Plan

2023-09-21

1. Homework Review
2. Bubble Sort
3. Binary search

---

1. Разбор домашнего задания
2. Сортировка пузырьком
3. Бинарный поиск

Theory

▶ **English**

▼ **На русском**

Сортировка пузырьком в Java

- начиная с начала массива просматриваем попарно по 2 элемента (первый со вторым, второй с третим, третий с четвертым и т.д.).
- Если второй элемент в паре меньше первого элемента – перемещаем его на место первого, а первый на место второго. Это мы делаем для всех элементов.

тоже самое, другими словами

1. Сравнить два элемента
2. Поменять местами или скопировать один из них
3. Перейти к следующему элементу

Принцип работы пузырьковой сортировки.

Пузырьковая сортировка считается самой простой, но перед тем как описывать этот алгоритм давайте представим

1. Вы перемещаетесь к нулевому элементу нашего массива;
2. Сравниваете нулевой элемент с первым;
3. Если элемент на нулевой позиции оказался больше, вы меняете их местами;
4. Иначе, если элемент на нулевой позиции меньше, вы оставляете их на своих местах;
5. Производите переход на позицию правее и повторяете сравнение

Общий принцип

Пузырьковая сортировка основана на идее "**всплытия**" наибольшего (или наименьшего) элемента массива к его **концу** (или началу). Для этого мы проходим по массиву, сравниваем пары соседних элементов и, если они не упорядочены, меняем их местами.

Пузырьковая сортировка в Java: Детальный разбор

Мы хотим отсортировать массив по возрастанию

Исходный массив

Начинаем с массива `[5, 2, -3, -10]`.

Общая логика

Алгоритм пузырьковой сортировки работает, переставляя соседние элементы, если они расположены не по порядку.

Первый проход (i = 0)

Общая задача: Переместить наибольший элемент в конец массива.

Детальный разбор шагов

1. **Сравниваем 5 и 2**:

- **Почему**: 5 больше 2, и они расположены не по порядку.
- **Действие**: Меняем местами.
- **Было**: `[5, 2, -3, -10]`
- **Стало**: `[**2**, **5**, -3, -10]`

1. **Сравниваем 5 и -3**:

- **Почему**: 5 больше -3, и они расположены не по порядку.
- **Действие**: Меняем местами.
- **Было**: `[2, 5, -3, -10]`
- **Стало**: `[2, **-3**, **5**, -10]`

1. **Сравниваем 5 и -10**:

- **Почему**: 5 больше -10, и они расположены не по порядку.
- **Действие**: Меняем местами.
- **Было**: `[2, -3, 5, -10]`
- **Стало**: `[2, -3, **-10**, **5**]`

Второй проход (i = 1)

Общая задача: Среди оставшихся элементов (первые три) переместить наибольший в конец.

Детальный разбор шагов

1. **Сравниваем 2 и -3**:

- **Почему**: 2 больше -3, и они расположены не по порядку.
- **Действие**: Меняем местами.
- **Было**: `[2, -3, -10, 5]`
- **Стало**: `[**-3**, **2**, -10, 5]`

1. **Сравниваем 2 и -10**:

- **Почему**: 2 больше -10, и они расположены не по порядку.
- **Действие**: Меняем местами.
- **Было**: `[-3, 2, -10, 5]`
- **Стало**: `[-3, **-10**, **2**, 5]`

Третий проход (i = 2)

Общая задача: Среди оставшихся элементов (первые два) переместить наибольший в конец.

Детальный разбор шагов

1. **Сравниваем -3 и -10**:

- **Почему**: -3 больше -10, и они расположены не по порядку.
- **Действие**: Меняем местами.
- **Было**: `[-3, -10, 2, 5]`
- **Стало**: `[**-10**, **-3**, 2, 5]`

Итог

Теперь массив полностью отсортирован: `[-10, -3, 2, 5]`.

