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

Выполнил студент группы 08-207 МАИ *Чекменев Вячеслав Алексеевич.*

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

Метод решения

примерное описание алгоритма:

1. заполнить массив префиксов так, чтобы число n, стоящее в индексе i встречается n раз в исходном массиве
2. для того же массива - на i-е место поставить такое число m_i, что $m_i = m_i + m_{i-1}$
3. затем создаем массив, который будет хранить отсортированный исходный вектор
4. проходим справа налево по исходному массиву, берем число n и рассматриваем его как индекс для массива префиксов, получаем по нему значение, это и есть индекс нашего числа n в отсортированном массиве (только нужно сместиться влево)

Описание программы

Разделил программу на четыре файла:

- заголовочный файл TDataItem.hpp содержит описание структуры данных и подключение библиотек
- заголовочный файл TData.hpp содержит описание класса, который обеспечивает работу с вектором структур из TDataItem.hpp
- файл TData.cpp содержит описание методов соответствующего класса
- файл main.cpp программа-драйвер

Дневник отладки

1. не прохожу 1 тест - ML, решил сразу. Решение: скинул весь код в один файл, все нормализовалось

2. не прохожу 3 тест - RE, неделю не мог решить. Решение: нужно было проверить первую строку на пустоту. Написал рандом тесты на питоне, увидел, что возникает std::length exception, когда первая строка пуста. В итоге завел глоб переменную, есть заходил в цикл считывания строк, то ставил true, иначе оставалось false.

3. не прохожу 13 тест - TL, решил сразу. Решение: изменил тип uint64_t на uint_fast64_t, отключил синхронизацию с stdio.h, что-то из этого сработало, не узнавал, что именно.

Тест производительности

Создал три файла с данными, размеры и время выполнения:

1. 10^4 строк – 0,075 (сек)

2. 10^5 строк – 0,638 (сек)

3. 10^6 строк – 5,900 (сек)

Как видно, сложность можно оценить сверху линией

Недочёты

Вывод ведущих нулей можно было бы сделать более эффективным способом, а я сделал с помощью средств библиотеки iomanip. Хотелось бы все-таки отослать на чекер программа с make, но тратить на это время нет смысла, так как код сам по себе несложный.

Выводы

- Сортировка подсчетом используется для набора данных, где δ = maxNumber - minNumber соответствует размерам памяти, которую можно выделить для массива. Если же колебание будет чересчур большим, то придется тратить много памяти. В моем случае этот алгоритм подходит, можно сказать, идеально.

- Если говорить о сложности именно написания алгоритма, то это было очень просто, однако сложнее оказалось (как всегда) отловить экстремальные случаи, например, пустую первую строку или очень большое количество строк во входных данных.

- Могу еще упомянуть занятность данного рода алгоритмов (за линейное время). Очень интересно решать задачу не напрямую, как в базовых сортировках. Не скажу, что это было для меня открытием, так как мы проходили эти сортировки ещ на первом курсе, однако написать их самостоятельно было очень увлекательно.