Санкт-Петербургский Государственный Технический Университет (Технологический институт)

Кафедра системного анализа и информационных технологий

**Лабораторная работа №3**

Выполнили:

Силуянов Д. К., Степанов Д. А.

Проверил:

Мусаев А. А.

Санкт Петербург

2022

**Цель:** изучить алгоритмы

**Задачи:**

1. Написать программу для пузырьковой сортировки. Оценить сложность данного метода. Сравнить с методом sort().

2. Придумать и реализовать алгоритмы, имеющие сложность O(3n), O(nlog(n)), O(n!), O( ), O(3log(n))

3. Построить зависимость между количеством элементом и количеством шагов для алгоритмов со сложностью О(1), O(logn), O(), O(). Сравнить сложность данных алгоритмов.

**Инструменты:** PyCharm

**Ход работы:**

1. Мы написали программу, которая сортирует числа в массиве по возрастанию методом «пузырька». Код программы представлен на рисунке 1:

Изображение выглядит как текст

Автоматически созданное описание

рис. 1

В Python-е существует метод sort(), который также может выполнить сортировку числовых данных. Ниже (рисунок 2) приведён код, сортирующий числа массива по возрастанию с использованием метода sort():

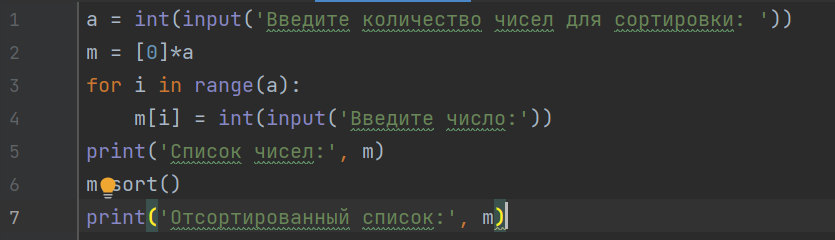


рис. 2

Итак, имея две программы, выполняющие, по сути своей, одну и ту же функцию, но разными способами, мы можем сравнить их. Программа, выполняющая сортировку «пузырьком» занимает 12 строк, в то время как на метод sort() уходит всего 7 строк. Сложность алгоритма сортировки методом «пузырька» можно оценить с помощью концепции Big O. Данный метод сортировки содержит в себе три цикла for, один из которых вложен в другой. Итого мы имеем сложность О(n + ). Однако n можно пренебречь, потому что это не что иное, как, так называемая, «ненужная сложность». Соответственно, окончательно сложность нашей программы можно охарактеризовать как О().

1. Мы написали программы, сложности которых:

А. О(3n) (рисунок 3);

Изображение выглядит как текст

Автоматически созданное описание

рис. 3

Данная программа создает n количество элементов для каждого из трех массивов.

Б. O(nlog(n)) (рисунок 4);

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

рис. 4, рис. 5

Merge sort (сортировка слиянием)

В. O(n!) (рисунок 6);

Изображение выглядит как текст

Автоматически созданное описание

рис. 6

Программа вычисляет факториал для каждого числа в заданном массиве.

Г. O( ) (рисунок 7);

Изображение выглядит как текст

Автоматически созданное описание

рис. 7

Эта программа проверяет цифру на четность.

Д. O(3log(n)) (рисунок 8);

Изображение выглядит как текст, экран, снимок экрана

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст, экран, снимок экрана

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст, экран, снимок экрана

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

Программа трижды выполняет бинарный поиск

рис. 8

1. А.

Можно заметить, что в сложности O(1) количество шагов не зависит от количества переменных, это означает, время выполнения программы с увеличением количества переменных никак увеличиваться не будет.

Б.

Здесь ускорение функции отрицательное, следовательно данная сложность выполнения алгоритма самая оптимизированная, особенно, когда речь идет о большом количестве n.

В.

Количество шагов в данной функции увеличивается полиномиально.

Г.

Количество шагов в этой функции увеличивается экспоненциально.

Отталкиваясь от оценки всех вышеперечисленных сложностей, можно сказать, что наихудшей сложностью из приведенных является экспоненциальная, лучшей – прямо пропорциональная.

**Выводы:**

Мы написали программу для пузырьковой сортировки, оценили ее сложность, сравнили данный метод сортировки с методом sort().

Реализовали алгоритмы имеющие сложность: O(3n), O(nlog(n)), O(n!), O( ), O(3log(n)).

Построили зависимость между количеством элементов и количеством шагов для алгоритмов, имеющих сложность О(1), O(log(n)), O(n^2), O(2^n) и сравнили сложности этих алгоритмов.