Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий и систем связи

**Лабораторная работа №4**

**Вариант №1**

Выполнил(и:)

Гусев Я.А.

Проверил

Мусаев А.А.

Санкт-Петербург,

2022

**Задание 1**

На рисунке 1 изображён написанный мной алгоритм сортировки расчёской (combSort).

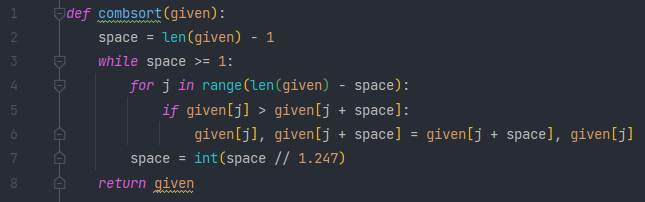


Рисунок 1 – Сортировка расчёской

На рисунке 2 изображён написанный мной алгоритм быстрой сортировки (quickSort)

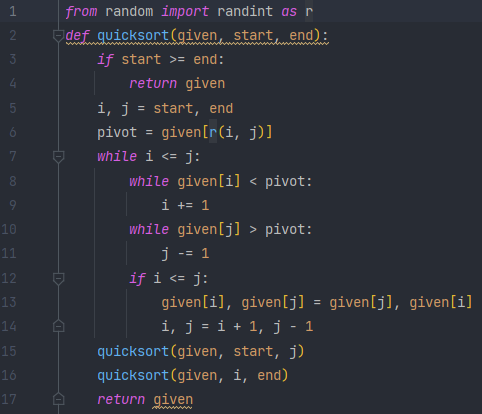


Рисунок 2 – Быстрая сортировка

Далее я импортирую эти функции сортировки в другую программу, в которой можно самому выбрать, какой сортировкой воспользоваться (Рисунок 3).

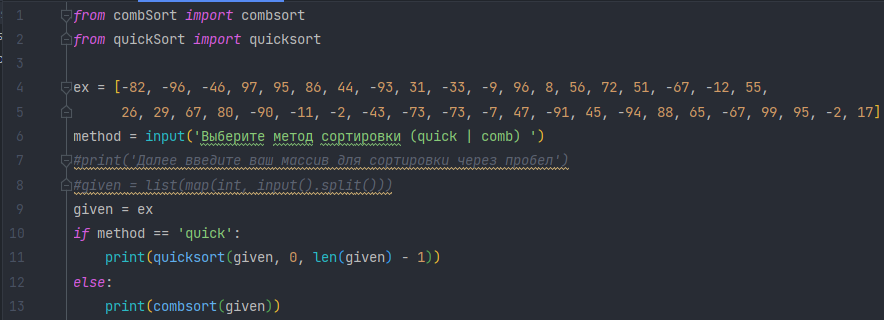


Рисунок 3 – Программа с двумя доступными видами сортировок

Импортирую библиотеки time и timeit (Рисунок 4) и с их помощью узнаю время выполнения обеих сортировок (Рисунок 5).

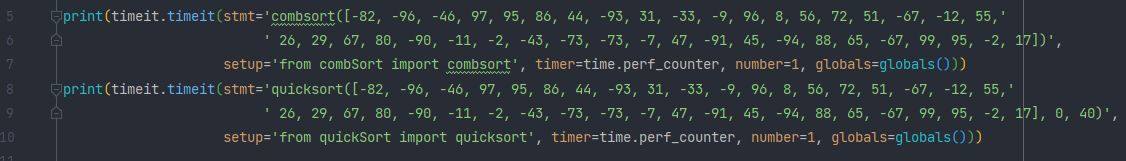


Рисунок 4 – Измерение времени выполнения сортировок с помощью библиотек timeit и time



Рисунок 5 – Время выполнения quickSort и combSort

Как мы видим, быстрая сортировка быстрее сортировки расчёской примерно в 1.7 раз.

**Задание 2**

На рисунке 6 изображён написанный мной алгоритм блочной сортировки (bucketSort).

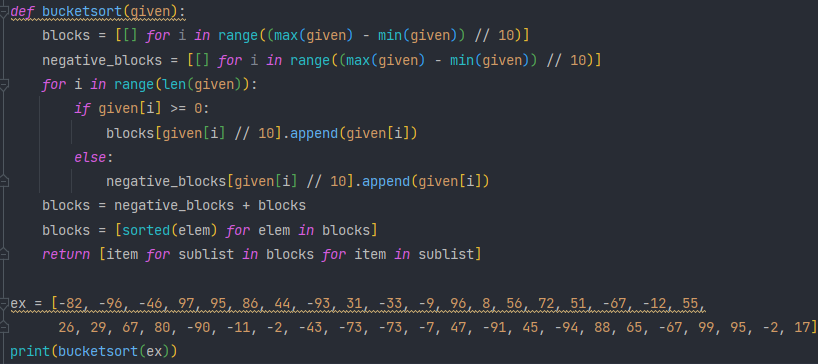


Рисунок 6 – Алгоритм блочной сортировки

Среднее время выполнения – 1.989999999999978e-05.

На рисунке 7 изображена пирамидальная сортировка (heapSort).

