ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 5

«Алгоритмы сортировки»

Выполнил работу

Трифонов Василий

Академическая группа № J3113

Принято

Должность, звание Фамилия Имя преподавателя

Санкт-Петербург 2024

**Структура отчёта:**

* Введение

Цели работы - разобраться в работе алгоритмов сортировки, выяснить какие есть виды сортировки, узнать в каких ситуациях необходимо применять определенные сортировки.

* Теоретическая подготовка

1. Изучение материалов связанных с алгоритмами сортировки (таблицы на различных сайтах о скорости и стабильности алгоритмов). Прочитать про самые используемые сортировки.
2. Выбрать алгоритмы на реализацию. (CombSort, TimSort, BucketSort)
3. В работе использовались: double, long int, std::vector, std::fstream, insertsort, merge sort

* Реализация

1. Изучение материалов о сортировке CombSort.
2. Особенностью данного алгоритма является, работа не с соседними элементами, а элементами на расстоянии, которое итеративно сходится к единице. Таким образом, теоретически, мы ускоряем работу сортировки, по сравнению с той же сортировкой пузырьком, оптимизацией которого является данный алгоритм.
3. Написание кода алгоритма
4. Написание тестов и тестирование кода сортировки расческой (CombSort)
5. Изучение материалов о сортировке TimSort.
6. В идее алгоритма лежит следующее утверждение, любой не отсортированный массив состоит из уже отсортированных частей. Следовательно сортировка сводится к 3 пунктам
   1. Поиск отсортированного подмассива. А затем увеличение его до примерного среднего размера (от 32 - 64 элементов), путем добавления не отсортированных элементов.
   2. Сортировка *сортировкой вставками* подмассива найденного на первом этапе
   3. Слияние отсортированных подмассивов в единый массив с помощью модифицированной *сортировки слиянием*.
7. Написание кода алгоритма
   1. Написание функции, которая ищет оптимальный размер подмассивов
   2. Написание *сортировки слиянием*
   3. Написание основного блока алгоритма
   4. Написание функции соединения подмассивов
8. Написание тестов и тестирование кода TimSort
9. Изучение материалов о сортировке Bucket sort
10. Идея данного алгоритма заключается в том, что все элементы входного массива можно разделить по какому то признаку на N подмассивов. Где N - размер входного массива. Затем эти массивы можно отсортировать любой сортировкой и вставить в изначальный массив в нужном порядке. Алгоритм хорошо себя показывает на большом объеме данных с нормальным распределением.
11. Написание кода алгоритма
12. Написание тестов и тестирование кода Bucket sort

* Экспериментальная часть

В этом разделе вам необходимо привести результаты работы вашего алгоритма, с таблицами и графиками, демонстрирующими выполнения алгоритма с различными условиями и наборами данных. Оценивается производительность и сравниваются результаты с теоретическими оценками.

Подсчёт по памяти (только для циклов и сложных структур) – как в лабораторной работе №2.

* Подсчет памяти.
  + Для алгоритма Comb sort используется O(1) дополнительной памяти. Точнее 8+(4+4)+1 - на создание дополнительных переменных.
  + Для алгоритма TimSort каждый раз создается дополнительный массив для слияния подмассивов массивов. Размер доп памяти - O(k) \* 4

