ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 5

«Название лабораторной работы Сортировки (Bubble, Quic, Radix Sort)»

Выполнил работу

Романов Сергей

Академическая группа №j3112

Принято

Практикант Дунаев Максим

Санкт-Петербург

2024

**Структура отчёта:**

1. Введение

**Цель** - В данной лабораторной работе мне нужно написать три алгоритма, которые сортируют массив. Это (Bubble) Алгоритм сортировки с лучшей сложностью - O(N^2) Пространственная сложность - O(1); (Quic) Алгоритм сортировки со средней до O(N^2) Пространственная сложность - до O(N); (Radix) Алгоритм сортировки, со средней сложностью O(N\*k); Пространственная сложность - до O(N\*k)  
где k << N (значительно меньше N)

1. Теоретическая подготовка

Знание контейнеров vector, array

Знание файловых и строковых потоков

Умение считать сложность алгоритма

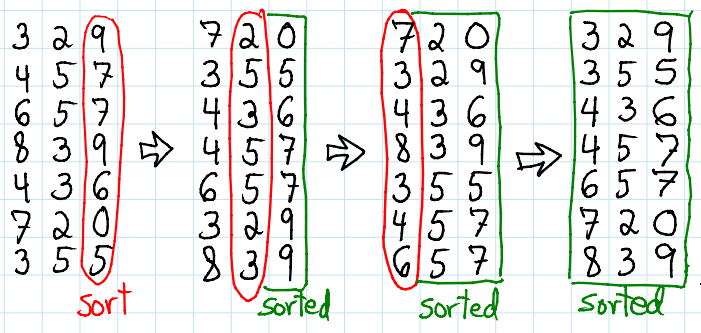
Умение считать выделенную оперативную память

1. Реализация

BubbleSort: Это самый просто алгоритм сортировки. Он хорошо работает с массивами, которые состоят из не большого количества элементов. Идея такая: Перебираются весь массив с элементами и сравниваются два соседа. Если сосед справа больше соседа слева, то они меняются местами.

QuicSort: Этот алгоритм уже по хитрее и один из самых распространённых алгоритмов сортировки. Это золотая серединка, он хорошо работает как с маленькими массивами, так и с большими. Идея такая: Берётся опорный элемент. Далее берутся число которые больше и которые меньше опорного элемента. Затем рекурсия, которая и сортирует весь массив. Затем изначальный массив заменяется на отсортированный массив с числами которые меньше опорного элемента, добавляется сам опорный элемент, и на конец присоединяется массив с элементами, которые больше опорного элемента

RadixSort:



1. Экспериментальная часть

Подсчёт по памяти (только для циклов и сложных структур)

Bubble: O(1)

Quic: от O(N) до O(log N)

Radix: O(W + N) W - количество бит, требуемых для хранения каждого ключа.

Подсчёт асимптотики (только для циклов и сложных структур)

Bubble: O(N^2)

Quic: от O(N^2) до O(log N)

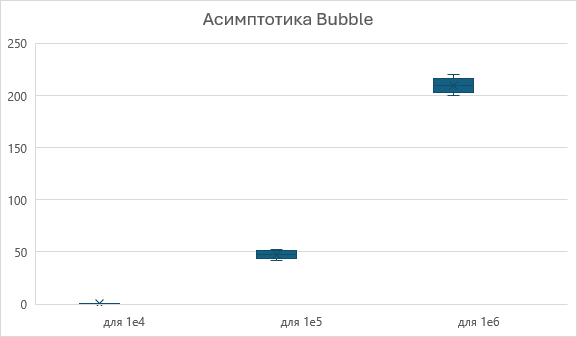
Radix O(W + N) W - количество бит, требуемых для хранения каждого ключа.

Таблица №1 - Подсчёт времени работы алгоритмов

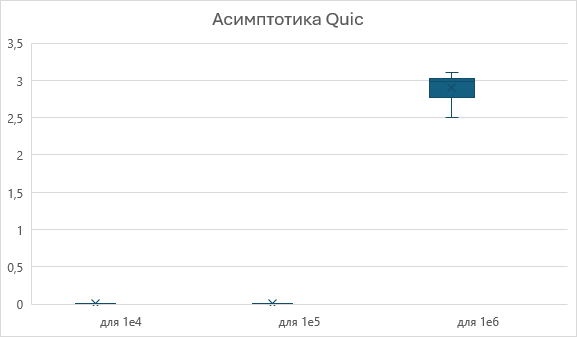
|  |  |  |  |
| --- | --- | --- | --- |
| Размер массива | Bubble | Quic | Radix |
| 1000 | 0,114 | 0,004 | 0,004 |
| 10000 | 0,93 | 0,009 | 0,003 |
| 100000 | 47,39 | 0,007 | 0,02 |
| 1000000 | 420 | 2,984 | 0,171 |

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

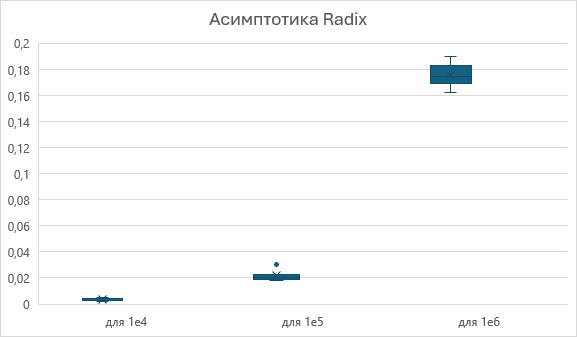
Изображение №1 - График работы алгоритмов



Изображение №2 – Свечной график асимптотики алгоритмов



Изображение №3 – Свечной график асимптотики алгоритмов



Изображение №4 – Свечной график асимптотики алгоритмов

Из графика видно, что Bubble быстро работает с маленьким количеством значений. И с увеличением он резко увеличивает время работы. Например миллион значений он считал больше 7 минут! Quic уже работает намного быстрее. А Radix вообще гонщик)

1. Заключение

Данная лабораторная работа помогла мне освоить алгоритмы сортировки массивов и выбрать для себя самые удобные и практичные. Так Bubble поможет в массивах с маленьком количеством элементов. А Quic и Radix уже помогут с большими данными.