▶ **Пример 2: [2, 7, 4, 1, 5]**

▶ **Пример 3: [6, 5, 4, 3, 2, 1]**

Реализация пузырьковой сортировки на языке Java

```
class ArrayBubble {

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

```
        int[] intArray = {-15, 100, 269, -56, 5}; // массив
        int n = intArray.length; // размер массива = 5

        // Первый цикл (внешний): проходим по всем элементам массива
        for (int i = 0; i < n - 1; i++) {
            // Второй цикл (внутренний): сравниваем и меняем местами пары э
            for (int j = 0; j < n - 1 - i; j++) {
                // Сравниваем текущий и следующий элементы
                if (arr[j] > arr[j + 1]) {
                    // Если текущий элемент больше следующего, меняем их ме
                    int temp = arr[j];      // Временная переменная для хра
                    arr[j] = arr[j + 1];    // Присваиваем текущему элемент
                    arr[j + 1] = temp;      // Присваиваем следующему элеме
                }
            }
        }
    }
}
```

Заключение

Алгоритм пузырьковой сортировки является одним из самых медленных. Если массив состоит из N элементов, то на первом проходе будет выполнено N-1 сравнений, на втором N-2, далее N-3 и т.д.

Таким образом, при сортировке алгоритм выполняет около 0.5x(N^2) сравнений.

- Для N = 5, количество сравнений будет примерно 10
- для N = 10 количество сравнений вырастит до 45.

Таким образом, с увеличением количества элементов сложность сортировки значительно увеличивается

Бинарный поиск

Бинарный поиск — это эффективный алгоритм поиска, который работает с **отсортированными** массивами. Вместо того, чтобы просматривать каждый элемент массива по очереди, бинарный поиск сравнивает искомый элемент с элементом в середине массива, а затем сужает область поиска в два раза. Это позволяет находить элементы гораздо быстрее, чем при использовании простого перебора.

Бинарный поиск — это алгоритм поиска элемента в отсортированном массиве. Основная идея заключается в том, что на каждом шаге область поиска сокращается **вдвое**.

Шаги алгоритма

1. Инициализация: Задаем два указателя — `left и right`, которые изначально указывают на первый и последний элементы массива.
2. Цикл поиска: Пока `left <= right`:
    1. Середина массива: Вычисляем индекс середины массива как `mid = left + (right - left) / 2`. Это помогает избежать переполнения для больших массивов.
    2. Сравнение: Сравниваем элемент в середине массива (`arr[mid]`) с искомым значением (`target`).
        - Элемент найден: Если `arr[mid]` равно `target`, поиск завершен. Возвращаем индекс `mid`.
        - Элемент больше: Если `arr[mid]` больше `target`, сужаем область поиска, присваивая `right = mid - 1`.
        - Элемент меньше: Если `arr[mid]` меньше `target`, сужаем область поиска, присваивая `left = mid + 1`.
3. Завершение: Если `left > right`, элемент не найден. Возвращаем `-1` или соответствующее значение.

▶ **Почему не просто (left + right) / 2?**

Пример

Пусть у нас есть массив arr = [1, 3, 5, 7, 9, 11], и мы ищем `target = 5`.

1. `left = 0`, `right = 5` (5 = `размер массива - 1`)
2. `mid = 0 + (5 - 0) / 2` = 2
3. `arr[mid] = 5`
4. arr[mid] равно target, поэтому возвращаем `mid = 2`.

Примечание: Массив должен быть отсортирован для корректной работы алгоритма.

Пример

`[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]`

Поиск **1**

Итерация	`left`	`right`	Участок массива	`mid`	`array[mid]`	Комментарий
1	0	9	[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]	4	9	9 > 1, уменьшаем `right` до 3
2	0	3	[1, 3, 5, 7]	1	3	3 > 1, уменьшаем `right` до 0
3	0	0	[1]	0	1	1 == 1, нашли элемент. Останавливаем

Поиск **17**

Итерация	`left`	`right`	Участок массива	`mid`	`array[mid]`	Комментарий
1	0	9	[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]	4	9	9 < 17, увеличиваем `left` до 5

Итерация	left	right	Участок массива	mid	array[mid]	Комментарий
2	5	9	[11, 13, 15, 17, 19]	7	15	15 < 17, увеличиваем left до 8
3	8	9	[17, 19]	8	17	17 == 17, нашли элемент. Останавливаем

Поиск **7**

Итерация	left	right	Участок массива	mid	array[mid]	Комментарий
1	0	9	[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]	4	9	9 > 7, уменьшаем right до 3
2	0	3	[1, 3, 5, 7]	1	3	3 < 7, увеличиваем left до 2
3	2	3	[5, 7]	2	5	5 < 7, увеличиваем left до 3
4	3	3	[7]	3	7	7 == 7, нашли элемент. Останавливаем

Поиск **13**

Итерация	left	right	Участок массива	mid	array[mid]	Комментарий
1	0	9	[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]	4	9	9 < 13, увеличиваем left до 5
2	5	9	[11, 13, 15, 17, 19]	7	15	15 > 13, уменьшаем right до 6
3	5	6	[11, 13]	5	11	11 < 13, увеличиваем left до 6
4	6	6	[13]	6	13	13 == 13, нашли элемент. Останавливаем

Задачи для закрепления

- Найти минимальный элемент в отсортированном и повернутом массиве.
- Найти количество вхождений заданного числа в отсортированном массиве.
- Напишите бинарный поиск для нахождения квадратного корня числа с точностью до 3-го знака.
- Найти "первое" вхождение заданного числа в отсортированном массиве.
- Найти "последнее" вхождение заданного числа в отсортированном массиве.

Homework

▶ **English**

▼ **На русском**

Задача 1:

Условие: Напишите программу на Java, которая использует бинарный поиск для нахождения индекса первого вхождения числа в отсортированном массиве с повторяющимися элементами.

Пример:

Вход: Отсортированный массив [1, 3, 3, 3, 7, 9, 11, 13, 15, 17, 17, 19] и число 3. Выход: Индекс первого вхождения числа 3 в массиве.

Задача 2:

Напишите программу, которая находит n-ное вхождение заданного числа в отсортированном массиве. Если число не найдено, программа должна вернуть -1.

Пример:

Вход: Массив [1, 3, 3, 3, 7, 9, 11, 13, 15, 17, 17, 19] и число 3 с номером вхождения 2.

Выход: Индекс 2, поскольку второе вхождение числа 3 находится на этой позиции.

Задача* 3:

Напишите программу, которая сначала сортирует массив целых чисел по возрастанию, а затем по убыванию, используя сортировку пузырьком.

Пример:

Вход: Массив [10, 3, 15, 7, 8, 5, 11, 2].

Выход: Два отсортированных массива: [2, 3, 5, 7, 8, 10, 11, 15] и [15, 11, 10, 8, 7, 5, 3, 2].

Массивы

Задача 4 "Слова в обратном порядке"

Пользователь вводит нескольких слов. Сохраните каждое слово в массиве и выведите все слова в обратном порядке.

Задача 5

С помощью вложенного цикла For (цикл в цикле) написать метод, выводящий на экран треугольник такого вида:

