Лабораторная работа №1 по курсу дискретного анализа: сортировки за линейное время

Выполнил студент группы 08-209 МАИ Кривошапкин Егор Борисович.

Условие

Требуется разработать программу, осуществляющую ввод пар «ключ-значение», их упорядочивание по возрастанию ключа указанным алгоритмом сортировки за линейное время и вывод отсортированной последовательности. Вариант задания определяется типом ключа (и соответствующим ему методом сортировки) и типом значения:
Поразрядная сортировка.

Тип ключа: числа от 0 до $2^{64} - 1$.

Тип значения: числа от 0 до $2^{64} - 1$.

Вариант: 3-3

Метод решения

Поразрядная сортировка — это алгоритм сортировки, который обрабатывает числа поразрядно, начиная с младшего разряда (LSD) или старшего разряда (MSD). Она использует устойчивую сортировку (например, сортировку подсчетом) для каждого разряда.

Алгоритм поразрядной сортировки (LSD):

1. Найти максимальное число в массиве и определить количество разрядов d.
2. Для каждого разряда i от младшего к старшему выполнить устойчивую сортировку (например, сортировка подсчётом) по текущей цифре.

Итоговая сложность:

$O(d \cdot (n + k))$, где n — количество элементов, k — основание системы счисления (обычно 10), d — количество разрядов в максимальном числе.

Описание программы

Моя программа состоит из 1 файла: main.cpp. В качестве структуры данных я создал структуру Pair, в которой хранятся ключ и значение.

Алгоритм программы состоит из 3 частей:

1. Сбор входных данных
2. Поразрядная сортировка
3. Вывод данных

Дневник отладки

Изначально я не догадался, что, когда мы сравниваем конкретный разряд числа, у всех чисел с длиной меньше рассматриваемого разряда соответствующее значение будет равно 0. В целях решения проблемы перемещения коротких чисел в начало исходного вектора, в первой версии программы массив temp имел 11 ячеек, в которую попадали все числа с длиной меньше номера текущего разряда.

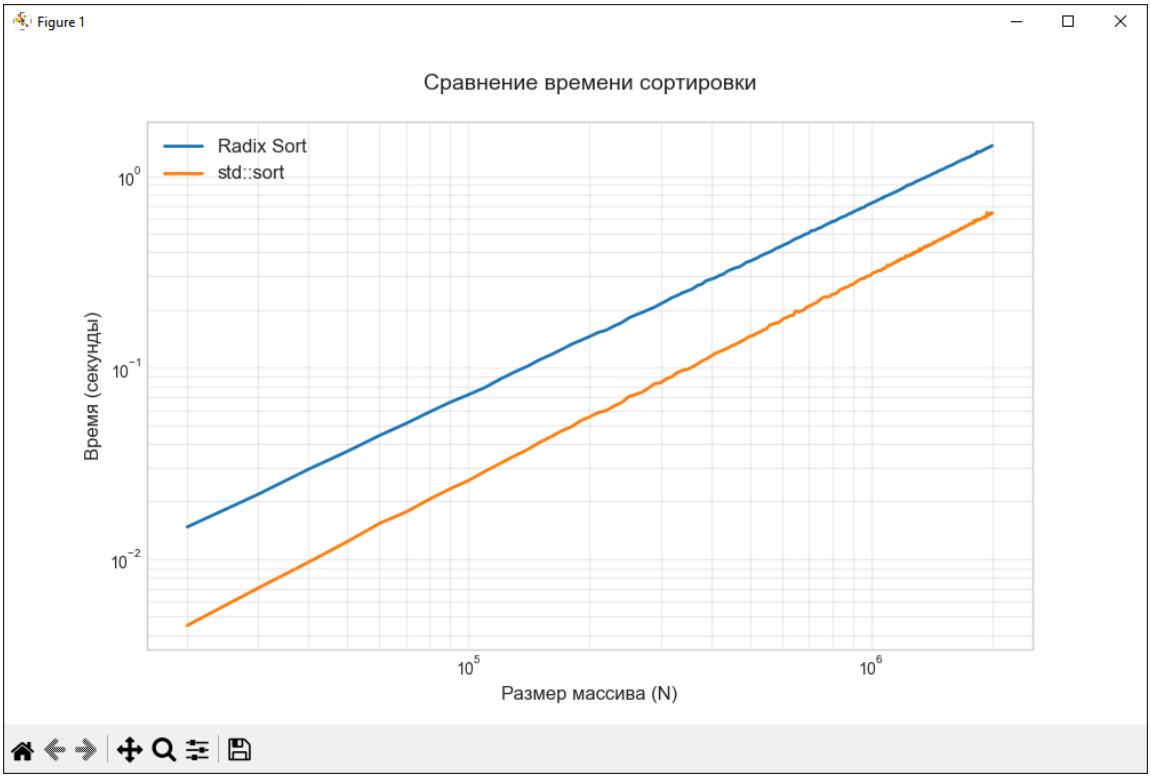
Тест производительности

После проведения бенчмарков, получились следующие результаты:

Std::sort took 3.53755 seconds.

Radix sort took 7.28143 seconds.

График сравнения времени выполнения Radix sort и std::sort.



В среднем поразрядная сортировка оказалась в 2 раза медленнее std::sort. Скорее всего это связано с тем, что в моей реализации очень часто выполняется трудозатратная операция деления, например, для получения значения конкретного разряда.

Выводы

Известно, что сортировка подсчетом эффективно лишь тогда, когда диапазон ключей достаточно мал. Однако, что делать, если диапазон ключей во много раз превышает количество элементов? Тогда на помощь приходит поразрядная сортировка. Пусть имеется n d-значных чисел, в которых каждая цифра принимает одно из k возможных значений. Тогда алгоритм Radix sort позволяет выполнить корректную сортировку этих чисел за время $O(d \cdot (n + k))$, если устойчивая сортировка, используемая в алгоритме имеет время работы $O(n + k)$. Сам алгоритм довольно прост: проходя по всем разрядам мы сортируем по массив по каждому разряду выбранной сортировкой.