ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 5

«Сортировка»

Выполнил работу

Феоктистов Виктор

Академическая группа C3100

Принято

Ментор Вершинин Владислав Константинович

Санкт-Петербург

2024

1. Введение

Цель: реализовать три алгоритма сортировок соответствующие следующим требованиям:

1. Алгоритм сортировки с лучшей сложностью - O(N^2), Пространственная сложность - O(1);
2. Алгоритм сортировки со средней до O(N^2), Пространственная сложность - до O(N);
3. Алгоритм сортировки, со средней сложностью O(N\*k), Пространственная сложность - до O(N\*k), где k << N (значительно меньше N).

Задачи:

1. Реализовать алгоритм Шейкерной сортировки;
2. Реализовать алгоритм Блинной сортировки;
3. Реализовать алгоритм Timsort;
4. Выполнить тесты;
5. Отладить ошибки;
6. Реализовать линейный график работы алгоритмов;
7. Реализовать box plot для каждого алгоритма.
8. Теоретическая подготовка

Шейкерная сортировка является разновидностью пузырьковой сортировки. Отличие заключается в том, что массив рассматривается как слева направо, так и справа налево. Также ограничиваются границы просмотра в двух краях массива. Худшее время: O(n^2), среднее время: O(n^2), лучшее время: O(n^2).

Основная идея сортировки расческой в том, чтобы первоначально использовать достаточно большое расстояние между сравниваемыми элементами и по мере упорядочивания массива уменьшать это расстояние вплоть до минимального. Первоначальный разрыв между сравниваемыми элементами лучше брать с учётом специальной величины, называемой фактором уменьшения, который может быть, например, 1.25. Сначала расстояние между элементами максимально, то есть равно размеру массива минус один. Затем, пройдя массив с этим шагом, необходимо поделить шаг на фактор уменьшения и пройти по списку вновь. Так продолжается до тех пор, пока разность индексов не достигнет единицы. В этом случае сравниваются соседние элементы как и в сортировке пузырьком, но такая итерация одна. Худшее время: O(n^2), среднее время: Ω(N^2/2^p), лучшее время: O(nlogn).

Timsort - гибридный алгоритм сортировки. Гибридный, потому что сочетает сортировку вставками и сортировку слиянием. Худшее время: O(nlogn), среднее время: O(nlogn), лучшее время: O(n).

