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

Выполнил студент группы М8О-212Б-22 МАИ *Корнев Максим*.

Условие

Вариант: 6-3

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: телефонные номера, с кодами стран и городов в формате +<код страны> <код города> телефон.

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

Метод решения

Для решения задачи нужно создать два вектора, в первом векторе я буду хранить индекс в начальном массиве и телефон в виде unsigned long long числа (`struct Data`). Второй вектор нужен для хранения начальных данных в виде `string` и `unsigned long long` из основного вектора тоже через `struct`. Далее я передаю в поразрядную сортировку второй вектор. После сортировки вывожу строки из первого массива по номерам из второго массива. Тем самым я экономлю память, не таская длинные строки внутри сортировки.

Описание программы

Исходные данные хранятся в структуре `struct Key`. Дополнительный вектор для сортировки, состоящий из структур данных `struct Data`. Программа состоит из следующих функций:

1. `void countSort(std::vector<Data> Arr, unsigned long long exp)` - функция сортировки подсчётом по каждой цифре числа, начиная с конца.
2. `void radix_sort(std::vector<Data>& Arr, unsigned long long max_elem)` - функция поразрядной сортировки для строки, сортирует значения строк в массиве передавая индексы в функцию сортировки подсчётом.
3. `int main()`.

Дневник отладки

- Сначала я сталкивался с проблемой `RE` теста №1. Чтобы решить проблему, нужно было учесть, что печать вспомогательных std::cout дает неправильный ответ)
- Далее я сталкнулся с проблемой `RE` теста №13, а именно ML(memory limit). Чтобы решить проблему, нужно было понять, что таскание за собой больших данных не очень хорошая идея)

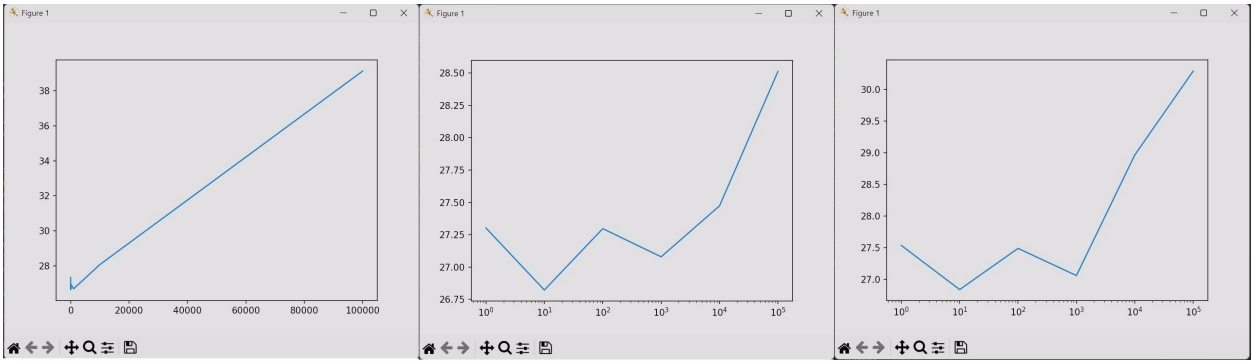

Рис. 1: Графики зависимости времени работы программы от количества введенных данных

Тест производительности

Сложность написанного алогоритма $O(n)$. Для построения графика (Рис. 1) использовались тесты от 1 до 100000 строк с данными. Из нескольких графиков видно, что количество входных данных почти не влияет на время работы программы.

Выводы

Поразрядная сортировка является ключевым методом для быстрой обработки массивов данных. Этот метод гарантирует, что время сортировки будет расти пропорционально количеству обрабатываемых элементов, что идеально подходит для управления огромными и постоянно увеличивающимися данными. Тем не менее, для более сложных структур данных, где требуется только сравнение элементов, более подходящими могут быть алгоритмы сортировки с временной сложностью $O(n \log n)$.