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

Выполнил студент группы 08-208 МАИ *Ибрагимов Далгат.*

Условие

- **Общая постановка задачи:** Требуется разработать программу, осуществляющую ввод пар «ключ-значение», их упорядочивание по возрастанию ключа указанным алгоритмом сортировки за линейное время и вывод отсортированной последовательности.
- **Вариант сортировки:** Сортировка подсчетом
- **Вариант ключа:** Числа от 0 до 65535
- **Вариант значения:** Числа от 0 до $\times 2^{64}$ - 1

Метод решения

Моя сортировка реализована в функции TVector<TItem> CountSort(TVector<TItem> & vec), которая принимает на вход вектор и возвращает его отсортированный вариант. Общий алгоритм поразрядной сортировки основан на том, что мы подсчитываем сколько раз в массиве встречается каждое значение и заполняем массив подсчитанными элементами в соответствующих количествах. Счётчики для всего диапазона чисел создаются заранее (изначально равны нулю).

Для счётчиков я инциализировал массив для каждого из значений от 0 до максимума среди поданных на вход ключей. Прошелся по элементам массива vec и сопоставил каждому элементу vec индекс массива countArray. Вычислите префикс сумму для каждого индекса массива vec. Создайте массив output одного размера с vec. Прошелся по массиву vec от конца и обновите output[countArray[vec[i]] – 1] = vec[i]. Кроме того, обновил значение countArray[vec[i]] = countArray[vec[i]]-–.

Описание программы

Были написаны один класс и структура:

- class Tvector – собственная реализация вектора с произвольным типом
- struct TItem – структура для хранения пары ключ-значение.

Для выполнения задания я использовал:

- TVector<TItem> vec – вектор, где хранятся пары ключ-значения.

Изначально считываются все входные данные, затем сортируется основной вектор, в конце — вывод отсортированного вектора.

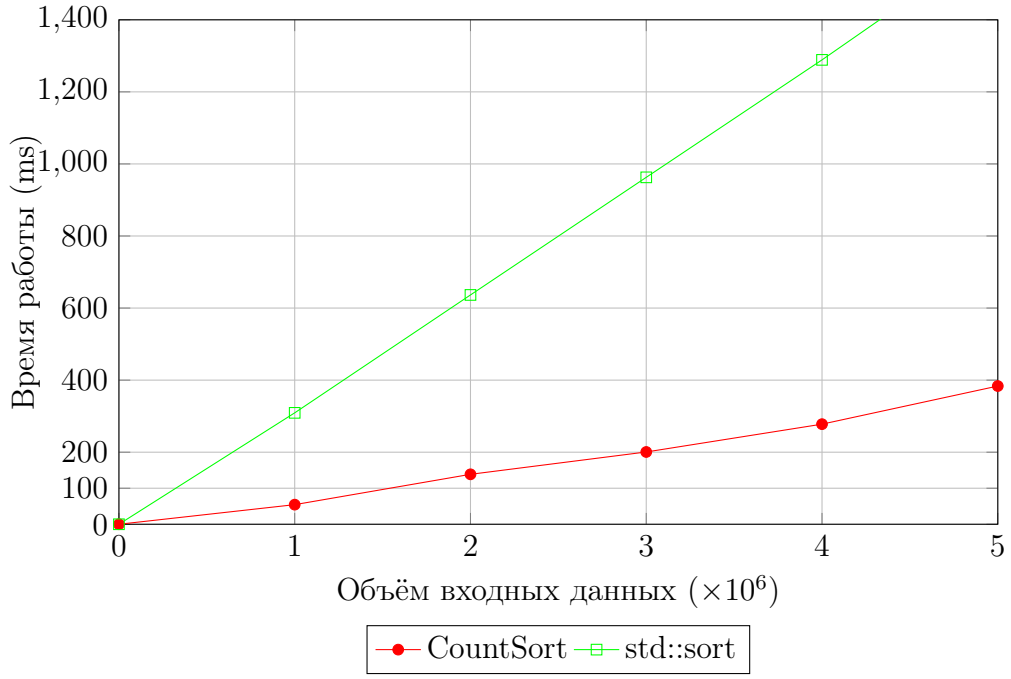
Дневник отладки

В первый раз был выбран неправильный компилятор в проверяющей системе. Затем проблемой стала синхронизация printf и cin, которая замедляла исполнение программы.

Тест производительности

Для проверки производительности алгоритма я использовала сравнение со стандартной сортировкой std::sort, сложность которой $O(n \log n)$. Сравнение производилось на входных данных больших размеров, не превышающих $5 * 10^6$.

Исходя из графика, представленного ниже, можно увидеть, что сложность моей сортировки подсчетом близка к линейной. А также на таком большом количестве данных сортировка подсчетом оказалась эффективнее стандартной сортировки std::sort.



Выводы

В результате данной лабораторной работы была написана и отлажена программа на языке C++, осуществляющая ввод пар «ключ-значение», их упорядочивание по возрастанию ключа алгоритмом поразрядной сортировки за линейное время и вывод отсортированной последовательности. Также были написаны аналоги стандартных контейнеров, такого как вектор.

Сложность моей поразрядной сортировки: $O(\mathrm{N}+\mathrm{K})$, где N – количество элементов, а K – максимальное значение ключа. В связи с тем, что максимальное значение ключа ограничено по условию, сложность получается линейной.

Данный алгоритм эффективен для быстрого поиска данных, диапазон ключей которых заранее известен, а также количество элементов сильно превышает количество элементов в массиве. То есть данные содержат много повторяющихся ключей.