ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 5

«Алгоритмы сортировки»

Выполнил работу

Косарев И. А.

Академическая группа №J3113

Принято

Ходненко И. В.

Санкт-Петербург

2024

**Структура отчёта:**

1. Введение

Цель: сравнить алгоритмы сортировки различной асимптотики и понять, для каких случаев взятые сортировки применимы.

Задачи:

1. Изучить различные алгоритмы сортировок.
2. Написать код взятых сортировок
3. Написать тесты, чтобы убедиться в работоспособности написанных сортировок.
4. Сравнить их по времени выполнения в зависимости от кол-ва сортируемых элементов.
5. Сделать выводы.
6. Теоретическая подготовка

* Cycle Sort — это алгоритм сортировки, который минимизирует количество операций записи, что делает его полезным в ситуациях, когда операции записи являются дорогими или ограниченными. Основная идея Cycle Sort заключается в том, чтобы каждый элемент массива переместить к его правильной позиции, осуществив как можно меньше пересылок.
* Heap Sort — это алгоритм сортировки, который использует структуру данных, называемую кучей (heap). Куча — это бинарное дерево, которое удовлетворяет свойству кучи: для максимальной кучи (max-heap) значение любого узла больше или равно значению его дочерних узлов; для минимальной кучи (min-heap) значение любого узла меньше или равно значению его дочерних узлов. Heap Sort обычно реализуется с использованием максимальной кучи.
* Bucket Sort, или сортировка по корзинам, — это алгоритм сортировки, который работает, разделяя элементы массива на несколько корзин (или бакетов). Затем каждая корзина сортируется отдельно, и содержимое всех корзин объединяется для получения окончательного отсортированного массива. Этот метод особенно эффективен для равномерно распределенных данных.

1. Реализация

По факту реализация разбивается на 3 части: реализация каждой сортировки + реализация тест-кейсов для каждой сортировки.

**Cycle Sort**

Описание действия алгоритма и анализ асимптотики и пространственной сложности представлен на рисунке 1.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 1 - код Cycle Sort

Напишем тест-кейсы (представлены на рисунке 2)

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 2 - тест-кейсы для Cycle Sort

**Heap Sort**

Описание действия алгоритма и анализ асимптотики и пространственной сложности представлен на рисунке 3.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 3 - код Heap Sort

Также напишем тесты (см. рисунок 4):

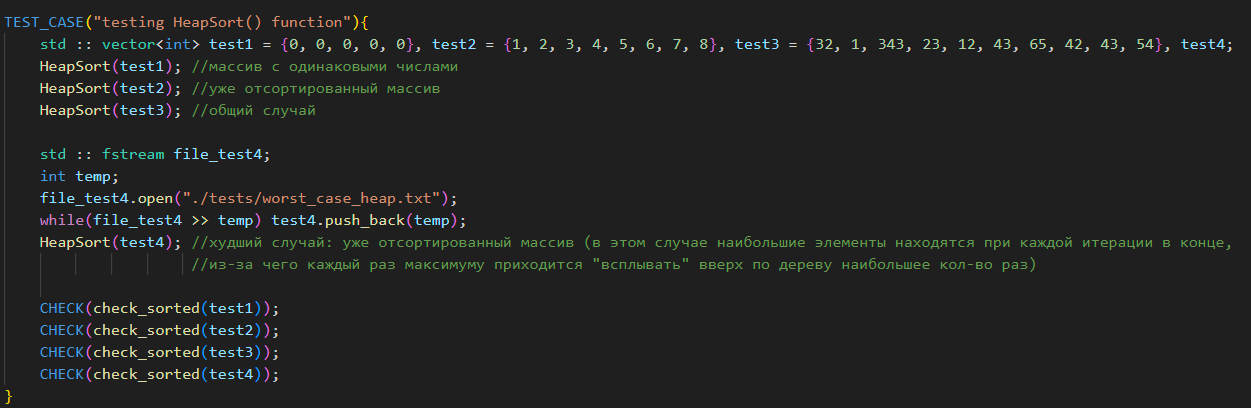


Рисунок 4 - тест-кейсы для Heap Sort

**Bucket Sort**

Принцип действия и анализ асимптотики и пространственной сложности представлены на рисунке 5.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 5 - код Bucket Sort

И также напишем тесты (см. рисунок 6):

