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-38..... Прилепский Артем Сергеевич
Ссылка на репозиторий: .....github.com/news1d/Algorithms_and_structures

Приняли: .....Пысин Максим Дмитриевич
.....Краснов Дмитрий Олегович
.....Лобанов Алексей Владимирович
.....Крашенинников Роман Сергеевич

Дата сдачи: 10.03.2023

Оглавление

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

3. На основании статьи реализовать проверки негативных случаев и устроить на них серии тестов аналогичные второму пункту:

    - Отсортированный массив
    - Массив с одинаковыми элементами
    - Массив с максимальным количеством сравнений при выборе среднего элемента в качестве опорного
    - Массив с максимальным количеством сравнений при детерминированном выборе опорного элемента

4. При работе сортировки подсчитать колличество вызовов рекурсивной функции, и высоту рекурсивного стека. Построить график худшего, лучшего, и среднего случая для каждой серии тестов.
5. Для каждой серии тестов построить график худшего случая.
6. Подобрать такую константу c, что бы график функции c * n * log(n) находился близко к графику худшего случая, если возможно построить такой график.
7. Проанализировать полученные графики и определить есть ли на них следы деградации метода относительно своей средней сложности.

Описание метода/модели.

Быстрая сортировка (англ. *quick sort*, сортировка Хоара) — один из самых известных и широко используемых алгоритмов сортировки. Среднее время работы $O(nlogn)$, что является асимптотически оптимальным временем работы для алгоритма, основанного на сравнении. Хотя время работы алгоритма для массива из n элементов в худшем случае может составить $\Theta(n^2)$, на практике этот алгоритм является одним из самых быстрых.

Выполнение задачи.

Для реализации программы был выбран язык программирования Python. Для каждой серии тестов было произведено 20 попыток.

Алгоритм quicksort:

В качестве аргументы функция принимает сортируемый массив элементов.

Алгоритм можно условно разделить на 3 блока:

- Выбирается из массива элемент, называемый опорным. Это может быть любой из элементов массива. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность.
- Сравнивается все остальные элементы с опорным и переставить их в массиве так, чтобы разбить массив на три непрерывных отрезка, следующих друг за другом: «элементы меньшие опорного», «элементы равные опорному» и «элементы большие опорного».
- Для отрезков «меньших» и «больших» значений выполняется рекурсивно та же последовательность операций, если длина отрезка больше единицы.

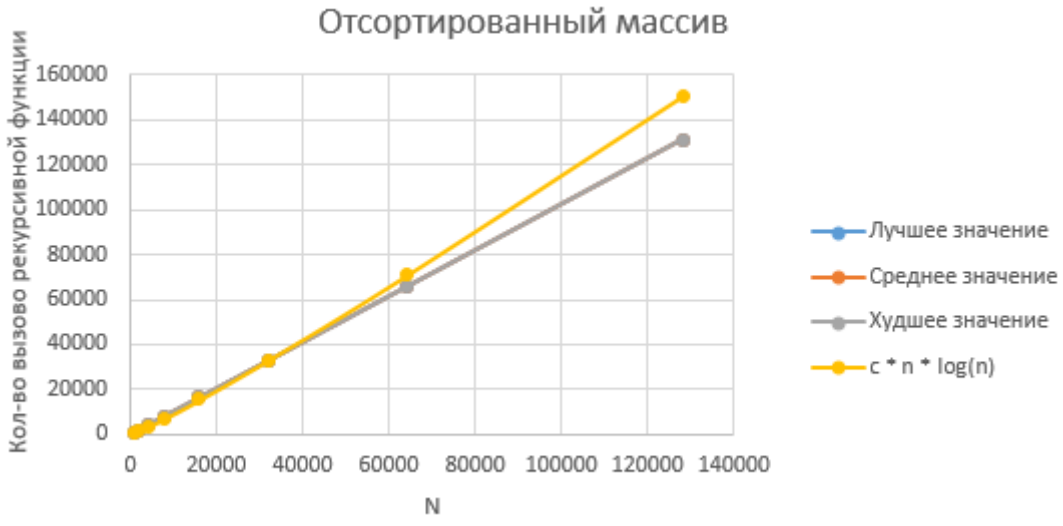
Код функции:

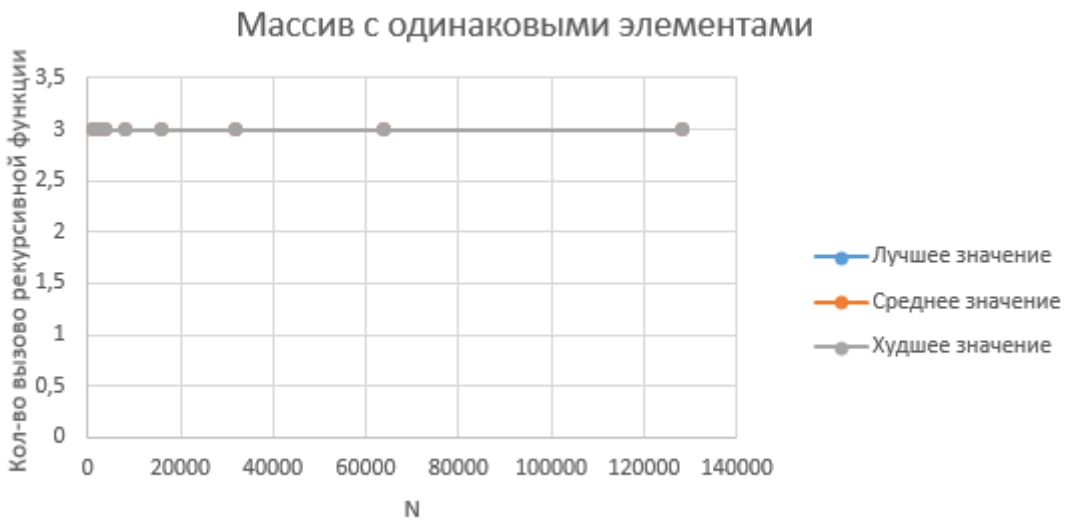
```
def quick_sort(array):

    if len(array) <= 1:
        return array

    elem = array[len(array) // 2]
    left = [i for i in array if i < elem]
    center = [i for i in array if i == elem]
    right = [i for i in array if i > elem]

    return quick_sort(left) + center + quick_sort(right)
```