1. Реализация

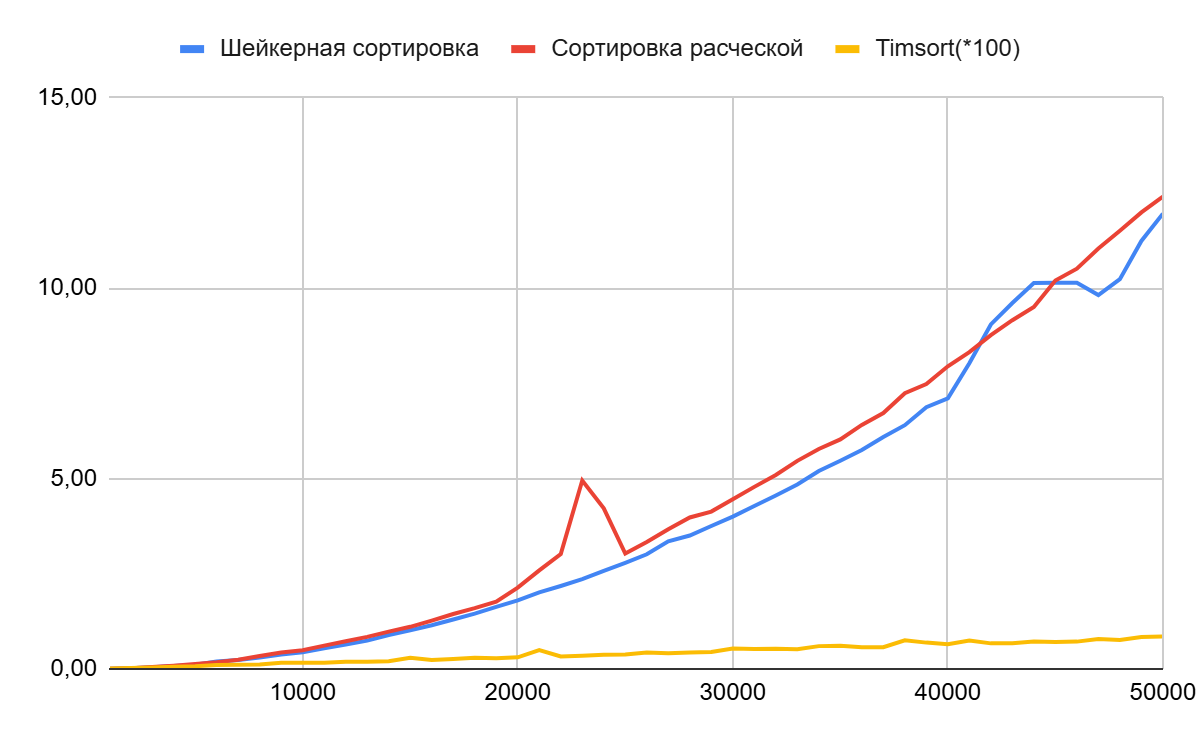
Для шейкерной сортировки сортировки используется два цикла for, перебирающие массив с разных сторон. Переменные left и right определяют границы перебора. Обмен элементов производится с помощью функции swap (приложение A).

Сортировка расчёской обменивает элементы на определенном расстоянии. Шаг вычисляется с помощью специального коэффициента 1,25 на который сначала делится длина массива, после предыдущий шаг (приложение B).

Для timsort реализован алгоритм сортировки слиянием и алгоритм сортировки вставками. Массив разбивается на меньшие массивы длиной 32 (оптимальная рассчитанная длина). Каждый новый массив сортируется вставками, после чего эти куски объединяют с помощью алгоритма слияния (приложение C).

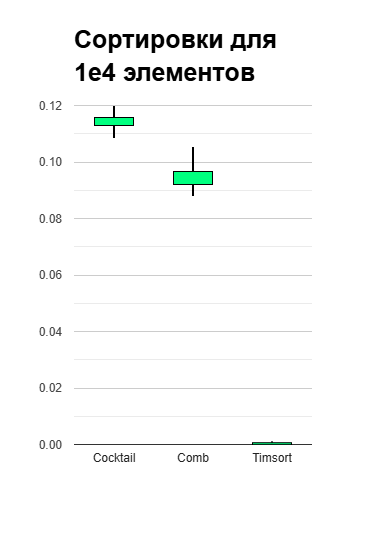
Экспериментальная часть

Для оценки эффективности алгоритма было рассчитано время для работы алгоритмов, для массивов от 1000 до 50000 с шагом в 1000. Результат представлен на изображении №1.



Изображение №1 - График работы алгоритмов

Для анализа времени работы алгоритмов были созданы box plot для 1e4 и 1e5 элементов. Результат представлен на изображении №2, №3.