```
1
12
123
1234
12345
123456
```

Задача 6 *

Цель задачи: Найти и вывести имя друга, который съел больше всех кусков пиццы.

У вас есть массив из 6 элементов, каждый из которых представляет количество кусков пиццы, съеденных 6 друзьями. Выведите имя друга, который съел больше всех.

- Создайте массив, который будет содержать количество кусков пиццы, съеденных каждым из 6 друзей. пример: `{2, 4, 3, 5, 1, 3}`
- Создайте массива с именами друзей: пример: `{"Алекс", "Борис", "Вера", "Галя", "Дима", "Елена"}`
- Найдите максимально значение в первом массиве и запомните индекс этого значения. *В нашем примере максимально е значение 5, под индексом 3*
- Найдите друга во втором массиве, который съел больше всех кусков пиццы. *В нашем примере это - Галя*

Code

code/HwSolution_12/src/Task1.java

```
import java.util.Scanner;

/**
 * @author Andrej Reutow
 * created on 20.09.2023
 */
//Создайте массив из 5 элементов для хранения оценок студента.
// Заполните массив оценками и вычислите средний балл студента.
//- используйте Scanner для заполнения массива.
public class Task1 {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Какое количество оценок вы желаете ввести?");
        int length = scanner.nextInt();

        int[] grades = new int[length];

        for (int i = 0; i < length; i++) {
            System.out.println("Введите оценку " + (i + 1));
            int grade = scanner.nextInt();
            grades[i] = grade;
        }

        int sum = calculateSum(grades);
```

```
        double average = sum / length;

        System.out.println("Средняя оценка: " + average);
    }

    public static int calculateSum(int[] source) {
        int result = 0;
        for (int i = 0; i < source.length; i++) {
            result += source[i];
        }
        return result;
    }

}
```

code/HwSolution_12/src/Task2.java

```
/**
 * @author Andrej Reutow
 * created on 20.09.2023
 */

// Напишите программу, которая удаляет из массива элемент по условиям:
//- по заданной позиции.
//- по значению
public class Task2 {

    public static void main(String[] args) {
        int[] source = {1, 5, 10, 115, 120};

