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

Выполнил студент группы М8О-207Б-20 Мерц Савелий Павлович

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 1-3:

Сортировка подсчетом.

Тип ключа: числа от 0 до 65535.

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

Метод решения

Для удобного хранения и использования пар ключ-значение написал структуру Item. Считанные данные помещаю в исходный вектор in_arr. Подсчитываю количество вхождений каждого из ключей in_arr и сохраняю в векторе count, затем подсчитываю префиксные суммы. Формирую и вывожу отсортированный вектор out_arr.

Описание программы

Программа состоит из одного файла. В котором реализована структура и основная программа.

Дневник отладки

Самое трудное - ошибки вывода. Опять потерял вывод ненужной пустой строки в середине программы. Надо будет тщательнее отмечать временный код или другими способами проверять работоспособность программы.

Столкнулся с превышением памяти в одном из тестов. Поменяв некоторые переменные продвинулся дальше.

Ошибка компиляции заставила подумать и посчитать возможные значения переменных. Изменение типа данных вектора count помогло.

Последней стеной было превышение времени работы на последнем тесте. Я не хотел использовать мув семантику до последнего, перепробовал всевозможные “оптимизации”, некоторые из которых мне подсказывали одногруппники, но ничего не помогало. Решил переписать

всё заново с вынесением сортировки в отдельную функцию, использовал итераторы, быстрые инты и мув семантику. Тем самым усложнив код, что мне очень не нравилось, но это сработало. Структура программы подсказывала, что можно упростить и поискать, что конкретно помогло. Выяснилось, что помог мув в методе push_back у вектора. Я использовал до этого eplace_back, считая его быстрее мува или равным. Также выяснилось что использование обычных интов вместо быстрых приводит к неправильному ответу в одном из тестов. Данную проблему я не смог разрешить даже с коллективной поддержкой одногруппников. От всего остального смог отказаться.

Тест производительности

Число входных строк	Время в миллисекундах
1000	0.3
10000	2
100000	10

Недочеты

Можно всё сделать по ооп, разбить на файлы и функции. Мне показалось это излишним. Код свелся к нескольким циклам, в каждом из которых вложена одна строчка кода.

Сделать без мув семантики и быстрых интов. У других ребят без них программы проходили проверку, но их кода не видел и внятного объяснения почему у них работает а у меня нет я не получил.

Выводы

В очередной раз выяснил, что к лабораторной надо подходить с умом и расчетом. Мог бы многие ошибки предусмотреть и не совершать их. Сортировка подсчетом весьма компактная и полезная. Формат лабораторной приятен и весьма дружелюбен, возможно стоит уменьшить количество посылок или ввести санкции за излишнее использование. Считаю лучше использовать помощь коллектива чем большое количество посылок.