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

Выполнил студент группы М8О-208Б-20 *Зубко Дмитрий*.

Условие

Кратко описывается задача:

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

2. Вариант задания: 6-1. Поразрядная сортировка. Ключами служат телефонные номера, с кодами стран и городов в формате +<код страны>-<код города>-телефон. Значения – строки фиксированной длины 64 символа.

Метод решения

В начале программы я создаю вектор, в который считываю входные данные. Далее сортирую его поразрядной сортировкой. Поразрядная сортировка заключается в том, что мы сортируем каждый разряд числа справа налево. Сортировка разряда происходит с помощью сортировки подсчётом. Используется сортировка подсчётом, так как сортируемые числа от 0 до 9 и её сложность O(n). Таким образом, мы сортируем все входные данные за O(n*k), где n – количество входных данных, k – количество разрядов в ключе. В конце программы выводит отсортированные вектор, в формате ключ + табуляция + значение.

Описание программы

При выполнении программы изначально я создал несколько файлов. Однако, так как тестирующая система с make поддерживает только с++ 11, то пришлось переписать всё в один файл и отослать на с++20. На нём программа стала работать в 2.5 раза быстрее и я смог решить программу за отведенное время.

Для хранения данных я реализовал свой шаблонный динамический массив:

```
template<class T>
class MyVector {
public:
   MyVector();
   MyVector(int initial_size);
   ~MyVector();
   void PushBack(T element);
   void Print();
   T &operator[](const size_t idx);
```

```
    const T &operator[](const size_t idx) const;
    int size();
private:
    int capacity;
    int current_size;
    T *data;
};
```

Затем реализовал структуру для хранения ключа и ключа+значения:

```
class TelephoneNumber {
public:
    friend std::istream &operator>>(std::istream &in, TelephoneNumber &telephone_number);
    friend std::ostream &operator<<(std::ostream &out, const TelephoneNumber
&telephone_number);
    std::string GetCountryCode() const;
    std::string GetCityCode() const;
    std::string GetPhone() const;

private:
// +<country code>-<city code>-phone
    char country_code[4];
    char city_code[4];
    char phone[8];
};


class Object {
public:
    Object();
    Object(const std::string& initial_string);
    TelephoneNumber GetKey() const;
    friend std::ostream &operator<<(std::ostream &out, const Object &object);
    ~Object();

private:
    char value[65];
    TelephoneNumber key;
};
```

Сортировка заключается в том, что вызывается функция Sort, которая в свою очередь вызывает private сортировку для каждого разряда ключа. Private метод FindIdx определяет, какой индекс сортируется.

```
class RadixSort {
public:
    void Sort(MyVector<Object> &vector);
```

```
private:
   const int SORT_PHONE = 1;
   const int SORT_CITY_NUMBER = 2;
   const int SORT_COUNTRY_NUMBER = 3;
   int FindIdx(MyVector<Object> &vector, int max_size, int i, int r, int flag);
   void CountSort(MyVector<Object> &vector, size_t max_size, int r, int flag, MyVector<Object>
&sorted_array);
};
```

Дневник отладки

Изначально проблема заключалась в том, что не проходил 6 тест. Чтобы понять, в чем проблема мне понадобилось около 20 попыток отправить свое решение. Оказалась, что у меня на компьютере каждая строка, считанная из файла, заканчивалась как '\r\n', а в тестирующей системе – ‘\n’. Тем самым, когда вводилась пустая строка, у меня она обрабатывалась правильно, а в тестирующей системе нет.

Далее возникла проблема на последних тестах в том, что не хватало памяти. Я исправил эту проблему, заменив тип данных для хранения ключей. Вместо std::string я начал использовать просто char*.

После всего этого, программа не проходила по времени. Ошибка заключалась в том, что при сортировке подсчётом я каждый раз создавал новый вектор, размером n. Чтобы это исправить, я создал вектор один раз, перед сортировкой разрядов, и передавал его в сортировку подсчётом по ссылке.

Тест производительности

Пусть n – количество входных данных. Тогда:

1) При n = 10000 время работы программы составляет ~ 0.035 секунд.
2) При n = 100000 время работы программы составляет ~ 0.24 секунд.
3) При n = 1000000 время работы программы составляет ~ 2.2 секунд.
4) При n = 10000000 время работы программы составляет ~ 24 секунд.

Из этих тестов видно, что программы работает за линейную сложность.

Недочёты

Можно сказать, что программа не имеет недочетов, ведь они все были исправлены. Единственное, что можно отнести к ним – программа работает корректно только при правильных входных данных.

Выводы

Поразрядная сортировка может иметь большую область применения, так как она является устойчивой и её сложность O(d*n), где d – количество разрядов ключа, а n – количество входных данных. Устойчивость алгоритма означает, что если на вход программе подаётся одинаковые ключи с разными значениями, то в конце сортировки их порядок не поменяется.

Можно представить такую задачу: есть очередь на запись к врачу в разное время и нужно отсортировать её по возрастанию времени. Если несколько человек записаны на одно и тоже время, то идёт тот, кто записался первым. Тогда, чтобы построить очередь, можно представить эти данные так: ключ – время, значение – ФИО. После применения поразрядной сортировки, мы получим отсортированные данные по времени с сохранением порядка очереди в одно и тоже время.