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

Выполнила студентка группы М80-407Б-17 МАИ *Алексюнина Юлия*.

Условие

1. Требуется разработать программу, осуществляющую ввод пар «ключ-значение», их упорядочивание по возрастанию ключа алгоритмом сортировки подсчётом и вывод отсортированной последовательности.

2. Вариант 3-3.

   Поразрядная сортировка.

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

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

Метод решения

1. Данные на вход программе подаются через стандартный поток ввода и, как следствие, весьма удобно считывать циклом **while** (пока значение может быть прочитано, продолжать цикл).

   ```
   while(std::cin >> el.Key >> el.Value)
   ```

2. По алгоритму сортировки необходимо знать максимальный ключ, он находится в цикле считывания данных.

3. Собственно сам алгоритм сортировки принимает на вход вектор **v**, пары "ключ-значение" которого необходимо отсортировать по порядку возрастания ключей, и максимальный ключ. Также в коде сортировки присутствуют 2 дополнительных массива: массив **b[c[(v[i].Key)]]** для хранения отсортированных выходных данных, массив для временной работы **c[0...PART_MASK]**, где **PART_MASK** - максимальный ключ.

Описание программы

На каждой непустой строке входного файла располагается пара «ключ-значение», поэтому создадим новую структуру **Elem**, в которой будем хранить ключ и значение.

А также мы не знаем количество входных данных, поэтому мы напишем динамический массив - вектор, в который будут помещаться структуры **Elem**.

Source.cpp	
Тип данных	Значение
struct Elem	Структура для хранения пары "ключ-значение"
template <typename T> class Vector	Вектор для хранения структур **Elem**
Функция	Значение
Vector()	Конструктор класса **Vector**
Vector(int size, const value_type& default_value = value_type())	Конструктор класса **Vector**
int Size() const	Размер вектора
friend void swap(Vector& lhs, Vector& rhs)	Обмен местами двух векторов
Vector& operator=(Vector other)	Перегрузка оператора присваивания для класса **Vector**
~ Vector()	Деструктор класса **Vector**
void PushBack(const value_type& value)	Добавить элемент в конец вектора
void CountingSortBit(Vector <Elem>& v, int size, int point)	Функция сортировки подсчетом
void RadixSort(Vector <Elem>& v)	Функция поразрядной сортировки
int main()	Главная функция, в которой происходит чтение данных, вызов функции сортировки и вывод.

Дневник отладки

При создании этой таблицы была использована таблица локального репозитория.

Дневник отладки		
Время	Коммит	Описание
2020/10/25 19:47:27	init	Начало работы, прочитаны главы из Кормена про алгоритмы сортировки, написаны шаблоны файлов и функций
2020/10/25 23:39:09	parsing	продумывание возможных идей парсинга входных данных так, чтобы можно было работать с неизвестным количеством пар "ключ-значение"
2020/10/26 03:10:18	сортировка	переписывание алгоритма сортировки по аналогии с алгоритмом, данным в Кормене.
2020/10/26 12:10:14	сортировка, ч2	дополнение алгоритма сортировки, организована побитовая сортировка подсчетом
2020/10/26 15:10:20	рабочая версия программы	программа работает в vs, но рантаймится на чекере:(
2020/10/26 18:30:49	optimization	поиск причин рантайма, оптимизация кода
2020/10/27 12:17:18	death	создание класса Вектор для объектов пар "ключ-значение прописывание основных полей и методов, оптимизация времени работы
2020/10/27 17:17:03	alive	рабочая версия программы, OK на чекере

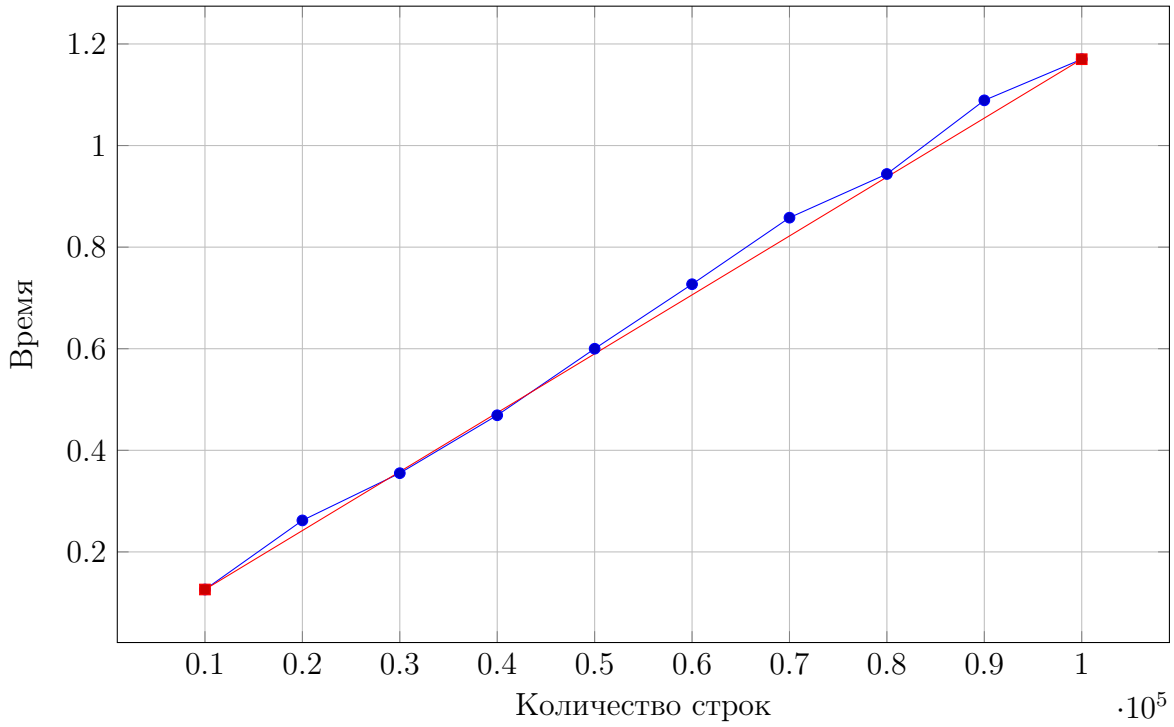
Тест производительности

Тесты создавались с помощью небольшой программы на языке C++:

```
#include <iostream>
#include <fstream>
#include <ctime>
#include <cstdlib>

int main(int argc, char *argv[]) {
        std::ofstream file(argv[1]);
        srand(time(0));
        size_t size = atoi(argv[2]);
        char value[65];
        for (register int i = 0; i < size; i++) {
                file << rand() % 65535 << " ";
                for (register int j = 0; j < 64; j++) {
                        value[j] = (char) (rand() % 26 + 97);
```

```
            }
            value[64] = '\0';
            file << value << "\n";
        }
        return 0;
}
```



Недочёты

При неиспользовании опций std::ios_base::sync_with_stdio(false) и std::cin.tie(NULL); программа не проходила тест по времени. После добавления этих опций время работы программы уменьшилось почти в 3 раза.

Выводы

Данный алгоритм сортировки можно применять для обработки списка людей по году рождения, где ключ – год рождения, а значение – фамилия, имя, отчество.

В целом написание программы было полезным, поскольку я получила опыт создания поразрядной сортировки, улучшила навыки отладки программы.