Сортировки

Составитель: Рощупкин Александр

Алгоритмы сортировки

* **Алгоритм сортировки** — это алгоритм для упорядочивания элементов в списке. В случае, когда элемент списка имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки.

* **Пузырьковая**
* **Вставкой**
* **Выбором**
* **Подсчётом**

* **Устойчивость** — устойчивая сортировка не меняет взаимного расположения элементов с одинаковыми ключами

Стандартные алгоритмы

* Алгоритмы работы с массивами собраны в классе **Arrays**
* Алгоритмы работы с коллекциями (ArrayList и т.д.) собраны в классе **Collections**

Пузырьковая сортировка

* Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N-1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции, как пузырёк в воде — отсюда и название алгоритма)

Порядок сортировки

* Для изменения порядка сортировки достаточно поменять условие сравнения, условие сравнения позволяет определить любой порядок сортировки
* Если нужно отсортировать объекты достаточно в условии сравнения использовать нужные для сравнения поля

Объекты оболочки

* Каждый примитивный тип имеет соответствующий класс оболочку, предназначенный для хранения примитивного типа
* Integer, Long, Character, Byte, Double, Float, Boolean
* За счёт возможности автобоксинга можно работать с объектами оболочками как с примитивными типами

Стандартная сортировка

* Алгоритм стандартной сортировки находится в методе **sort**, классов **Arrays** и **Collections** соответственно
* Порядок сортировки можно задавать с помощью интерфейса Comparator
* Нужно реализовать метод **int compare(Object, Object)**
* Метод возвращает число в качестве результата сравнения: положительное (первый объект больше второго), отрицательное (второй объект больше первого), ноль (объекты равны)
* **sort(vector, comparator)**

Пример использования компаратора

```
Comparator intComp = new Comparator<>() {
    @Override
    public int compare(Object o1, Object o2) {
        Integer i1 = (Integer) o1;
        Integer i2 = (Integer) o2;
        if (i1 > i2) {
            return 1;
        } else if (i1 < i2) {
            return -1;
        } else {
            return 0;
        }
    }
};
int[] vector = {1,3,2,4};
Arrays.sort(vector, intComp);
```

```
Comparator<String> strComp = new
Comparator<>() {
    @Override
    public int compare(String o1, String o2) {
        return o1.compareTo(o2);
    }
};

List<String> strs = Arrays.asList("2", "1","3");
Collections.sort(strs, strComp);
```

Задание

* Модифицировать метод сортировки пузырьком для задания порядка сортировки через не параметризированный компаратор, передаваемый в качестве аргумента

Сортировка вставкой

* Элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов

Сортировка выбором

* Идея метода состоит в том, чтобы создавать отсортированную последовательность путем присоединения к ней одного элемента за другим в правильном порядке
* На каждом i-м шаге выбираем наименьший элемент и меняем его местами с текущим

Частичное упорядочивание

* Идея заключается в выборе в массиве опорного элемента (центрального) и на основании его значения разместить все меньшие и равные ему элементы слева от него, а все большие справа от него
* Самое главное – выполнить такое упорядочивание в один проход по массиву

Слияние массивов

* На каждом шаге мы берём меньший из двух первых элементов подмассивов и записываем его в результирующий массив. Счётчики номеров элементов результирующего массива и подмассива, из которого был взят элемент, увеличиваем на 1
* Когда один из подмассивов закончился, мы добавляем все оставшиеся элементы второго подмассива в результирующий массив

Домашнее задание

* Разобраться с алгоритмом сортировки подсчётом
* Написать метод, который выполняет частичное упорядочивание массива
* Написать метод, выполняющий слияние двух массивов