Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий

**Лабораторная работа №3**

Выполнил:

Шишминцев Д. В.

Миляев Д. Д.

Проверил

Мусаев А.А.

Санкт-Петербург,

2022

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc118503816)

[ЗАДАНИЕ 1 4](#_Toc118503817)

[ЗАДАНИЕ 2 5](#_Toc118503818)

[ЗАДАНИЕ 3 6](#_Toc118503819)

[ВЫВОД 8](#_Toc118503820)

[СПИСОК ЛИТЕРАТУРЫ 9](#_Toc118503821)

[ПРИЛОЖЕНИЕ 10](#_Toc118503822)

# ВВЕДЕНИЕ

Цель данной работы – знакомство с нотацией «О большое», оценкой сложности алгоритмов, а также сравнения сложностей различных алгоритмов.

# ЗАДАНИЕ 1

Был разработан алгоритм пузырьковой сортировки. Алгоритм имеет сложность O(n2). В качестве метода .sort() в языке программирования Python используется сортировка методом Timsort. Данный алгоритм является комбинацией алгоритмов сортировки слиянием и вставками. Данный алгоритм имеет сложность O(n log n). Алгоритм сортировки пузырьком сложнее, чем метод .sort()

# ЗАДАНИЕ 2

Были разработаны различные алгоритмы имеющие сложности O(3n), O(nlogn), O(n!), O(n3), O(3log(n)).

Для реализации алгоритма сложностью O(3n) был написан следующий код на Python. (Фрагмент кода 1)

for i in range(len(s)):

a = s[i]

for j in range(3):

s[i] \*= a

Фрагмент кода 1

В качестве алгоритма сложностью O(n log n) был разработан алгоритм сортировки. Для алгоритма сложностью O(n!) был реализован алгоритм вычисления факториала. В качестве примера алгоритма сложностью O(n3) был написан простейший алгоритм сортировки. Все разработанные программы добавлены в репозиторий на GitHub (Приложение 1).

# ЗАДАНИЕ 3

Была проанализирована зависимость между количеством элементов и количеством шагов для алгоритмов. Для каждой зависимости построены графики. (Рисунок 1, 2, 3, 4)

Chart

Description automatically generated

Рисунок 1

Chart, line chart

Description automatically generated

Рисунок 2

Chart, line chart

Description automatically generated

Рисунок 3

Chart, scatter chart

Description automatically generated

Рисунок 4

Для сравнения сложности алгоритмов было проведено сравнение производных.

Чем больше производная, тем быстрее растет количество шагов при увеличении количества элементов. Следовательно, можно расставить алгоритмы от самого легкого до самого сложно следующим образом:

* O(1)
* O(3)
* O(n\*log n)
* O(n2)
* O(2n)

# ВЫВОД

Была изучена нотация «О большое», а также в ходе лабораторной работы было разработано несколько алгоритмов, оценена их сложность и сравнена между собой.

# СПИСОК ЛИТЕРАТУРЫ

# Aditya Y. Bhargava, Grokking Algorithms: An illustrated guide for programmers and other curious people. ISBN: 9781617292231

# ПРИЛОЖЕНИЕ

Репозиторий с кодом лабораторной работы на GitHub: <https://github.com/dimilka/alg__lab_3>