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

Факультет инфокоммуникационных технологий и систем связи

Лабораторная работа №3

Вариант №1

Выполнил(и:)

Алексеев Т.Ю.

Оншин Д.Н.

Проверил

Мусаев А.А.

Санкт-Петербург,

2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы являлось знакомство со сложностями алгоритмов и с примерами алгоритмов различной сложности.

Для достижения данной цели необходимо было выполнить следующие задания:

1. Задание 1: создание алгоритма для сортировки пузырьком, оценка его сложности и сравнение его с методом sort().
2. Задание 2: реализация алгоритмов различных сложностей: O(3n), O(nlogn), O(n!), $O(n^3)$, O(3log(n)).
3. Задание 3: построение зависимости между количеством элементов и количеством шагов для алгоритмов сложности O(1), O(logn), $O(n^2)$, $O(2^n)$ и сравнение их сложности.

Решения данных задач будут находиться на GitHub по ссылке: https://github.com/NorthPole0499/Algoritms_task_3

1. ЗАДАНИЕ 1

В данном задании нужно было написать алгоритм сортировки «пузырьком» и сравнить время его работы с временем работы внутреннего метода сортировки sort(). В начале программы пользователь должен ввести список. Для удобства ввод осуществляется в одну строку, то есть сначала вводимые пользователем числа воспринимаются как одна строка. После чего программа разделяет введенные числа (которые для Python на данный момент являются строками) на отдельные строки и добавляет их в список. Затем эти строки преобразуются в числа с помощью отдельной функции.

После создания списка с числами запускается алгоритм сортировки «пузырьком»: его принцип работы основан на перемещении наименьших чисел в начало списка путем сравнения этих чисел с их соседями. То есть, если пользователь ввел числа 3, 1, 2, то программа сравнит сначала 3 и 1, так как 3 больше чем 1, программа поменяет эти числа местами, и получится новый список 1, 3, 2. После этого программа сравнит числа 3 и 2 и поменяет их местами, так как 3 больше 2. Получится отсортированный список 1, 2, 3.

Перед началом работы данного алгоритма с помощью библиотеки timeit заведен счетчик времени и, после его завершения на экран выведется время, затраченное на работу. Далее заводится еще один счетчик, который будет считать, сколько времени уйдет на выполнение команды sort(). После на экран выведется затраченное время.

```
Input list: 5 4 3 2 1
[1.0, 2.0, 3.0, 4.0, 5.0]
5.990000000011264e-05
1.4000000003733248e-06
[1.0, 2.0, 3.0, 4.0, 5.0]
```

Приложение 1 – Пример ввода и вывода программы для Задания 1

2. ЗАДАНИЕ 2

В задании 2 нужно реализовать ряд алгоритмов определённой сложности.

Функция dif_3n(n) содержит алгоритм со сложность O(3n). Он содержит в себе два цикла for. Внешний цикл совершает 3 итерации, тогда как вложенный цикл – n итераций. В теле цикла происходит добавление переменной-счётчика в список data.

Функция dif_nlog(n) содержит алгоритм со сложностью O(nlogn). В функции представлен цикл for, совершающий n итераций. В теле цикла представлен бинарный поиск среди n элементов. Найденные числа добавляются в список data.

Функция dif_n_fact(n) содержит алгоритм со сложностью O(n!). Он содержит в себе цикл for на n итераций. В теле цикла представлен вызов этой же функции с аргументом n – 1, то есть функция получилась рекурсивной.

Функция dif_n_3(n) содержит алгоритм со сложностью $O(n^3)$. Он содержит в себе три вложенных друг в друга цикла for на n итераций. В самом теле цикла происходит увеличение переменной-счётчика.

Функция dif_3log(n) содержит алгоритм со сложностью O(3log(n)). В функции представлен цикл for, совершающий 3 итерации. В теле цикла представлен бинарный поиск среди n элементов. Найденные числа добавляются в список data.

