Санкт-Петербургский Национальный Исследовательский Университет Информационных технологий, механики и оптики

Факультет инфокоммуникационных технологий

Лабораторная работа №3

Выполнили: Крем польская Е.А

Петрова Н.Г

Проверил: Мусаев А.А.

Санкт-Петербург 2022

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc118387787)

[1 ЗАДАНИЕ 1 4](#_Toc118387788)

[1.1 Сортировка 4](#_Toc118387789)

[2 ЗАДАНИЕ 2 5](#_Toc118387790)

[2.1 Реализация алгоритмов различной сложности 5](#_Toc118387791)

[3 ЗАДАНИЕ 3 6](#_Toc118387792)

[3.1 Построение зависимости алгоритмов различной сложности 6](#_Toc118387793)

[ПРИЛОЖЕНИЕ 8](#_Toc118387794)

# ВВЕДЕНИЕ

В этой работе мы отсортируем числа в массиве, пользуясь методом пузырьковой сортировки, а также используя встроенную функцию sort, напишем алгоритмы, которые имеют разную асимптотическую сложность и построим график зависимости количества шагов для алгоритмов разной сложности от количества элементов, которые были задействованы.

Задания, которые необходимо выполнить:

1. Задание 1: написать программу для пузырьковой сортировки, оценить сложность данного метода, сравнить с методом sort().
2. Задание 2: придумать и реализовать алгоритмы, имеющие сложность O(3n), O(nlogn), O(n!), O(n3), O(3log(n));
3. Задание 3: построить зависимость между количеством элементом и количеством шагов для алгоритмов со сложностью О(1), O(logn), O(n^2), O(2^n), сравнить сложность данных алгоритмов.

# ЗАДАНИЕ 1

# Сортировка

Была написана программа пузырьковой сортировки, в которой на каждом шаге находится наибольший элемент из двух соседних. Этот элемент ставится в конец пары. Таким образом, происходит эта сортировка (рисунок 1).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Ввод/вывод программы с пузырьковой сортировкой

Данный алгоритм пузырьковой сортировки имеет сложность O(n^2). В то время, как встроенный в Python метод sort() предназначен для более широкого круга решения задач и имеет сложность O(nlogn). То есть написанная программа имеет большую сложность.

**Вывод:** Сортировка методом пузырька – это очень легкий в реализации алгоритм сортировки, но при этом не самый эффективный, за счет двойного цикла в нём. Написали программу для пузырьковой сортировки и оценили сложность данного метода, сравнили с методом sort().

# ЗАДАНИЕ 2

# Реализация алгоритмов различной сложности

1. Алгоритм сложности O(3n) можно построить при помощи использования цикла for. Так как в нотации O отбрасываются все константы перед n, то есть O(3n)~ O(n).
2. Алгоритм сложности O(nlogn) можно получить при применении сортировки слиянием (числа разбиваются на 2 части, и каждая часть сортируется отдельно, после чего эти части объединяются).
3. Алгоритм сложности O(n!) можно получить, например, если составлять всевозможные уникальные комбинации из заданных элементов.
4. Алгоритм сложности O(n^3) можно построить при помощи трех вложенных друг в друга циклов for.
5. Алгоритм сложности O(3log(n)) можно получить при использовании бинарного поиска. Сложность алгоритма растёт с увеличением входных данных. Это такой алгоритм, где для каждой итерации берется половина элементов.

**Вывод:** Придумали и реализовали алгоритмы, имеющие сложность O(3n), O(nlogn), O(n!), O(n3), O(3log(n)).

# ЗАДАНИЕ 3

# Построение зависимости алгоритмов различной сложности

Построим зависимость между количеством элементом и количеством шагов для алгоритмов с различной сложностью в виде таблице и графиков. Сравнив все алгоритмы, можно заметить, что самый сложным алгоритмом является O(2^n), далее O(n^2), затем O(logn) и потом O(1).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество шагов | O(1) | O(logn) | O(n^2) | O(2^n) |
| 1 | 1 | 0 | 1 | 2 |
| 2 | 1 | 1 | 4 | 4 |
| 3 | 1 | 1 | 9 | 8 |
| 4 | 1 | 2 | 16 | 16 |
| 5 | 1 | 2 | 25 | 32 |
| 6 | 1 | 2 | 36 | 64 |
| 7 | 1 | 2 | 49 | 128 |
| 8 | 1 | 3 | 64 | 256 |
| 9 | 1 | 3 | 81 | 512 |
| 10 | 1 | 3 | 100 | 1024 |
| 11 | 1 | 3 | 121 | 2048 |
| 12 | 1 | 3 | 144 | 4096 |
| 13 | 1 | 3 | 169 | 8192 |
| 14 | 1 | 3 | 196 | 16384 |
| 15 | 1 | 3 | 225 | 32768 |
| 16 | 1 | 4 | 256 | 65536 |

**Вывод:** Построили зависимость между количеством элементом и количеством шагов для алгоритмов со сложностью О(1), O(logn), O(n^2), O(2^n), сравнили сложность данных алгоритмов.

# ПРИЛОЖЕНИЕ

Ссылка на гитхаб: [Algoritms/Лаба 3 (сортировка, сложность алгоритмов) at main · NatalyaPetrova/Algoritms (github.com)](https://github.com/NatalyaPetrova/Algoritms/tree/main/%D0%9B%D0%B0%D0%B1%D0%B0%203%20(%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0%2C%20%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%BE%D0%B2))