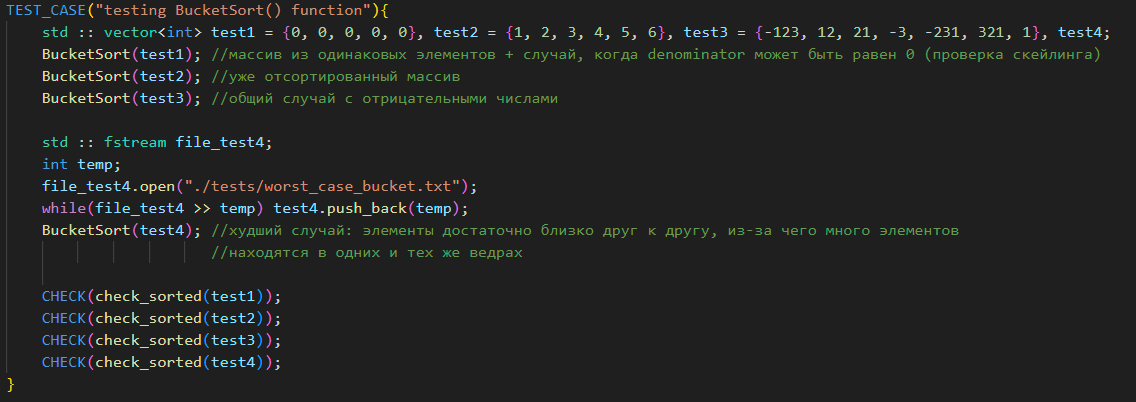


Рисунок 6 - тест-кейсы для Bucket Sort

1. Экспериментальная часть

Для анализа асимптотики построим линейные графики, показывающие зависимость времени выполнения программы от кол-ва элементов.

**Cycle Sort**

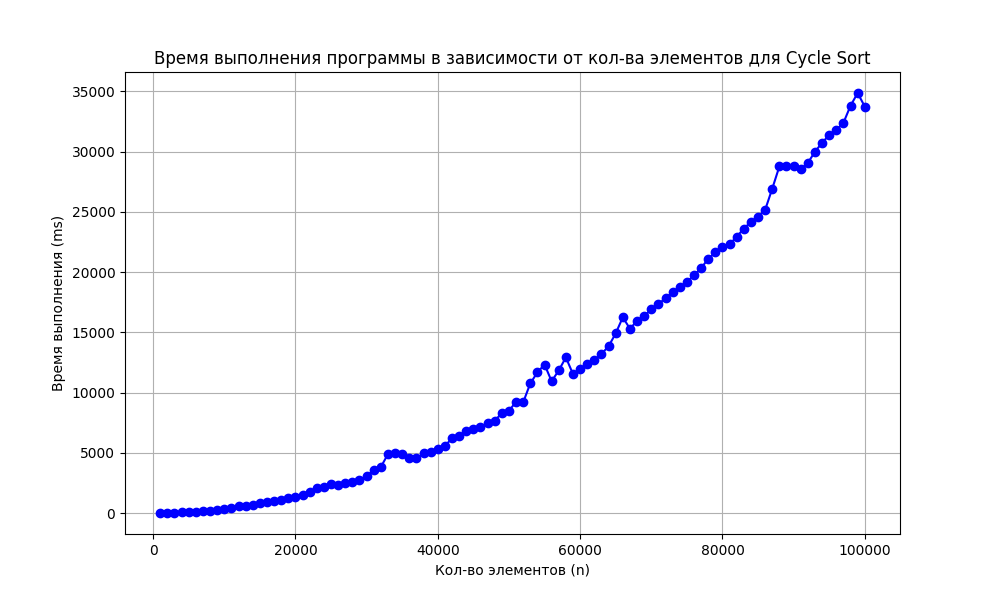


Рисунок 7 - линейный график для Cycle Sort (нетрудно заметить, что график напоминает O(n^2))