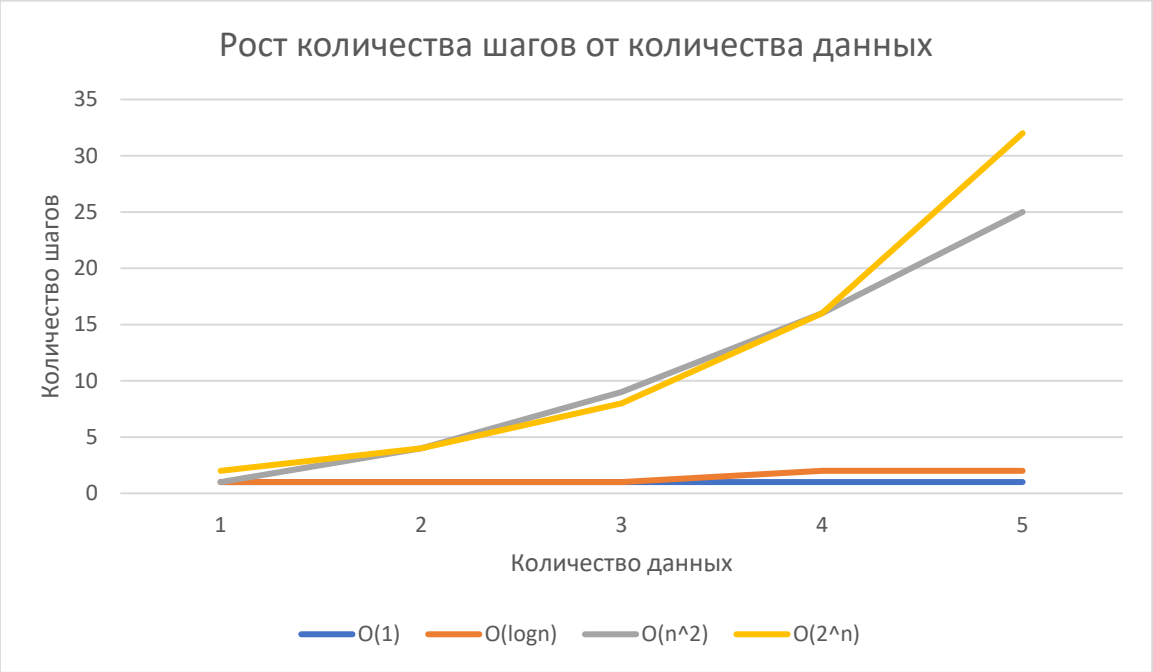
Таким образом, мы написали и описали алгоритмы всех необходимых сложностей.

3. ЗАДАНИЕ 3

В данном задании было необходимо составить зависимость между количеством данных и количеством шагов для алгоритмов различной сложности. Это всё можно представить в виде таблицы и графика.

Таблица 1 – Зависимость количества шагов от количества данных

	O(1)	**O(logn)**	**O(n^2)**	**O(2^n)**
2	1	1	4	4
4	1	2	16	16
8	1	3	64	256
10	1	3	100	1024



Приложение 2 – График роста количества шагов для алгоритмов различной сложности

По графику можно заметить, что количество необходимых шагов растёт значительно быстрее у алгоритмов со сложностью O(2^n). И наоборот, алгоритмом, чьё количество шагов вообще не изменяется, является алгоритм со сложностью O(1).

Таким образом, можно выстроить алгоритмы по мере резкости роста шагов: O(1), O(logn), O(n^2), O(2^n).

ЗАКЛЮЧЕНИЕ

Таким образом, в ходе выполнения данной лабораторной работы были достигнуты все поставленные цели. Произошли знакомство со сложностями алгоритмов и со множеством примеров, иллюстрирующих различные сложности. Также, были проведены сравнения алгоритмов с основными сложностями.

СПИСОК ЛИТЕРАТУРЫ

1) Как создать диаграмму или график в Word 2007// URL: https://support.microsoft.com/ru-ru/office/как-создать-диаграмму-или-график-в-word-2007-58516b99-55fc-4f45-ac81-cc6868a18a8a#:~:text=На%20вкладке%20Вставка%20в%20группе,ячейку%2C%20при%20переходе%20к%20следующей
(дата обращения: 03.11.2022)