Изображение №2 - Время алгоритмов для 1e4 элементов



Изображение №3 - Время алгоритмов для 1e5 элементов

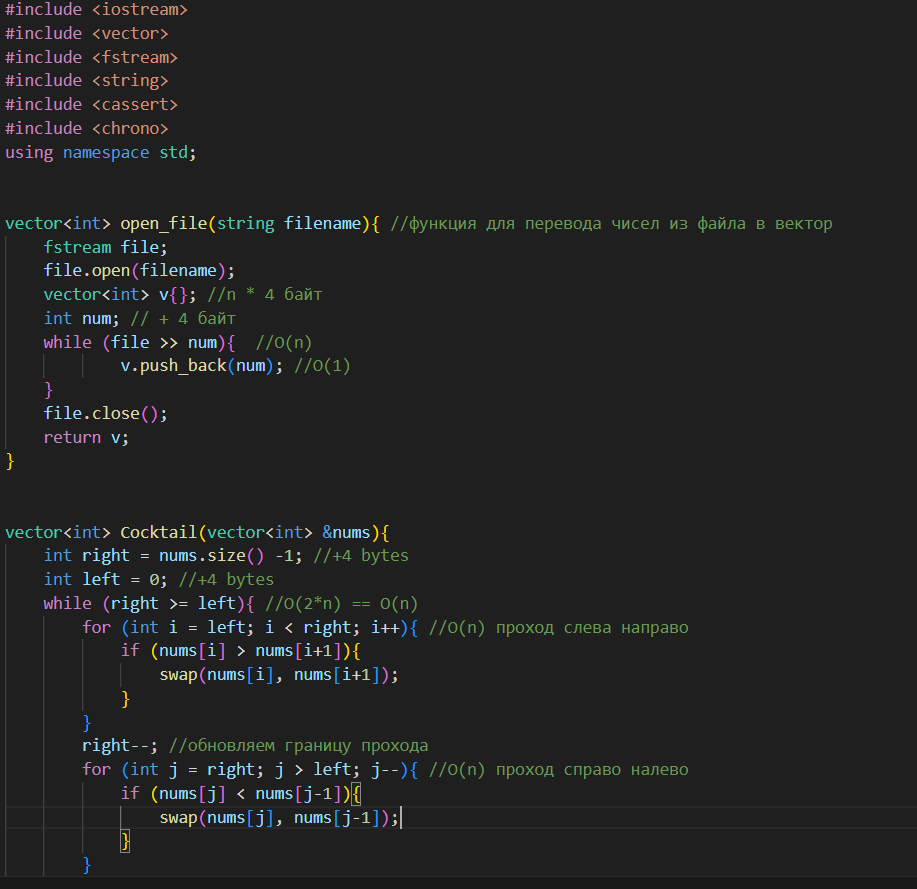
Графики показали, что самым эффективным алгоритмом является timsort, самым неэффективным - шейкерная сортировка.