Были проведены серии тестов на различных массивах:

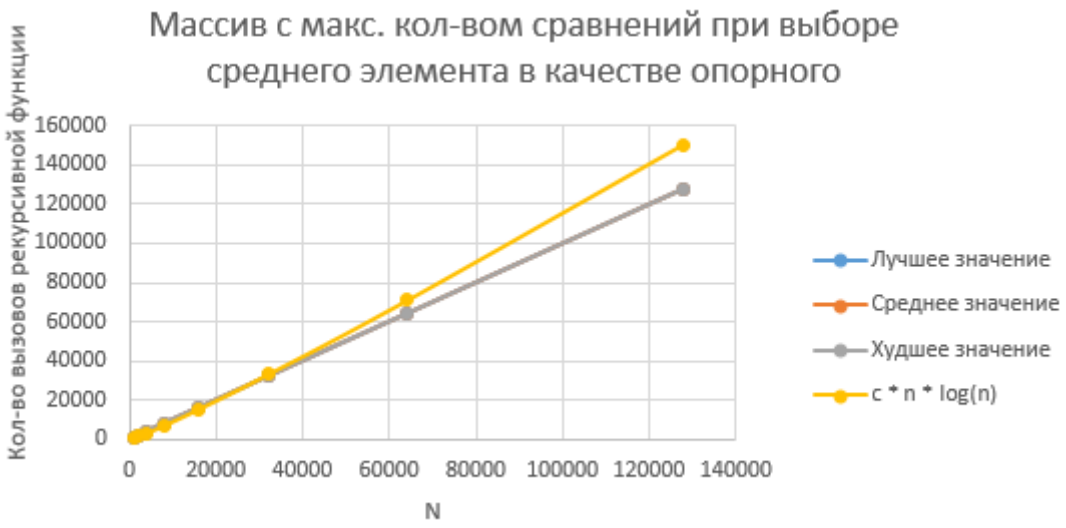
- Отсортированный массив
- Массив с одинаковыми элементами
- Массив с максимальным количеством сравнений при выборе среднего элемента в качестве опорного
- Массив с максимальным количеством сравнений при детерминированном выборе опорного элемента

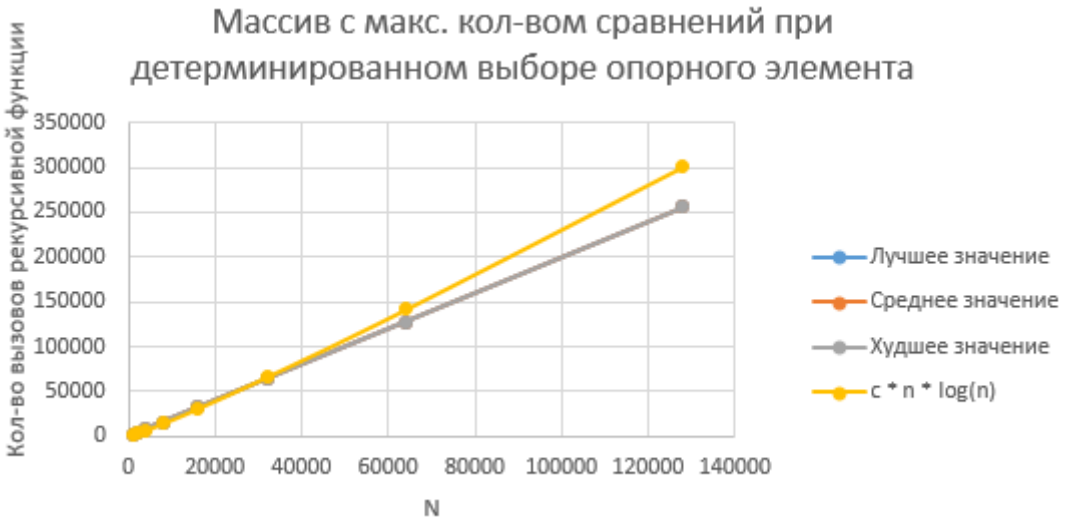
Ниже представлены графики полученные в результате обработки данных извлеченных из тестов:





Для массива с одинаковыми элементами невозможно подобрать такую константу c, что бы график функции c * n * log(n) находился близко к графику худшего случая, т.к. в данном случае график является прямой параллельной оси N.





Заключение.

Алгоритм быстрой сортировки является одним из самых эффективных по времени и по памяти, так как в его реализации используется рекурсивные вызовы. Использование данного алгоритма сортировки в его рекурсивной реализации дает преимущество по времени работы по сравнению с другими алгоритмами. Деградации метода не было обнаружено, чем больше элементов, тем больше вызовов рекурсивной функции.