Московский авиационный институт (национальный исследовательский университет)

Факультет компьютерных наук и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Дискретный анализ»

Студент: А. А. Каримов
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б-22

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: Поразрядная сортировка.

Вариант ключа: `MD5-суммы (32-разрядные шестнадцатиричные числа)`

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

1 Описание

Поразрядная сортировка подразумевает применение сортировки подсчетом по каждому разряду элемента последовательности.

Для оценки сложности поразрядной сортировки вспомним сложность сортировки подсчетом: $\Theta(n+k)$, где n - количество элементов последовательности, а k - максимум этой последовательности. Поскольку мы запускаем сортировку подсчетом для каждого знака ключа, то получаем, что оценка поразрядной сортировки равна $O(n * l)$, где n - количество элементов, а l - количество разрядов.

2 Исходный код

На каждой непустой строке входного файла располагается пара «ключ-значение», поэтому создадим новую структуру `TValue`, в которой будем хранить ключ в виде статического массива `char[]` и значение в виде числа типа `int64_t`.

```
#include <string>

const int array_digit_size = 16;
const int key_length = 32;

class TValue {
public:
    char key[key_length];
    uint64_t value;

    TValue() = default;
    ~TValue() = default;

    TValue(std::string& _key, uint64_t _value) {
        for(int i = 0; i < key_length; ++i)
            key[i] = _key[i];

        value = _value;
    }

    void set(std::string& _key, uint64_t _value) {

        for(int i = 0; i < key_length; ++i)
            key[i] = _key[i];

        value = _value;
    }
};
```

В файле **sort.cpp** находится реализация поразрядной сортировки.
Здесь в цикле происходит запуск сортировки подсчетом для каждого знака ключа. По окончанию сортировки подсчетом для i-го разряда происходит обмен буферами вспомогательного и результирующего векторов.

```
#include "sort.hpp"
#include <iostream>


namespace sort {

    const int array_digit_size = 16;
    const int key_length = 32;

```

```
10      void radix_sort( vector::Vector<TValue>& elems ) {
11
12          vector::Vector<TValue> tmpResult(elems.get_size() + 1, true);
13
14          for(int j = key_length - 1; j >= 0; --j) {
15
16              int tmp[16];
17              for(int k = 0; k < array_digit_size; ++k)
18                  tmp[k] = 0;
19
20
21              for(int i = 0; i < elems.get_size(); ++i) {
22                  if('0' <= elems[i].key[j] and elems[i].key[j] <= '9')
23                      ++tmp[elems[i].key[j] - '0'];
24                  else
25                      ++tmp[elems[i].key[j] - 'a' + 10];
26              }
27
28              for(int k = 1; k < 16; ++k) {
29                  tmp[k] += tmp[k - 1];
30              }
31
32              for(int i = elems.get_size() - 1; i >= 0; --i) {
33                  int key;
34                  if('0' <= elems[i].key[j] and elems[i].key[j] <= '9')
35                      key = elems[i].key[j] - '0';
36                  else
37                      key = elems[i].key[j] - 'a' + 10;
38
39                  int pos = tmp[key]--;
40                  tmpResult[pos-1] = elems[i];
41              }
42              vector::swap(elems, tmpResult);
43          }
44
45          tmpResult.~Vector();
46      }
47  }
```

3 Консоль

```
karseny99@karseny99:/mnt/study/DA/lab1$ make
g++ -std=c++20  -c sort.cpp
g++ -std=c++20  sort.o lab1.cpp -o lab1
karseny99@karseny99:/mnt/study/DA/lab1$ ./lab1 <tests/02.t
11b3a318b750add4482f67160fd1c7af        5506620685087310468
2904a441415ad22c0ccbc8bab780ff01        15631025563904442920
2d57ead15d45f3b95217fc26f64e624f        17967741058834373087
5fb9cc898b5dc6ca6478b0f61ca58579        15549767237243378484
7d685d1f631b298beb848e4845e7d70b        13093843595709548303
88f32327a698e79035ea91b25c9fe79f        1522945456683991812
9ebaa12c5c594e3c6cf597c586120fc3        1741633272260555417
c23fe20a6b98698e44b378628cb02624        14690737309650550198
f2ad2b0ee93e3e0f8c94dac8b8181bf1        10823747237399279286
f7b994993cd43e8a1e8bc567008a5c34        3420891951338888443
```

4 Тест производительности

Производительность оценивается так: на один и тех же тестовых данных запускается поразрядная сортировка и встроенная сортировка в библиотеку C++. Тесты на 10^4, 10^5 и 10^6 элементов.

```
karseny99@karseny99:/mnt/study/DA/lab1$ ./benchmark <tests/05.t
Count of lines is 10000
Counting sort time: 4977us
STL stable sort time: 391us
karseny99@karseny99:/mnt/study/DA/lab1$ ./benchmark <tests/06.t
Count of lines is 100000
Counting sort time: 52848us
STL stable sort time: 4873us
karseny99@karseny99:/mnt/study/DA/lab1$ ./benchmark <tests/07.t
Count of lines is 1000000
Counting sort time: 542384us
STL stable sort time: 77753us
```

Как видно,на всех тестах STL-сортировка выигрывает. Стабильная сортировка из STL имеет сложность Θ(n*log_n),но,хотя поразрадная сортировка имеет линейную сложность,она все равно уступает. Так происходит из-за того,что константа в работе поразрядной сортировки довольна большая в сравнении с std::stable_sort(). Тем не менее,начиная с некоторого размера данных поразрядная сортировка выйдет победителем.

5 Выводы

В ходе выполнения лабораторной работы я смог разобраться в работе алгоритмов поразрядной сортировки и сортировки подсчетом. Также мне понадобилось реализовать свой вектор или динамический массив. Здесь возникла основные трудности в работе, потому что мое исходное решение задачи не проходило тестирующую систему из-за превышения ограничений по памяти. Это было вызвано тем, что емкость буфера увеличивалась вдвое при достижении его занятости до некоторого значения.

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.))

[2] *Цифровая сортировка*
URL: `https://neerc.ifmo.ru/wiki/index.php?title=Цифровая_Сортировка`