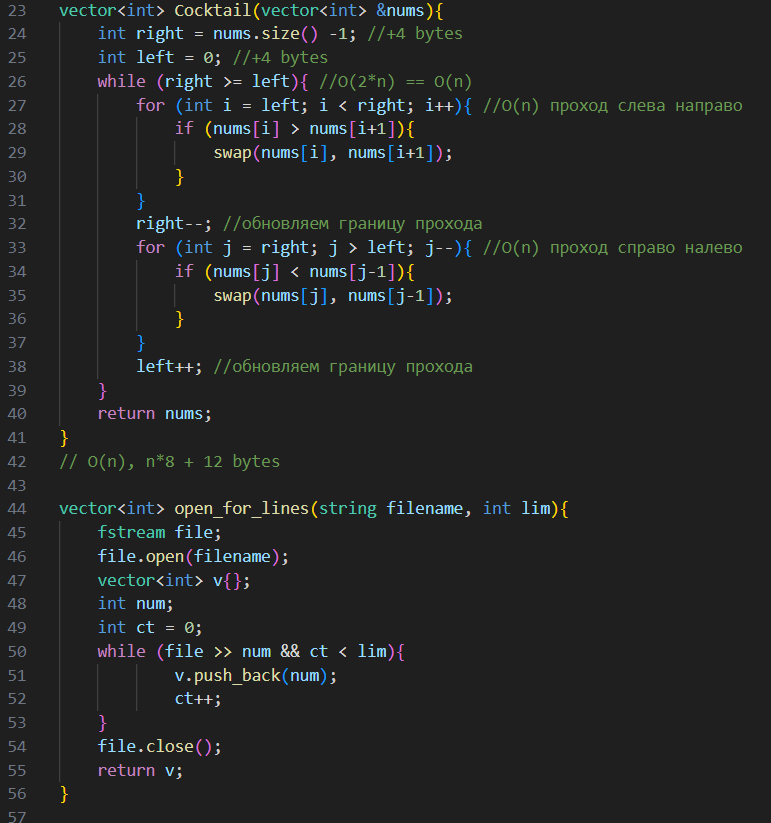
2. Заключение

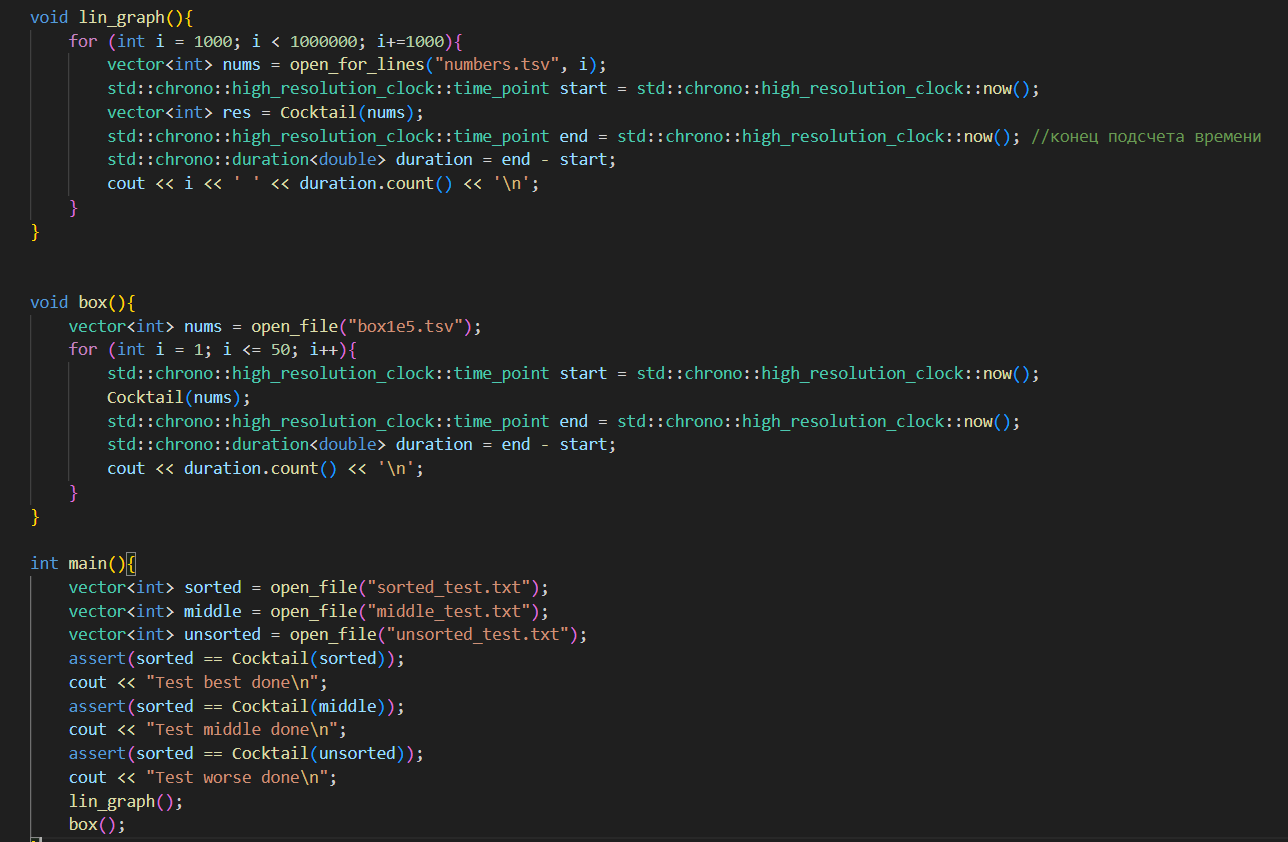
В ходе выполнения работы мною был реализованы три алгоритма сортировки. Цель работы была достигнута путём реализации задач. В качестве дальнейших исследований можно предложить оптимизацию алгоритма с точки зрения уменьшения затрат использования памяти, а также найти возможность более раннего завершения алгоритмов.