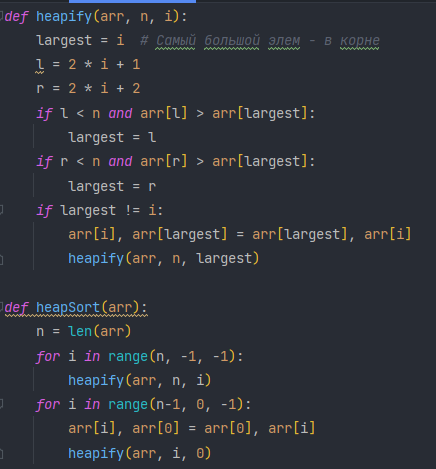


Рисунок 7 – Пирамидальная сортировка

Время выполнения - 1.2599979527294636e-05

**Задание 3**

Для сравнения времени всех реализованных сортировок, я построил их график с помощью библиотек numpy, matplotlib и pandas (Рисунок 8).

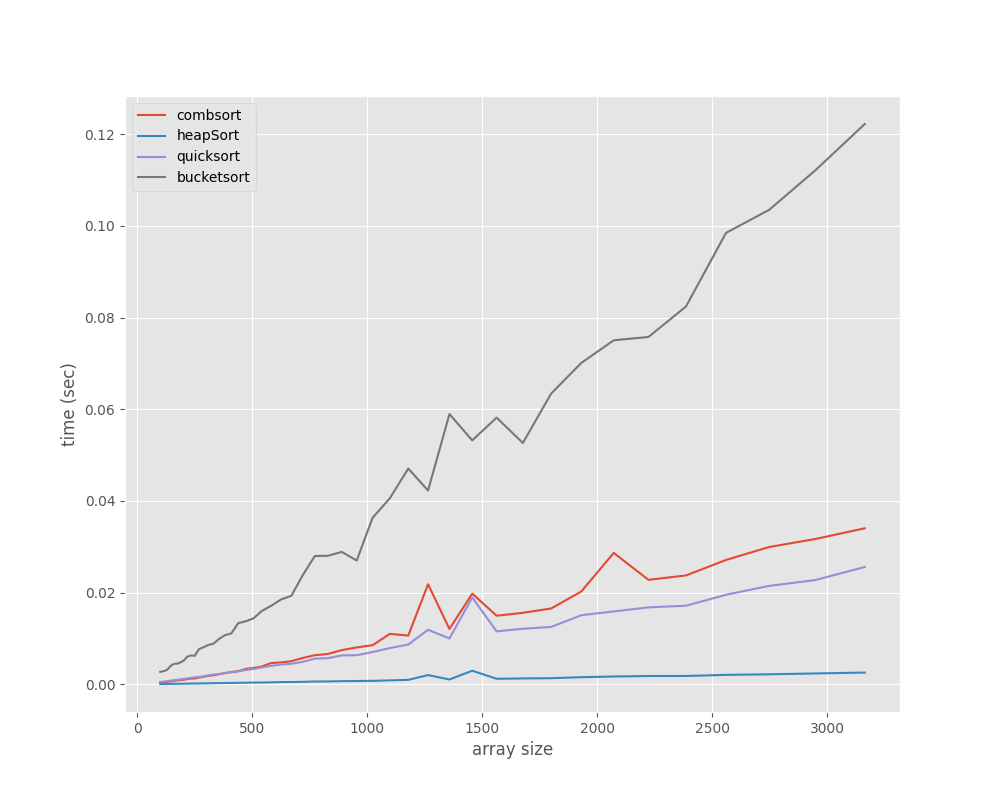


Рисунок 8 – Зависимость времени работы различных алгоритмов сортировок от размера массива входных данных

Как мы видим, quicksort и combsort находятся относительно на одном уровне, однако quicksort всё же быстрее. Самый быстрый алгоритм сортировки – heapsort. Bucketsort же в разы медленнее других алгоритмов, так он эффективен лишь на равномерно распределённых данных (там у него сложность О(n). Мы могли видеть это во втором задании, где блочная сортировка оказалась быстрее сортировки расчёской и быстрой сортировки, потому что входной массив был достаточно равномерно распределён. Для графика же я параллельно с увеличением размера массива увеличивал его распределённость (Рисунок 9) (r – randint), чтобы показать несовершенство bucketsort.



Рисунок 9 – Генерация массива чисел для построения графиков

Общая оценка алгоритмов:

1. Сортировка расчёской. Улучшенная версия сортировки пузырьком. Сложность в лучшем случае – O(nlogn), в худшем сложность равна сложности сортировки пузырьком (O()).
2. Быстрая сортировка. Один из самых быстрых алгоритмов сортировки на массиве с неизвестными данными. Худшее время работы – O(), среднее время – O(nlogn). Из минусов – на почти отсортированном массиве работает так же долго как и на неотсортированном. Из плюсов: короткий алгоритм, малый расход памяти.
3. Блочная сортировка. Как говорилось выше, хорошо работает на удачных входных данных – O(n). В остальных случаях O().
4. Пирамидальная сортировка. В основе лежит построение двоичного дерева. Сложность алгоритма в лучшем, худшем и среднем случае равна O(nlogn), поэтому он является крайне эффективным. Выделяемая память линейна. Из-за сложности алгоритма выигрыш получается только на больших n.