        System.out.println("Начальное состояние массива");
        printArray(source); // печать массива

        System.out.println("##############################");

        // Удаление элемента по индексу
        int indexToRemove = 1;
        System.out.println("Удаление по индексу: " + indexToRemove);
        removeByIndex(source, indexToRemove); // удаление по индексу
        printArray(source); // печать массива

        System.out.println("##############################");

        // Удаление элемента по значению
```

```
        int valueToRemove = 120;
        System.out.println("Удаление по значению: " + valueToRemove);
        removeByElement(source, valueToRemove); // удаление по значению
        printArray(source); // печать массива
    }

    public static void removeByElement(int[] array, int element) {
        for (int i = 0; i < array.length; i++) {
            if (array[i] == element) {
                array[i] = 0;
                System.out.println("Элемент " + element + " удален");
            }
        }
    }

    public static void removeByIndex(int[] array, int index) {
        if (index >= 0 && index < array.length) {
            array[index] = 0;
        } else {
            System.out.println("Ошибка, индекса" + index + " не существует!
        }
    }

    public static void printArray(int[] array) {
        for (int i = 0; i < array.length; i++) {
            System.out.println("Index:\t" + i + " value: " + array[i]);
        }
    }
}
```

code/HwSolution_12/src/Task3.java

```
/**
 * @author Andrej Reutow
 * created on 20.09.2023
 */

// Программа должна подсчитать, сколько раз каждое число встречается в масс
public class Task3 {

    public static void main(String[] args) {
        int[] source = {1, 2, 3, 1, 1, 4};
        int length = source.length;
```

```
            for (int outerLoopIndex = 0; outerLoopIndex < length; outerLoopInde
                int currentNumber = source[outerLoopIndex];
                int repeatCounter = 1;

                for (int innerLoopIndex = 0; innerLoopIndex < length; innerLoop
                    if (innerLoopIndex != outerLoopIndex && source[innerLoopInd
                        repeatCounter++;
                    }
                }

                System.out.println("Число " + currentNumber + " повторяется: "
            }
        }


    //    public static void main(String[] args) {
    //        // Создание и заполнение массива случайными числами
    //        int[] array = {1, 2, 3, 1, 1, 4};
    //        int arrayLength = array.length;  // Пример длины массива
    //
    //        // Подсчет частоты каждого числа в массиве
    //        for (int i = 0; i < arrayLength; i++) {
    //            int count = 0;
    //            boolean alreadyCounted = false;
    //
    //            int currentNumber = array[i]; // значение текущего элемента в
    //
    //            for (int j = 0; j < i; j++) {
    //                // currentNumber - значение текущего элемента внешнего ци
    //                // если currentNumber повторяется, значит мы уже посчитал
    //                // записываем в alreadyCounted значение true, что бы не с
    //                /*
    //                Пример:
    //                Есть масив {127, 221, 127, 87}
    //                    Первая итерация внешнего цикла: i = 0; currentNumber =
    //                        заходим с этими значениями в этот цикл.
    //                        этот цикл выполняется пока значение j меньше знач
    //                            Первая итерация: j = 0, i = 0
    //                                при первой итерации в этом цикле ни чего
    //
    //                    Вторая итерация внешнего цикла: i = 1; currentNumber =
    //                        заходим с этими значениями в этот цикл.
    //                        этот цикл выполняется пока значение j меньше знач
```