Приложение A

shake.cpp

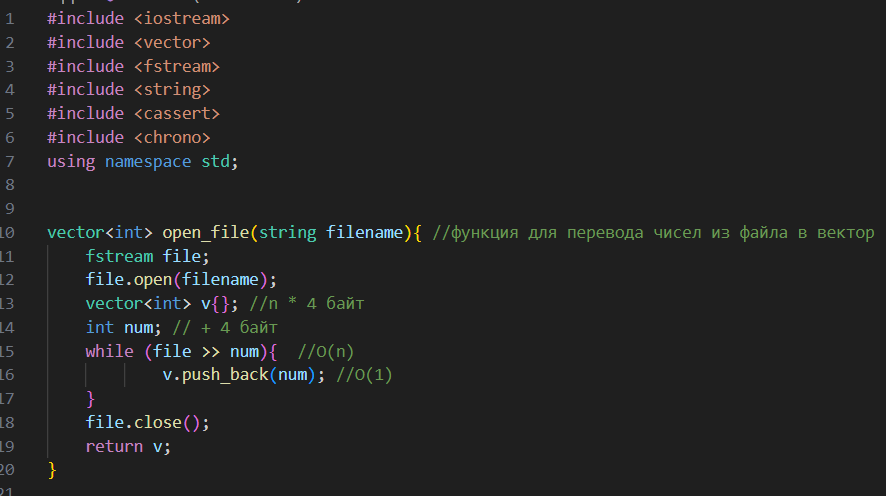


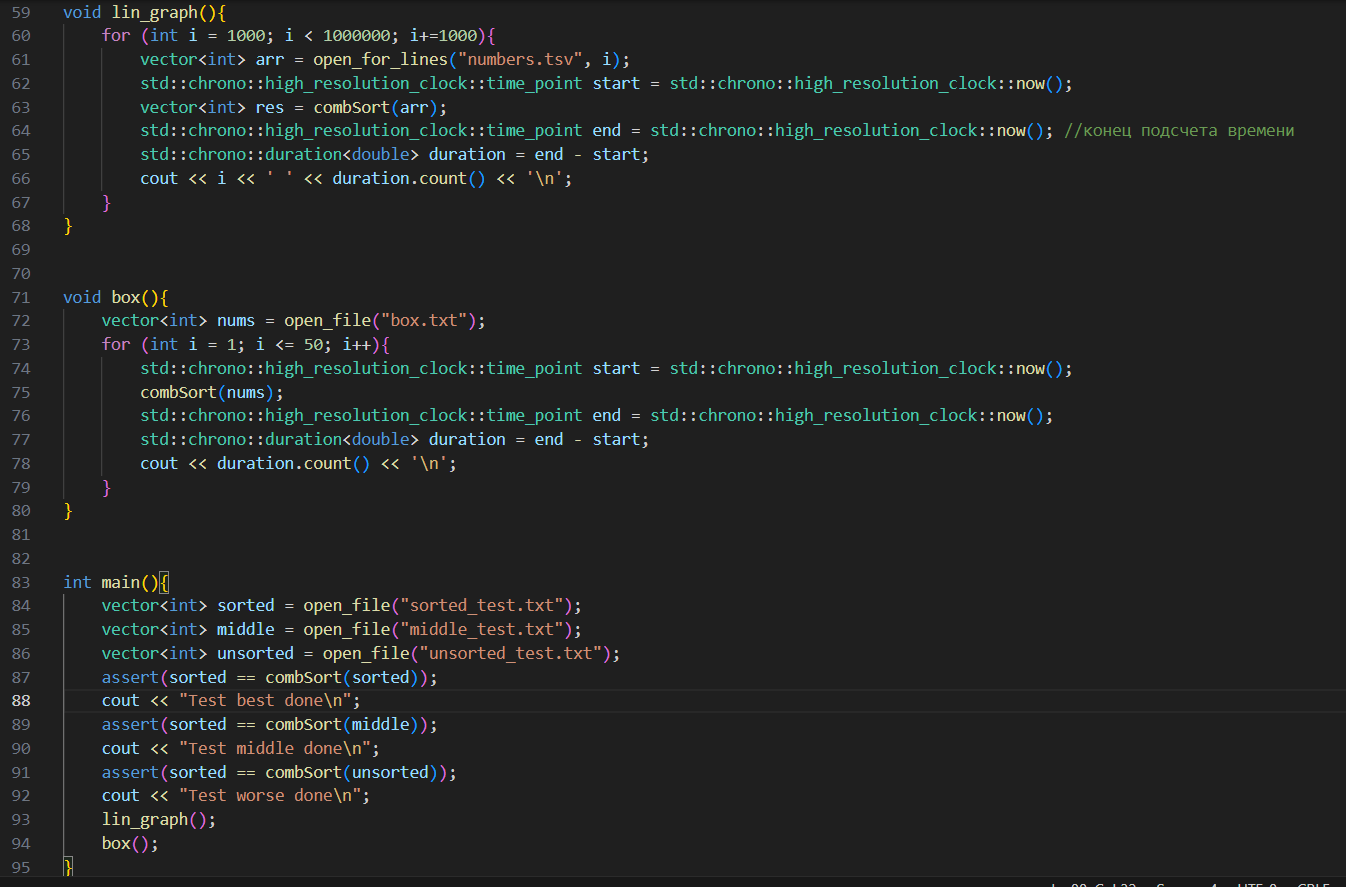
