Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Дискретный анализ»

Студент: А. О. Ларченко
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

Москва, 2024

Лабораторная работа №1

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

Вариант сортировки: Сортировка подсчётом.

Вариант ключа: `Числа от 0 до 65535.`

Вариант значения: `Строки фиксированной длины 64 символа, во входных данных могут встретиться строки меньшей длины, при этом строка дополняется до 64-х нулевыми символами, которые не выводятся на экран.`

1 Описание

Требуется написать реализацию алгоритма сортировки подсчётом.
Основная идея сортировки подсчетом заключается в том, чтобы для каждого входного элемента x нужно определить количество элементов, которые меньше x [1] .

2 Исходный код

На каждой непустой строке входного файла располагается пара «ключ-значение». Нам нужно отсортировать пару по ключам и следовательно для экономии памяти мы не будем таскать по функциям значения ключа. Поэтому разобъем пару на 2 класса: $TStr\_64$ в котором будем хранить значение значение, и $TObj$, в котором будем хранить ключ и индекс значения. При вводе элементов будем сохранять ключи в вектор key_array (вектор типа $TMy\_vector < TObj >$), а значения в вектор val_array (вектор типа $TMy\_vector < TStr\_64 >$).

```
class TStr_64{
    friend istream& operator >>(istream& is, TStr_64 &el);
    friend ostream& operator <<(ostream& os, TStr_64 &el);
    public:
        TStr_64() = default;
        int size(){
            return val_size;
        }
        string To_str(){
            return string(val, STR_SIZE);
        }
        bool Is_nonval(){
            return val_size==1;
        }
    private:
        char val[STR_SIZE];
        int val_size;
};
```

```
class TObj{
    public:
        unsigned int key;
        int64_t idx;
};
```

Считывание значений, вызов функции сортировки, а также вывод результата описаны в функции $main$.

```
int main(){
    ios::sync_with_stdio(false);
    cin.tie(0);
    TMy_vector<TStr_64> val_array;
    TMy_vector<TObj> key_array;
    TStr_64 val;
    unsigned int key;
    int64_t counter=0;
```

```
    while (cin>>key>>val){
        TObj tmp;
        if (!val.Is_nonval()){
            tmp.idx=counter;
            tmp.key=key;

            key_array.Push_back(tmp);

            val_array.Push_back(val);
            counter++;
        }
    }
    TObj q_array[counter];
    counting_sort(q_array, counter, key_array);
    if (counter>0){
        cout<<q_array[0].key<<val_array[q_array[0].idx];
    }
    for(int i=1;i< counter; ++i){
        cout<<'\n'<< q_array[i].key<<val_array[q_array[i].idx];
    }
}
```

Функция *counting_sort* сортирует исходный массив ключей. Принимает на вход: указатель на статический массив $*q\_array$, в который будет записаться отсортированые элементы; его длину q_array_size и ссылку на исходный вектор с данными(ключами) $\&in\_array$.

```
void counting_sort(TObj *q_array, int64_t q_array_size, TMy_vector<TObj> &in_array){
    int64_t counting_array[EL_RANGE];
    fill_n(counting_array, EL_RANGE, 0);
    for(int i=0; i< in_array.Size(); ++i){
        counting_array[in_array[i].key]++;
    }
    for(int i=1; i< EL_RANGE; ++i){
        counting_array[i]+=counting_array[i-1];
    }
    for (int i=in_array.Size()-1; i>=0;--i ){
        counting_array[in_array[i].key]--;
        q_array[counting_array[in_array[i].key]]=in_array[i];
    }
}
```