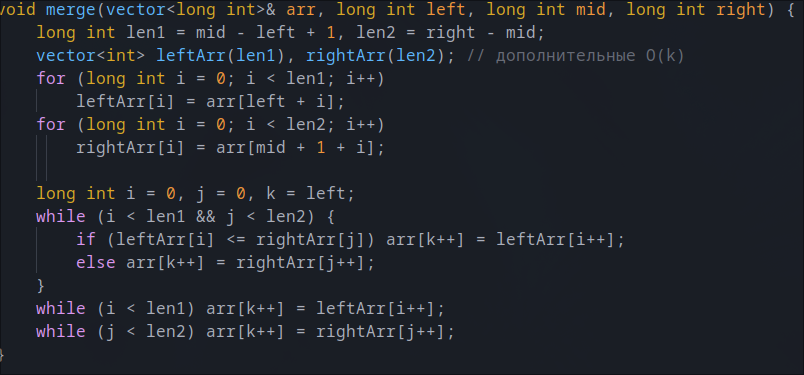


Рисунок 1 - код реализации слияния подмассивов

* + Для алгоритма Bucket sort

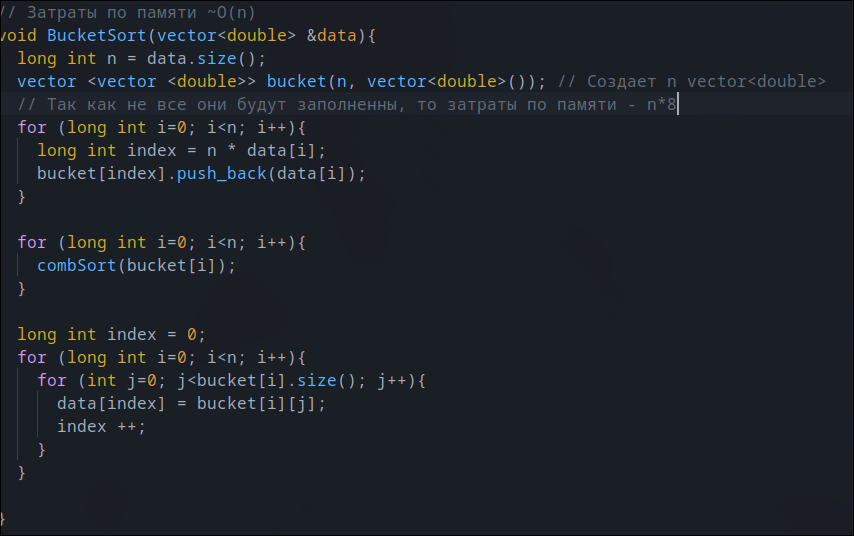


Рисунок 2 - создание дополнительного массива для Bucket sort

* Подсчет асимптотики
  + Comb sort

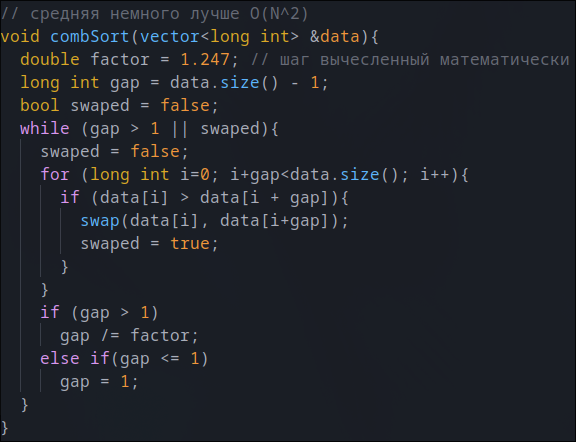


Рисунок 3 - асимптотика comb sort

* + TimSort

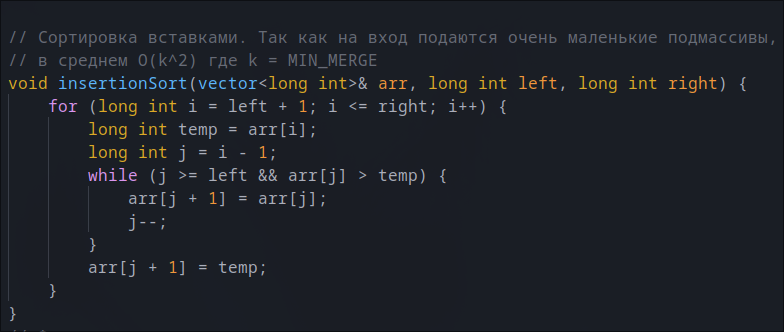


Рисунок 4 - Часть асимптотики TimSort

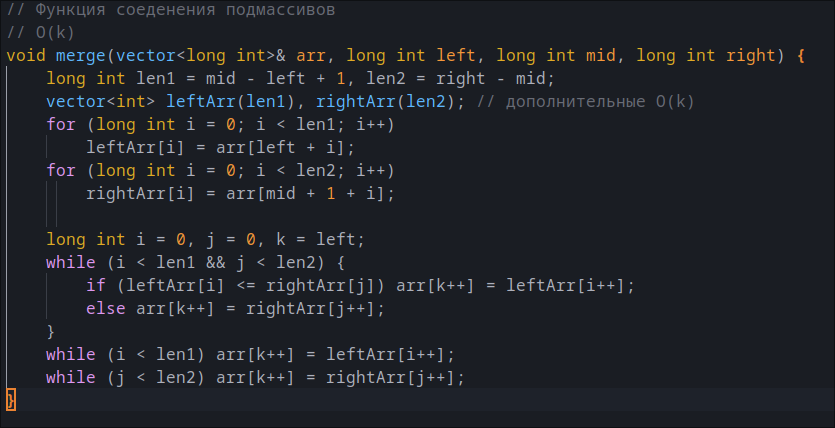


Рисунок 5 - часть асимптотики TimSort

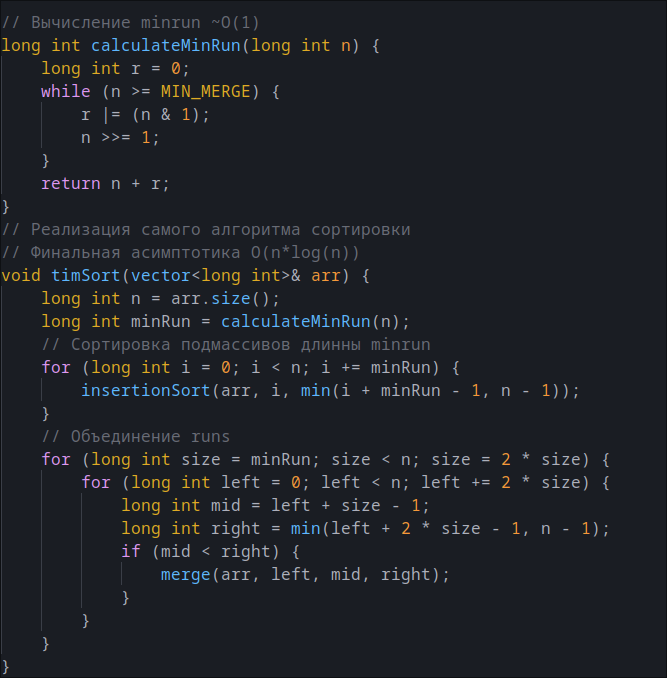


Рисунок 6 - часть асимптотики TimSort

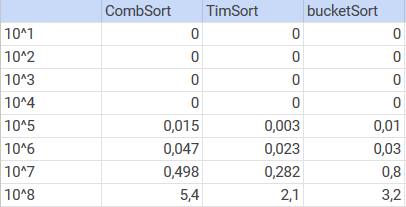
* + Bucket sort

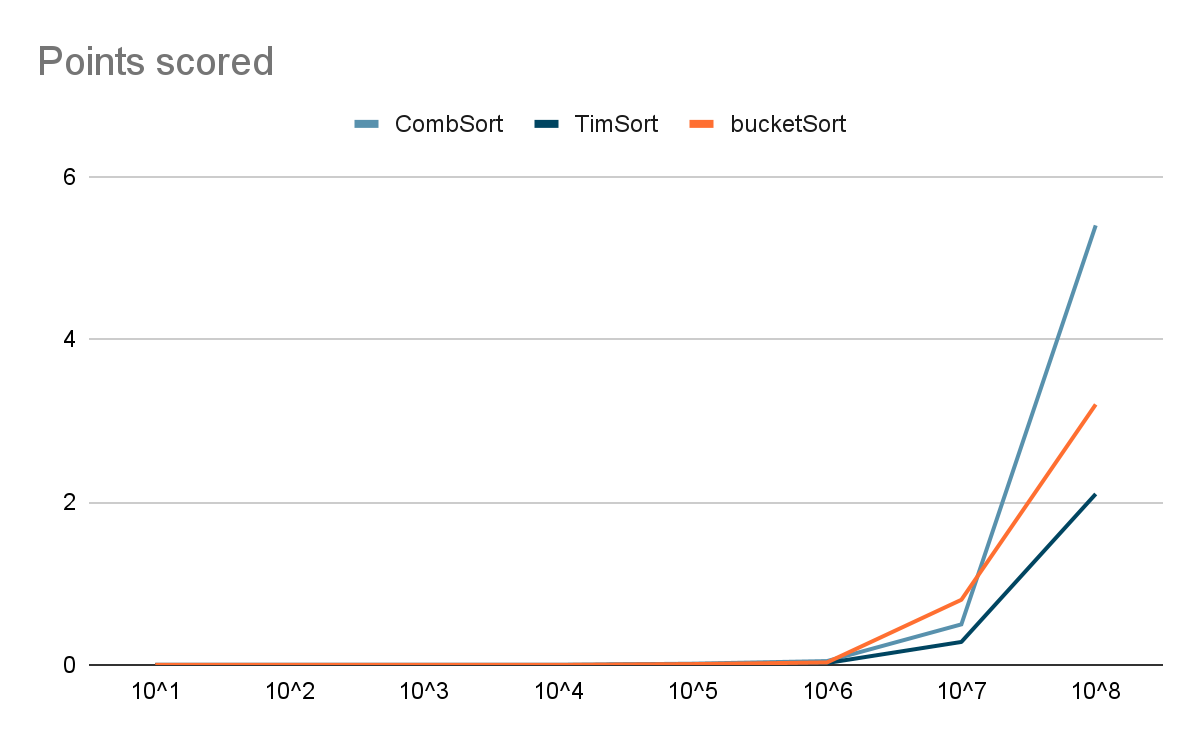


Рисунок 7 - асимптотика BucketSort

* Графики зависимости времени от числа элементов. Пример выполнения:

Согласно требованиям моего варианта, на вход к моему алгоритму подаётся до 25 элементов. Теоретически заданная сложность задачи составляет O(2^N) и более. Для тестирования алгоритма была собрана статистика, приведенная в таблице №\*.

  
Таблица №1 - Подсчёт сложности реализованного алгоритма



Изображение №8 - График работы алгоритма

График представляющий визуально удобный формат данных из таблицы №1 представлен на изображении №8.

* Заключениt

«В ходе выполнения работы мною были реализованы алгоритмы сортировки combSort timSort BucketSort. Цель работы была достигнута путём тестирования на массивах данных различной длины. Полученные результаты для BucketSort не совпадают с теоретическими оценками сложности алгоритма, скорее всего это связано с реализацией и переписыванием vector. Для остальных алгоритмов полученные результаты совпадают с теоретическими.

В качестве дальнейших исследований можно предложить оптимизацию алгоритма с точки зрения уменьшения затрат использования памяти и времени.»