**Heap Sort**

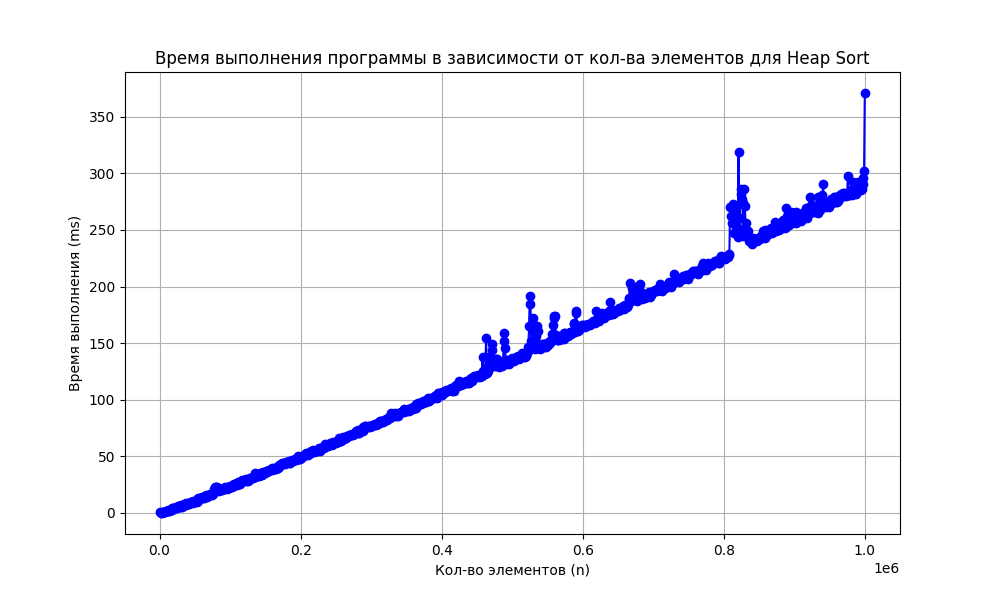
****

Рисунок 8 - линейный график для Heap Sort

**Bucket Sort**

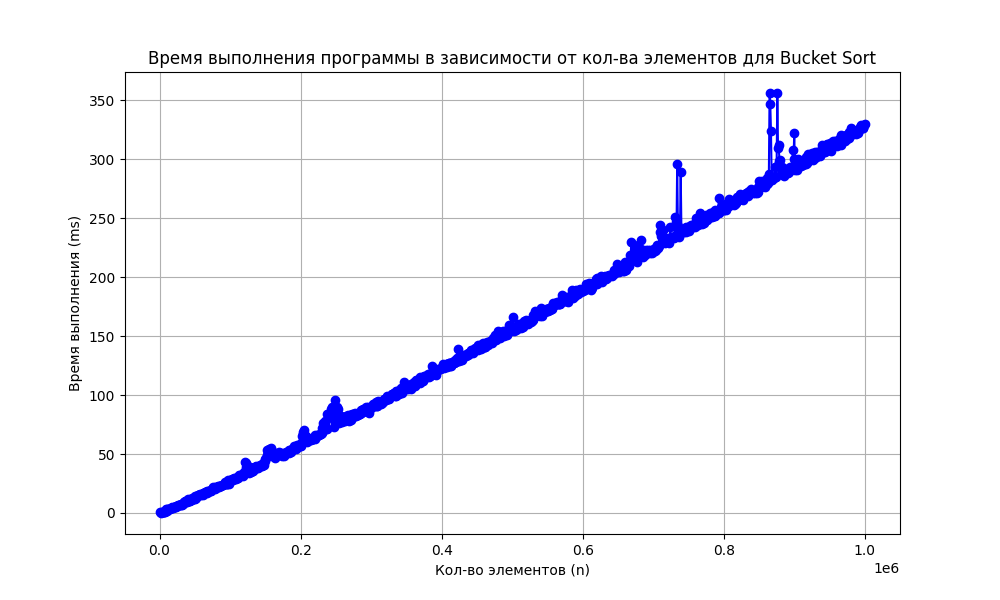


Рисунок 9 - линейный график для Bucket Sort

Также построим графики вида box plot для анализа среднего времени выполнения на наборах 1е4 и 1е5:

**1е4**

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 10 - график вида box plot (нетрудно заметить, насколько более быстрые сортировки опережают Cycle Sort)

**1е5**

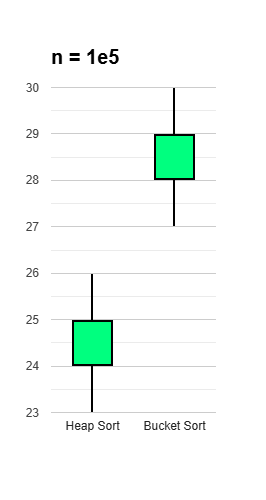
****

Рисунок 11 - график вида box plot для Bucket Sort и Heap Sort

Так как Cycle Sort выполнялся бы на наборе 1е5 примерно 30 с (по тестовым данным, использованным при построении линейных графиков), ее рассматривать бесполезно (так как тогда невозможно было бы сравнить Heap Sort и Bucket Sort). То, что Heap Sort работает быстрее, чем Bucket Sort на наборах 1е5, можно объяснить тем, что в основе сортировок ведер лежит тот же Heap Sort, из-за чего на относительно малых наборах данных из-за вычисления скейлинга и перезаписей в ведра Bucket Sort действительно работает медленнее.

1. Заключение

В целом, теоретическая асимптотика почти полностью совпала с реальной, правда, довольно неожиданно было увидеть то, что Bucket Sort действительно работает немного медленнее, чем Heap Sort при относительно малом кол-ве данных.

Все сортировки имеют практическое применение: Cycle Sort применим в тех ситуациях, когда перезапись на данном носителе данных дорогая, из-за чего необходимо отсортировать данные минимальным кол-вом перестановок (что и обеспечивает сортировка), Heap Sort за счет своей оптимизации применим в общем случае, Bucket Sort работает лучшим образом при относительно равномерно распределенных данных (хотя, конечно, можно создать взвешенный скейлинг, который может в некоторой степени оптимизировать с первого взгляда неравномерно распределенные данные, однако это можно рассмотреть в качестве дальнейших исследований).

Выбросы присутствуют в относительной степени у всех алгоритмов, что можно объяснить тем, что у асимптотики каждого алгоритма есть некоторая зависимость от данных (например, если Cycle Sort’ом попытаться отсортировать массив, уже отсортированный по невозрастанию, это будет более затратно по времени, чем отсортировать уже отсортированный массив), а также тем, что так как при сборе результатов тестирований на процессоре моего компьютера происходили другие процессы, которые могли повлиять на производительность в тот конкретный момент времени.

1. Приложения

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 12 - полный листинг Cycle Sort

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 13 - полный листинг Heap Sort

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 14 - листинг основной части Bucket Sort

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 15 - часть с тестами Bucket Sort