место `B[count]`.

```
int A[N], B[N], count = 0;
// здесь заполнить массив A
for( i=0; i < N; i++ )
    if( A[i] < 0 ) {
        B[count] = A[i];
        count ++;
    }

// вывод массива B
for( i=0; i < count; i++ )
    System.out.printf(
"%d\n", B[i]);
```



# Задания

---

**Задача 1:** Заполнить массив случайными числами и отобразить в другой массив все числа, у которых вторая с конца цифра (число десятков) – ноль.

**Пример:**

**Исходный массив:**

40    105    203    1    14

**Результат:**

105    203    1

**Задача 2:** Заполнить массив случайными числами и выделить в другой массив все числа, которые встречаются более одного раза.

**Пример:**

**Исходный массив:**

4    1    2    1    11    2    34

**Результат:**

1    2

# Программирование на языке Java

## Тема 25. Поиск в массиве



# Поиск в массиве

---

**Задача** – найти в массиве элемент, равный **X**, или установить, что его нет.

**Решение:** для произвольного массива: **линейный поиск** (перебор)

недостаток: **низкая скорость**

**Как ускорить?** – заранее подготовить массив для поиска

- как именно подготовить?
- как использовать «подготовленный» массив?

# Линейный поиск

nX – номер нужного  
элемента в массиве

```
nX = -1; // пока не нашли ...  
for ( i = 0; i < N; i ++ ) // цикл по всем элементам  
    if ( A[i] == X )        // если нашли, то ...  
        nX = i;            // ... запомнили номер  
  
if ( nX < 0 ) System.out.printf("Не нашли...");  
else         System.out.printf("A[%d]=%d", nX, X);
```

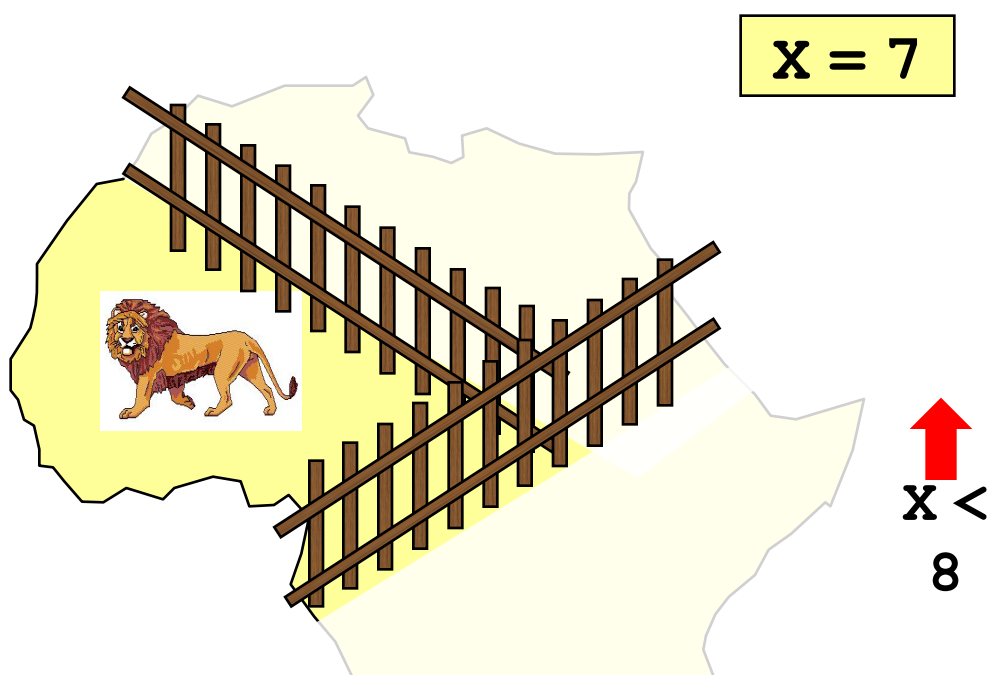


Что можно улучшить?

