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

Выполнил студент группы М8О-212Б-22 МАИ *Мамонтов Егор*.

Условие

Вариант: 6-2

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: строки переменной длины (до 2048 символов).

Метод решения

Для решения задачи нужно создать два вектора, в первом векторе я буду хранить ключи и значения в виде `std::pair`. Второй вектор нужен для хранения ключа в виде `unsigned long long` и номер строки из основного вектора тоже через `std::pair`. 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 `std::pair<std::string, std::string>`. Дополнительный вектор для сортировки, состоящий из структур данных `std::pair<unsigned long long, int>`. Программа состоит из следующих функций:

1. `unsigned long long getMax(std::vector<std::pair<unsigned long long, int» array)` - функция возвращает максимальный элемент из вектора.
2. `void input(std::vector<std::pair<std::string, std::string» array)` - функция совершает ввод ключей и значений из консоли в основной вектор.
3. `unsigned long long to_ull(std::string s)` - функция переводит std::string в unsigned long long.
4. `void countSort(std::vector<std::pair<unsigned long long, int» array, unsigned long long exp)` - функция сортировки подсчётом по каждой цифре числа, начиная с конца.
5. `void radix_sort(std::vector<std::pair<unsigned long long, int»& arr)` - функция поразрядной сортировки для строки, сортирует значения строк в массиве передавая индексы в функцию сортировки подсчётом.
6. `int main()`.

Дневник отладки

- Сначала я сталкивался с проблемой `RE` теста №3. Чтобы решить проблему, нужно было учесть пустые введённые строки (очень хотелось бы, чтобы об этом указывалось в задании.)

- Далее я сталкивался с проблемами `WA` теста №7. Решением было не выкорёживаться, и нормально хранить все строки, как они задавались изначально, без попыток превратить номер `79851234567` в `+7-985-1234567`.

- Далее программа, путём эксперемента с вариантом моего товорища, получила вердикт `"OK"`.

- Далее я пытался эксперементировать с программой, пытался её ускорить и ещё сильнее оптимизировать. По итогам этих манипуляций получилось ускорить программу со 1.5 секунды до 1.1 секунды, без ущерба по памяти.

- Я разработал ещё один способ, как ускорить программу, путём работы с дополнительным вектором не как с `std::vector<std::pair<unsigned long long, int»` , а как с `std::vector<unsigned long long>`, записывая номер строки в конец `unsigned long long`. Но тогда программа ограничится максимумом в 999.999 элементов.

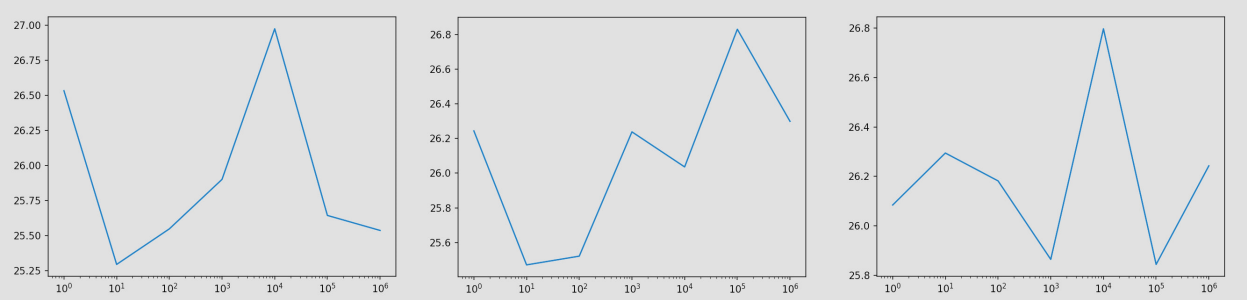

Рис. 1: Графики зависимости времени работы программы от количества введенных данных

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 $O(n \log n)$.