```
//                          Первая итерация: j = 0, i = 1
//                              цикл запускается, т.к. j < i (0 < 1) - tr
//                              выполням проверку:
//                              if (currentNumber == array[j]) -> (221 ==
//                          Вторая итерация: j = 1, i = 1
//                              выход из цикл, т.к. j < i (1 < 1) - false
//
//                  Третья итерация внешнего цикла: i = 2; currentNumber =
//                      заходим с этими значениями в этот цикл.
//                      этот цикл выполняется пока значение j меньше знач
//                          Первая итерация: j = 0, i = 2
//                              цикл запускается, т.к. j < i (0 < 2) - tr
//                              выполням проверку:
//                              if (currentNumber == array[j]) -> (127 ==
//                                  внутри if:
//                                  устанавливаем значение переменной alr
//                                  Что бы дальше наш код не считал колич
//                                  В нашем случае в самой первой итераци
//                          Вторая итерация: j = 1, i = 2 (следующие итера
//                              j < i (1 < 2) - true (1 меньше 2 ? - да,
//                              выполням проверку:
//                              if (currentNumber == array[j]) -> (127 ==
//                          Третья итерация: j = 2, i = 2 (следубщие итер
//                              выход из цикл, т.к. j < i (2 < 2) - false
//
//                  4 итерация внешнего цикла: i = 3; currentNumber = 87
//                      заходим с этими значениями в этот цикл.
//                      этот цикл выполняется пока значение j меньше знач
//                          Первая итерация: j = 0, i = 3
//                              цикл запускается, т.к. j < i (0 < 3) - tr
//                              выполням проверку:
//                              if (currentNumber == array[j]) -> (87 == a
//                          Вторая итерация: j = 1, i = 3 (следующие итера
//                              j < i (1 < 3) - true (1 меньше 3 ? - да,
//                              выполням проверку:
//                              if (currentNumber == array[j]) -> (87 == a
//                          Третья итерация j = 2, i = 3
//                              j < i (2 < 3) - true (2 меньше 3 ? - да,
//                              if (currentNumber == array[j]) -> (87 == 1
//                          4 итерация j = 3, i = 3
//                              j < i (3 < 3) - true (3 меньше 3 ? - нет,
//
//              Тут не описан дальшейшее выполнение кода после этого цикла!!!
```

```
//                 */
//                if (currentNumber == array[j]) {
//                    alreadyCounted = true;
//                }
//            }
//
//            if (!alreadyCounted) {
//                for (int j = 0; j < arrayLength; j++) {
//                    if (currentNumber == array[j]) {
//                        count++;
//                    }
//                }
//                // Вывод результата
//                System.out.println("Число " + currentNumber + " встречает
//            }
//        }
//    }
}
```

code/HwSolution_12/src/Task4.java

```
import java.util.Arrays;
import java.util.Random;
import java.util.Scanner;

/**
 * @author Andrej Reutow
 * created on 20.09.2023
 */

//Пользователь вводит два числа: длину массива и число X.
//Программа заполняет массив случайными числами (число X может не оказаться
//Найти, сколько раз число X встречается в массиве.
public class Task4 {


    public static void main(String[] args) {
        // Пользователь вводит два числа: длину массива и число X.
        Scanner scanner = new Scanner(System.in); // ручка

        System.out.println("Введите размер массива? 0 - n");
        int arrayLength = scanner.nextInt(); // arrayLength- листок, "=" ко
```

```
        //Программа заполняет массив случайными числами (число X может не о
        int[] source = new int[arrayLength];

        fillArray(source); // заполнение массива рандомными числами

        System.out.println(Arrays.toString(source));

        System.out.println("Какое число вы хотите найти и узнать частоту ег
        int x = scanner.nextInt(); // arrayLength- листок, "=" команда к на

        int counter = 0;
        for (int i = 0; i < source.length; i++) {
            int currentValue = source[i];
            if (currentValue == x) {
                counter++;
            }
        }

        System.out.println("Число " + x + " встречается " + counter + " раз
    }

    public static void fillArray(int[] source) {
        Random random = new Random();
        for (int index = 0; index < source.length; index++) { // 0 ... длин
            int randomValue = random.nextInt(100);// 0 ... 100
            source[index] = randomValue;
        }
    }

}
```

code/HwSolution_12/src/Task5.java

```
import java.util.Arrays;

/**
 * @author Andrej Reutow
 * created on 20.09.2023
 */

// Программа должна проверять, все ли элементы в массиве уникальны.
public class Task5 {
```

```
        public static void main(String[] args) {
            int[] source = {1, 2, 3, 4, 5, 6, 1000, 100, 60};

            boolean isUniq = isUnique(source);
            System.out.println("Массив: " + Arrays.toString(source) + (isUniq ?
        }

        public static boolean isUnique(int[] arr) {
            boolean result = true;
            for (int i = 0; i < arr.length; i++) {
                int currentValue = arr[i];

                for (int j = i + 1; j < arr.length; j++) {
                    int currentInnerLoopValue = arr[j];
                    if (currentValue == arr[j]) {
                        result = false;
                    }
                }