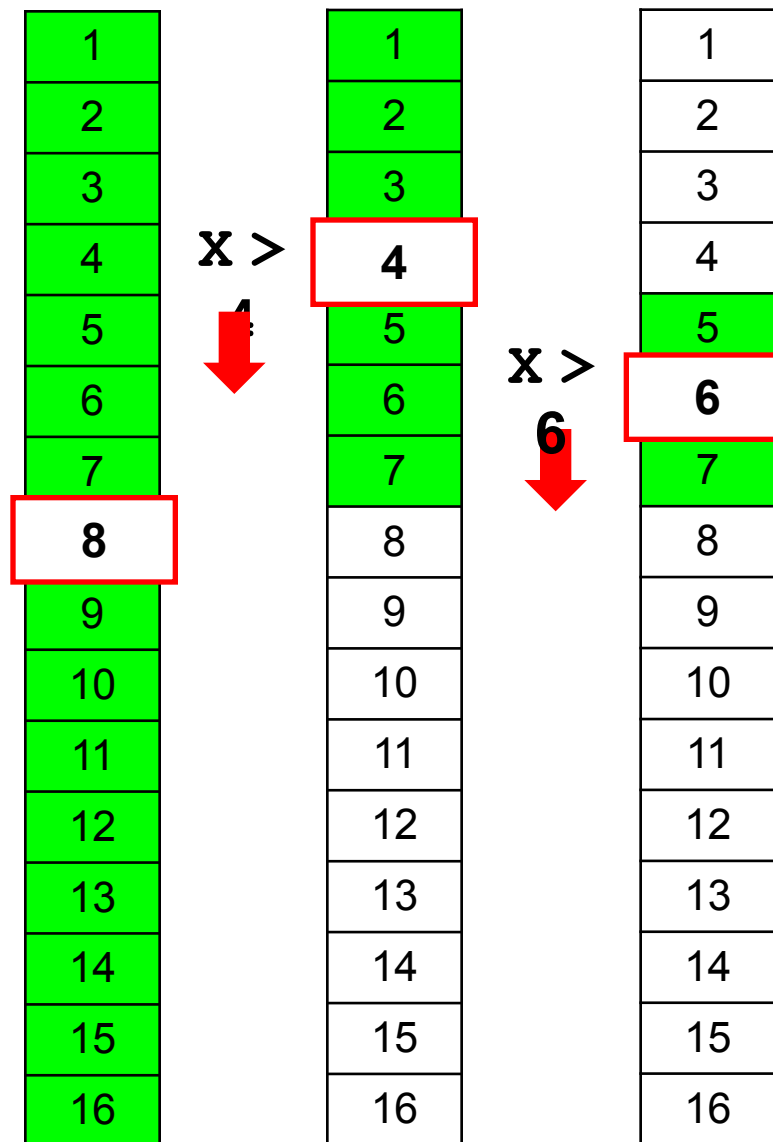
**Улучшение:** после  
того, как нашли X,  
выходим из цикла.

```
nX = -1;  
for ( i = 0; i < N; i ++ )  
    if ( A[i] == X ) {  
        nX = i;  
        break // выход из цикла  
    }  
;
```

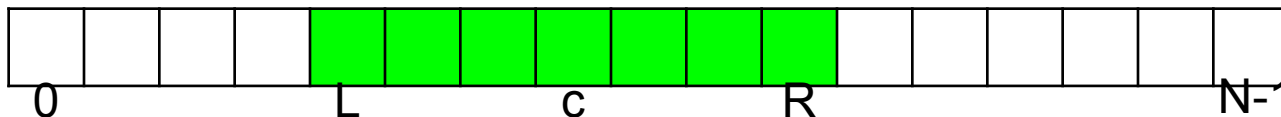
# Двоичный поиск



1. Выбрать средний элемент  $A[s]$  и сравнить с  $X$ .
2. Если  $X = A[s]$ , нашли (выход).
3. Если  $X < A[s]$ , искать дальше в первой половине.
4. Если  $X > A[s]$ , искать дальше во второй половине.



# ДВОИЧНЫЙ ПОИСК



```
nX = -1;  
L = 0; R = N-1; // границы: ищем от A[0] до A[N-1]
```

```
while ( R >= L ){  
    c = (R + L) / 2;  
    if (X == A[c]) {  
        nX = c;  
        break;  
    }
```

номер среднего элемента

если нашли ...

ВЫЙТИ ИЗ ЦИКЛА

```
    if (X < A[c]) R = c - 1;  
    if (X > A[c]) L = c + 1;  
}
```

сдвигаем  
границы

```
if (nX < 0) System.out.printf("Не нашли...");  
else      System.out.printf("A[%d]=%d", nX, X);
```



Почему нельзя `while ( R > L ) { ... } ?`

# Сравнение методов поиска

|                    | Линейный                   | Двоичный                              |
|--------------------|----------------------------|---------------------------------------|
| подготовка         | нет                        | <b>отсортировать</b>                  |
|                    | число шагов                |                                       |
| <b>N = 2</b>       | <b>2</b>                   | <b>2</b>                              |
| <b>N = 16</b>      | <b>16</b>                  | <b>5</b>                              |
| <b>N = 1024</b>    | <b>1024</b>                | <b>11</b>                             |
| <b>N = 1048576</b> | <b>1048576</b>             | <b>21</b>                             |
| <b>N</b>           | <b><math>\leq N</math></b> | <b><math>\leq \log_2 N + 1</math></b> |

# Задания

---

**Задача 1:** Написать программу, которая сортирует массив **ПО УБЫВАНИЮ** и ищет в нем элемент, равный  $X$  (это число вводится с клавиатуры).  
Использовать **двоичный поиск**.