Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

Выполнил студент группы КС-36 Гакиев Артур Лечиевич

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Дата сдачи: 21.03.2022

Оглавление

[Описание задачи. 2](#_Toc63548272)

[Описание метода/модели. 2](#_Toc63548273)

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

# Описание задачи.

1. Необходимо реализовать метод быстрой сортировки.
2. Для реализованного метода сортировки необходимо провести серию тестов для всех значений N из списка (1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000), при этом:

* в каждом тесте необходимо по 20 раз генерировать вектор, состоящий из N элементов
* каждый элемент массива заполняется случайным числом с плавающей запятой от -1 до 1

1. На основании статьи реализовать проверки негативных случаев и устроить на них серии тестов аналогичные второму пункту:
2. Отсортированный массив
3. Массив с одинаковыми элементами
4. Массив с максимальным количеством сравнений при выборе среднего элемента в качестве опорного
5. Массив с максимальным количеством сравнений при детерминированном выборе опорного элемента

* При работе сортировки подсчитать количество вызовов рекурсивной функции, и высоту рекурсивного стека. Построить график худшего, лучшего, и среднего случая для каждой серии тестов.

# Описание метода/модели.

Алгоритм быстрой сортировки:

Один из самых быстрых известных универсальных алгоритмов сортировки массивов: в среднем обменов при упорядочении элементов; из-за наличия ряда недостатков на практике обычно используется с некоторыми доработками.

1. Выбрать элемент массива. Он же является опорным.
2. Разбиение: перераспределение элементов в массиве таким образом, что элементы, меньшие опорного, помещаются перед ним, а большие или равные - после.
3. Рекурсивно применить первые два шага к двум подмассивам слева и справа от опорного элемента. Рекурсия не применяется к массиву, в котором только один элемент или отсутствуют элементы.

# Выполнение задачи.

Алгоритм быстрой сортировки реализован на языке Python 3.10. Были проведены тесты(Рисунок 1), вычисляющие скорость работы алгоритма и считающие количество вызовов рекурсий функции (Таблица 1).

Рис. 1. Время работы алгоритма.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Наилучший случай | Средний случай | Наихудший случай | Рекурсии |
| 1000 | 0,000993013 | 0,001484406 | 0,0022614 | 26778 |
| 2000 | 0,002000093 | 0,004066384 | 0,006396532 | 80224 |
| 4000 | 0,005993843 | 0,008242846 | 0,013342381 | 187120 |
| 8000 | 0,01328826 | 0,018629301 | 0,027470827 | 400382 |
| 16000 | 0,027551174 | 0,031752074 | 0,036988258 | 827036 |
| 32000 | 0,057166576 | 0,066479003 | 0,113755226 | 1679774 |
| 64000 | 0,344644785 | 0,364630616 | 0,388974428 | 3386422 |
| 128000 | 0,751099825 | 0,781695664 | 0,838529825 | 6799888 |

Таблица 1. Время работы функции, а также количество рекурсивных вызовов.

Так же были проведены проверки негативных случаев быстрой сортировки. На таблице 2 и рисунке 2 можно наблюдать данные по работе быстрой сортировки на заранее отсортированном массиве при помощи функции питона sorted().

Рис 2. Данные по заранее отсортированному массиву

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Наилучший случай | Средний случай | Наихудший случай | Рекурсии |
| 1000 | 0,000999212 | 0,001573515 | 0,002485991 | 26764 |
| 2000 | 0,002991438 | 0,003356433 | 0,004639864 | 80124 |
| 4000 | 0,005927324 | 0,006831205 | 0,008538485 | 186866 |
| 8000 | 0,012531519 | 0,013853276 | 0,01721096 | 400118 |
| 16000 | 0,026157856 | 0,030572331 | 0,047008276 | 826740 |
| 32000 | 0,054726362 | 0,06358906 | 0,082612276 | 1680320 |
| 64000 | 0,122895718 | 0,137383604 | 0,172170639 | 3386914 |
| 128000 | 0,272862911 | 0,375710559 | 0,838680029 | 6799638 |

Таблица 2. Данные по заранее отсортированному массиву

На рисунке 3 вместе с таблицей 3 представлены данные по массиву, который содержит в себе одинаковые элементы. На этих данных мы можем заметить, что количество рекурсивных вызовов функции всегда изменяется на одно и то же количество рекурсий – 60. Это происходит, из-за того что количество элементов массива возрастает всегда в двое, но сами элементы остаются неизменными.

Рисунок 3. Данные по массиву с одинаковыми элементами

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Наилучший случай | Средний случай | Наихудший случай | Рекурсии |
| 1000 | 0 | 0 | 0 | 60 |
| 2000 | 0 | 5,06E-05 | 0,001012564 | 120 |
| 4000 | 0 | 0,000101852 | 0,001011848 | 180 |
| 8000 | 0 | 0,000260782 | 0,00102663 | 240 |
| 16000 | 0 | 0,000561106 | 0,001097441 | 300 |
| 32000 | 0,000992537 | 0,001704407 | 0,002676487 | 360 |
| 64000 | 0,002048016 | 0,002983499 | 0,005053282 | 420 |
| 128000 | 0,003921509 | 0,004776156 | 0,00699544 | 480 |

Таблица 3. Данные по массиву с одинаковыми элементами.

На рисунке 4 и таблице 4 представлены данные, полученные при выборе среднего элемента в качестве опорного.

Рисунок 4. Данные при выборе среднего значения в качестве опорного.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Наилучший случай | Средний случай | Наихудший случай | Рекурсии |
| 1000 | 0,000999451 | 0,001618695 | 0,00240922 | 26878 |
| 2000 | 0,002945662 | 0,003898728 | 0,005670071 | 80500 |
| 4000 | 0,004999638 | 0,007241881 | 0,010569811 | 187768 |
| 8000 | 0,013058901 | 0,01567049 | 0,021027565 | 402244 |
| 16000 | 0,027245283 | 0,042140591 | 0,078172922 | 831268 |
| 32000 | 0,059465885 | 0,065961409 | 0,075646162 | 1689418 |
| 64000 | 0,126722097 | 0,132418275 | 0,146542788 | 3405076 |
| 128000 | 0,279694319 | 0,293345773 | 0,308341265 | 6838576 |

Таблица 4. Данные при выборе среднего значения в качестве опорного.

# Заключение.

Изначально в моей реализации метода быстрой сортировки в качестве опорного элемента я использую первый элемент массива, из-за этого время выполнения моей сортировки в среднем составляет O(n logn),а в наихудшем имеет время работы O(n2). Но при выборе среднего элемента массива в качестве опорного элемента можно наблюдать ускорение работы сортировки больше, чем в два раза. Это связано с тем, что теперь в любом случае время работы алгоритма становится O(n logn)