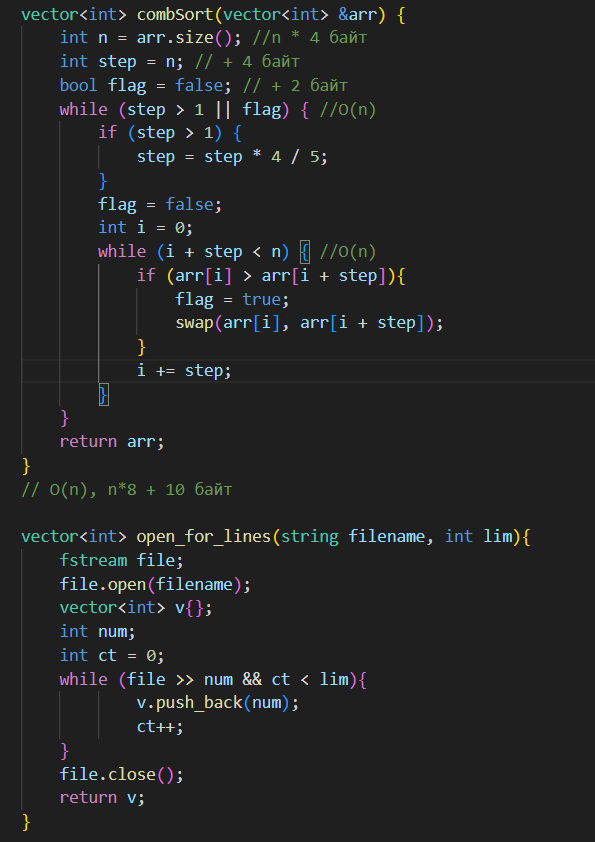




Приложение B

comb.cpp





Приложение C

timsort.cpp