            }

            return result;
        }
    }
```

code/HwSolution_12/src/ArrayTask3.java

```
    /**
     * @author Andrej Reutow
     * created on 20.09.2023
     */

    //Задача 3 Частотный анализ
    //Программа должна подсчитать, сколько раз каждое число встречается в масси
    public class ArrayTask3 {

        // DATA: {1, 5, 1, 3, 5, 98}
        // Output
        // 1 - 2
        // 5 - 2
        // 1 - 2
        // 3 - 1
        // 5 - 2
        // 98 - 1
```

```
    public static void main(String[] args) {
        int[] source = {-100, 1, 5, 1, 3, 5, 98, 1, 100, -100}; // длина ма
        int arrLength = source.length; // 6

//        int lastElementInArray = source[arrLength - 1];

        for (int i = 0; i < arrLength; i++) {
            int currentValue = source[i]; // i = 0, currentValue = 1 | i =
            int counter = 0;

            for (int j = 0; j < arrLength; j++) {
                if (currentValue == source[j]) {
                    counter++;
//                    counter = counter + 1;
                }
            }


            System.out.println("Число " + currentValue + " ,повторяется " +
        }
    }
}
```

code/Lesson_13/src/BinarySearch.java

```
import java.util.Scanner;

/**
 * @author Andrej Reutow
 * created on 21.09.2023
 */
public class BinarySearch {

    public static void main(String[] args) {
        int[] source = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19}; // размер 10

        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("Введите число для поиска");
            int target = scanner.nextInt(); // искомое значение

            // значение искомого элементиа по умолчанию
```

```
            int searchElementIndex = -1; // Если элемент не найдет, вернетс
            int left = 0; // индекс первого элемента в массиве
            int right = source.length - 1; // индекс последнего элемента в м

            // int mid = (right + left) / 2; // (0 + 9) / 2 = 4
            while (left <= right) {
                int mid = left + (right - left) / 2; // // Вычисляем середи
                int currentValue = source[mid]; // текущее значение
                if (currentValue == target) {
                    searchElementIndex = mid;
                } else if (currentValue < target) { // 9 < 1 - fasle
                    left = mid + 1; // Сужаем область поиска справа
                } else if (currentValue > target) { // 9 > 1 - true
                    right = mid - 1; // Сужаем область поиска слева
                }
            }

            // пример для числа 1 и массива [1, 3, 5, 7, 9, 11, 13, 15, 17,
            // 1 итерация - left = 0, right 9, mid = 4, currentValue = 9 диа
            // currentValue == target (9 == 1) - false
            // currentValue < target (9 < 1) - false
            // currentValue < target (9 > 1) - true
            // смещаемся влево, обрезаем правую границу right = mid - 1. (4
            // новый диапазон поиска 1, 3, 5, 7  |||  эту часть обрезали сме

            // 2 итерация - left = 0, right 3, mid = 2, currentValue = 3 диа
            // currentValue == target (9 == 1) - false
            // currentValue < target (9 < 1) - false
            // currentValue < target (9 > 1) - true
            // смещаемся влево, обрезаем правую границу right = mid - 1. (2
            // новый диапазон поиска 1  |||  эту часть обрезали сместив пра


            System.out.println("Индекс искомого элемента " + target + " " +
        }

    }
}
```

code/Lesson_13/src/BubbleSort.java

```
/**
 * @author Andrej Reutow
 * created on 21.09.2023
```

```
 */
public class BubbleSort {

    public static void main(String[] args) {
        int[] source = new int[]{5, -10, 9, 99, 189, -186};

        sort(source);

        for (int i = 0; i < source.length; i++) {
            System.out.println("Index: " + i + " value " + source[i]);
        }
    }
//6 30
//5 25
//5 15



    public static void sort(int[] array) {

        int innerCounter = 0;
        int outerCounter = 0;

        for (int indexOut = 0; indexOut < array.length - 1; indexOut++) { /
            // if (arr[0] > arr[1]) { int temp = arr[0]; arr[0] = arr[1]; a
            for (int indexIn = 0; indexIn < array.length - 1 - indexOut; in
                if (array[indexIn] > array[indexIn + 1]) {
                    int temp = array[indexIn];                  // сохраняю
                    array[indexIn] = array[indexIn + 1];        // перемещаı
                    array[indexIn + 1] = temp;
                }
                innerCounter++;
            }
            outerCounter++;
        }

        System.out.println("outerCounter " + outerCounter);
        System.out.println("innerCounter " + innerCounter);
    }
}
```