3 Консоль

```
arsenii@PC-Larcha14:~/Documents/VS_code_prog/DA/lab_1$ g++ main.cpp -o main
arsenii@PC-Larcha14:~/Documents/VS_code_prog/DA/lab_1$ cat >test1
52717 NZ9I6HDnH27mCubGlsgac7
46177 l6SfupqyVMxr0CvrzGN6I0He4RrLvumj
19705 tic6LNgKtfYNtZUkyqZtRyD1wE70pXUyciHboJoUozNS29RlCAM
63392 D0KwGkm1dOTg6eKCys5UWFSile20NGjSKOlfw2tPMyFIw
37165 hveFlsl6TivQv7AYhQGyGwebrUKJQEW5eOwAUZS
9198 SeAXGGY7tFKv8A967sxEgpGi7HoTwa7h6IdZ
31303 GufobnnkNVVGmW
15 P2giYzvNG1WlLtJrIEqGqVDNCRFIbJSIacJxBYyTzhfyL4LC5M0I
33051 2senTbKvbA8wuTHrYDYmnJGXeOh8dyYyD78v5UxUSZxL
48457 nu5Cg74Z8TRmlR6cFFKgGUV2eQ
16357 7yI
19504 fbN9zNtVrSPvohtxYvz
4955 7vCesjeBA8HMy5vBoGYb8JKpoNV1hLqXOZNCs0y4HyVYfmOnCnx3AEFNNOO
12358
51888 H6ejcLXDSRdP5oMMFwaKWC6Lb9ryVp6s8HVOqKEyIknJHEyGVT7Z3xKo
9811 fFgqpIhfq3VFM
26995 rwhJ0o9YYsQHwhE4nQl0SGm
8887 8EW25hCE3Lx8kemOpVDVOq
13873 ruUZsz1cXakZaFxih3SXqtk3wjVLOpWBt
34075 iY

arsenii@PC-Larcha14:~/Documents/VS_code_prog/DA/lab_1$ ./main <test1
15 P2giYzvNG1WlLtJrIEqGqVDNCRFIbJSIacJxBYyTzhfyL4LC5M0I
4955 7vCesjeBA8HMy5vBoGYb8JKpoNV1hLqXOZNCs0y4HyVYfmOnCnx3AEFNNOO
8887 8EW25hCE3Lx8kemOpVDVOq
9198 SeAXGGY7tFKv8A967sxEgpGi7HoTwa7h6IdZ
9811 fFgqpIhfq3VFM
13873 ruUZsz1cXakZaFxih3SXqtk3wjVLOpWBt
16357 7yI
19504 fbN9zNtVrSPvohtxYvz
19705 tic6LNgKtfYNtZUkyqZtRyD1wE70pXUyciHboJoUozNS29RlCAM
26995 rwhJ0o9YYsQHwhE4nQl0SGm
31303 GufobnnkNVVGmW
33051 2senTbKvbA8wuTHrYDYmnJGXeOh8dyYyD78v5UxUSZxL
34075 iY
37165 hveFlsl6TivQv7AYhQGyGwebrUKJQEW5eOwAUZS
46177 l6SfupqyVMxr0CvrzGN6I0He4RrLvumj
```

48457 nu5Cg74Z8TRmlR6cFFKgGUV2eQ
51888 H6ejcLXDSRdP5oMMFwaKWC6Lb9ryVp6s8HVOqKEyIknJHEyGVT7Z3xKo
52717 NZ9I6HDnH27mCubGlsgac7
63392 D0KwGkm1dOTg6eKCys5UWFSile20NGjSKOlfw2tPMyFIw

4 Тест производительности

Тест производительности представляет из себя следующее: поразрядная сортировка сравнивается со стабильной сортировкой из STL. Сравним алгоритмы 3 тестами(по 10^4, 10^5 и 10^6 элементов)

```
arsenii@PC-Larcha14:~/Documents/VS_code_prog/DA/lab_1$ g++ benchmark.cpp -o
benchmark
arsenii@PC-Larcha14:~/Documents/VS_code_prog/DA/lab_1$ ./benchmark <prod_tests/00.txt
Array size = 985
Counting sort time: 0.000919219 s
STL-sort time:      0.000121407 s
arsenii@PC-Larcha14:~/Documents/VS_code_prog/DA/lab_1$ ./benchmark <prod_tests/03.txt
bash: prod_tests/03.txt: No such file or directory
arsenii@PC-Larcha14:~/Documents/VS_code_prog/DA/lab_1$ ./benchmark <prod_tests/01.txt
Array size = 9857
Counting sort time: 0.002717170 s
STL-sort time:      0.006044357 s
arsenii@PC-Larcha14:~/Documents/VS_code_prog/DA/lab_1$ ./benchmark <prod_tests/02.txt
Array size = 98433
Counting sort time: 0.001397388 s
STL-sort time:      0.020190178 s
```

Можно заметить, что линейная сортировка выигрывает в 2 из 3 тестов(на 10^5 и 10^6 элементов). Из чего можно сделать вывод, что на большем количестве данных сортировка подсчетом работает быстрее STL. Это связано с тем, что сложность сортировки подсчетом - $\emptyset(N + K)$, где N - количество элементов в массиве, а K - диапазон, разность максимального и минимального элементов массива. Учитавая, что по условию, значение K у нас постоянное и равное 65536 мы находим обяснение, почему сортировка подсчетом выигрывает у STL-сортировки со сложностью $O(N\text{logN})$ только на большом наборе данных($> 10^3$).

P.S. Значение $Array\_size$ отлично от степени 10-ки, т.к. счетчик сразу исключает пустые строки.

5 Выводы

В этой лабораторной работе я познакомился с сортировками за линейное время, которые достигают таких показателей благодаря тому, что не используют сравнения. Но как и у всего, что нас окружает, у линейных сортировок есть свои минусы и плюсы, они будут работать быстрее сортировок за $O(N\text{logN})$ либо на небольшом диапазоне, либо на большом диапазоне, но при большом количестве данных.

Данная лабораторная сложностей у меня не вызвала, я также реализовал сортировку подсчетом, поэтому счита. что с поставленной задачей справился успешно.

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))