Algorithms With Java

Week 0 Day 0: Bubble Sort

Академия Ковалевского

Содержание

1. Теория

Сортировка пузырьком — Bubble Sort

В данном курсе мы реализуем основные виды сортировок - от более простой к более сложной. Мы начнем с наиболее простого вида сортировки: Bubble Sort.

Сортировки отличаются по скорости работы и размером потребляемой памяти.

Несколько неплохих материалов по теме сортировки пузырьком (**!** лучше искать материал самостоятельно): **видео** (и **еще** одно) с разбором сортировки.

Сложность алгоритма (Big O Notation)

Для понимания того, насколько быстро работает сортировка (и в целом чем отличается одна сортировка от другой), нужно изучить понятия большой О нотации. Некоторые ссылки, которые пригодятся:

- **Статья** на Хабре
- Неплохое **видео** по теме

Для выполнения данного дня важно уметь оценивать лучшую сложность, худшую и среднюю сложность алгоритма.

Сопроводительные темы

Несколько тем, которые необходимы для прохождения этого урока (их вы уже знаете или изучите прямо сейчас):

- **Обобщения** (generic)
- **Lambda** функции в Java
- **Default** методы в Java

2. Практическая работа

Сортировка

В данном задании нужно реализовать **интерфейс Sort** с несколькими **default** методами:

```
package academy.kovalevskyi.algorithms.week0.day0;

import java.util.Comparator;
import java.util.stream.IntStream;

public interface Sort {
 default <T extends Comparable<? super T>> void sort(final T[] target) {
   sort(target, T::compareTo);
 }
 default <T> void sort(final T[] target, final Comparator<T> comparator) {
  // TODO
 }
 default <T> T[] createSortedArray(final T[] target, final Comparator<T>
comparator) {
  // TODO
 }
 default String complexityBest() { // TODO }
 default String complexityAverage() { // TODO}
 default String complexityWorst() { // TODO}
 default String spaceComplexityWorst() { // TODO }
}
```

Такая реализация позволит:

- Иметь один метод **<T extends Comparable<? super T>> void sort(final T[] target)**, для всех потомков (мы его нигде больше переопределять не будем). *Внимательно изучите данную реализацию метода **void sort(final T[] target)**, не продолжайте выполнение задания, если не понимаете как он реализован (если нужно, спросите коллег или освежите тему **lamda в Java**).*
- Задает стандартную реализацию сортировки методом пузырька

Задачи

1. Сначала реализуйте метод **`sort(final T[] target, final Comparator<T> comparator),`** который выполнен пузырьком.
2. После этого реализуйте метод **`createSortedArray(final T[] target, final Comparator<T> comparator`**), его задача создать копию массива, выполнить сортировку на копии массива и вернуть копию массива.
3. Оставшиеся методы это методы, которые должны вернуть ответ (в виде строки) на следующие вопросы:
    - Какова сложность алгоритма в самом лучшем случае
    - Какова сложность алгоритма в среднем
    - Какова сложность алгоритма в худшем случае
    - Какова сложность алгоритма с точки зрения занимаемого места в памяти

Значения, которые можно возвращать (ответ должен быть в виде одного из этих значений):

- ☐ N
- ☐ N^2
- ☐ N^3
- ☐ N^4
- ☐ log(N)
- ☐ N*log(N)
- ☐ N*log^2(N)
- ☐ N^2*log(N)
- ☐ N*log(N^2)
- ☐ const

Дополнительные задачи

В данном дне нужно решить две дополнительные задачи в следующем классе:

```
package academy.kovalevskyi.algorithms.week0.day0;

import java.util.Arrays;

public class Task {

  public static boolean sameCharactersSorting(String left, String right) {
    // TODO
  }

  public static boolean sameCharactersO1(String left, String right) {
    // TODO
  }
}
```

Оба метода решают одну и ту же проблему, они должны вернуть true если вторую строку можно написать путем перестановок символов в первой строке. То есть:

- для строк: “abcd” и “bcda” ответ true, так как переставляя символы первой строки можно получить вторую
- однако для строк “bbca” и “abcd” ответ false.

Разница между методами в том, что в первом методе **sameCharactersSorting**, можно использовать свой метод сортировки (соответственно метод будет иметь сложность равную сложности метода сортировки). А вот во втором варианте (**sameCharactersO1**) нужно написать метод, который имеет сложность **O(const*N)**, const — может быть любой константой.

При приеме решения у других студентов убедитесь, что их решение имеет корректную сложность!

3. Важно

В данном дне можно (но не обязательно) использовать:

- `Arrays.copyOf`
- `IntStream.range`

Все остальное использовать запрещено!

Важно понимать, что Zeus не умеет тестировать корректность сложности программы. Так что очень важно при приеме задачи прочитать код и убедится, что код действительно работает с той сложностью, которая заявляется автором кода. Если код должен работать с **O(N* log(N)),** однако в реальности, после чтения кода, есть подозрение, что он имеет сложность **$O(N^2)$ — не принимайте задачу!**