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: К. М. Воронов
Преподаватель: А. А. Кухтичев
Группа: М8О-207Б-19

Дата:
Оценка:
Подпись:

Москва, 2020

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: `Даты в формате DD.MM.YYYY, например 1.1.1, 1.9.2009, 01.09.2009, 31.12.2009.`

Вариант значения: `Строки переменной длины (до 2048 символов).`

1 Описание

Требуется написать реализацию алгоритма поразрядной сортировки с использованием стабильной сортировки подсчётом.

Основная идея поразрядной сортировки заключается в том, что мы разбиваем данные по разрядам и сортируем их по отдельности. Для сортировки разрядов воспользуемся стабильной сортировкой подсчётом, суть которой заключается в создании двух дополнительных массивов: C, в котором мы будем хранить, сколько элементов C[i] меньших или равных i в исходном массиве, и v2, в котором будет хранится результат. После заполнения массива C, проходим справа налево по исходному массиву и записываем в массив v2 с индексом C[i]-1 элемент v[i].

2 Исходный код

На каждой непустой строке входного файла располагается пара «ключ-значение», поэтому создадим новую структуру $TValue$, в которой будем хранить ключи и строку. Считывая строку, мы будем парсить её, вытаскивая оттуда ключи. Далее посортируем вектор отельно по дням, месяцам и годам и выведем результат. lab1.cpp

```
#include <iostream>
#include <string>
#include "vector.hpp"

struct TValue {
  int Day;
  int Month;
  int Year;
  std::string Data;

  TValue() : Day(1), Month(1), Year(1) {};
};

  void Sort(TVector<struct TValue> &v) {

  TVector<int> c(10000);
  c.Nul();
  TVector<int> c2(32);
  c2.Nul();
  TVector<int> c3(32);
  c3.Nul();
  TVector<struct TValue> v2(v.Capacity);

  for(int i = 0; i < v.Capacity; ++i) {
    c3.Data[(v.Data[i]).Day] += 1;
  }

  for(int i = 1; i < c3.Capacity; ++i) {
    c3.Data[i] += c3.Data[i - 1];
  }

  for(int i = v.Capacity - 1; i >= 0; --i) {
    (v2.Data[c3.Data[(v.Data[i]).Day] - 1]) = v.Data[i];
    c3.Data[(v.Data[i]).Day] -= 1;
  }

  for(int i = 0; i < v.Capacity; ++i) {
    v.Data[i] = v2.Data[i];
  }

  for(int i = 0; i < v.Capacity; ++i) {
    c2.Data[(v.Data[i]).Month] += 1;
```

```
  }

  for(int i = 1; i < c2.Capacity; ++i) {
    c2.Data[i] += c2.Data[i - 1];
  }

  for(int i = v.Capacity - 1; i >= 0; --i) {
    (v2.Data[c2.Data[(v.Data[i]).Month]-1]) = v.Data[i];
    c2.Data[(v.Data[i]).Month] -= 1;
  }

  for(int i = 0; i < v.Capacity; ++i) {
    v.Data[i] = v2.Data[i];
  }

  for(int i = 0; i < v.Capacity; ++i) {
    c.Data[(v.Data[i]).Year] += 1;
  }

  for(int i = 1; i < c.Capacity; ++i) {
    c.Data[i] += c.Data[i - 1];
  }

  for(int i = v.Capacity - 1; i >= 0; --i) {
    (v2.Data[c.Data[(v.Data[i]).Year] - 1]) = v.Data[i];
    c.Data[(v.Data[i]).Year] -= 1;
  }

  for(int i = 0; i < v.Capacity; ++i) {
    v.Data[i] = v2.Data[i];
  }
}

void Add(TVector<struct TValue> &v, int day, int month, int year, std::string d) {
    v.Push();
    (v.Data[v.Capacity - 1]).Day = day;
    (v.Data[v.Capacity - 1]).Month = month;
    (v.Data[v.Capacity - 1]).Year = year;
    (v.Data[v.Capacity - 1]).Data = d;
}

int main() {
    TVector<struct TValue> v;
    int day = 0;
    int month = 0;
    int year = 0;
    std::string d,d2;
    int u = 0;
    std::ios::sync_with_stdio(false);
```

```
  std::cin.tie(0);
  std::cout.tie(0);
    while(std::cin >> d2 >> d) {
      for(int i = 0; i < d2.size(); ++i) {

        if(d2[i] == '.') {
          u += 1;
          continue;
        }

        if(u == 0) {
          day = day * 10 + d2[i] - '0';
        }

        if(u == 1) {
          month = month * 10 + d2[i] - '0';
        }

        if(u == 2) {
          year = year * 10 + d2[i] - '0';
        }

      }
      d2 = d2 + " " + d;
      Add(v,day,month,year,d2);
      day = 0;
      month = 0;
      year = 0;
      u = 0;
  }

  Sort(v);
  for(int i = 0; i < v.Capacity; ++i) {
    std::cout << (v.Data[i]).Data << " " << std::endl;
  }
}
```

vector.hpp

```
#ifndef VECTOR_HPP
#define VECTOR_HPP
#include<iostream>

template <class T>
class TVector {
   private:
      T *Data2;
    public:
      int Capacity;
      int Max;
```

```
12      T *Data;
13
14    TVector(int n) {
15      Capacity = n;
16      Max = n * 2;
17      Data = new T[Max];
18      Data2 = 0;
19    }
20    TVector() {
21    Capacity = 0;
22      Max = 0;
23      Data = 0;
24      Data2 = 0;
25    }
26
27    ~TVector() {
28      delete[] Data;
29      Data = 0;
30   }
31
32  void Push() {
33
34   if(Data == 0) {
35       Data = new T[1];
36       Max = 1;
37   }
38
39   if(Capacity == Max) {
40        Data2 = new T[Max * 2];
41        Max = Max * 2;
42       for(int i = 0; i < Capacity; ++i) {
43          Data2[i] = Data[i];
44       }
45        delete[] Data;
46        Data = Data2;
47        Data2=0;
48   }
49
50    Capacity += 1;
51 }
52    void Nul() {
53      for(int i = 0; i < Capacity; ++i) {
54        Data[i] = T();
55      }
56    }
57 };
58
59 #endif
```

3 Консоль

```
kirill@kirill-VirtualBox:~/Рабочий стол/DA$ cat test
01.10.2001 avdkkljhfdjkslhhfjkdshlsklhfuewkfhkldjs
2.3.1987 dfshldgckkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk
15.08.2000 gjgj
23.25.2000 pkfjldjlkjkldfjgkl;jdfklgj
1.2.3 fdgkljl,jk6+56+dfldg
3.6.1988 kcjgjdfhkgilddkffhgdfjdg
kirill@kirill-VirtualBox:~/Рабочий стол/DA$ cat test | ./solution
1.2.3 fdgkljl,jk6+56+dfldg
2.3.1987 dfshldgckkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk
3.6.1988 kcjgjdfhkgilddkffhgdfjdg
15.08.2000 gjgj
23.25.2000 pkfjldjlkjkldfjgkl;jdfklgj
01.10.2001 avdkkljhfdjkslhhfjkdshlsklhfuewkfhkldjs
```

4 Тест производительности

Сравним работу поразрядной сортировки и std::stable_sort. Проверим на 10^6, 10^5 и 10^4 входных данных.

```
kirill@kirill-VirtualBox:~/Рабочий стол/DA$ ./a.out | ./solution
Количество элементов 1000000
Время поразрядной сортировки: 1621795us
Время сортировки stl: 3209358us
kirill@kirill-VirtualBox:~/Рабочий стол/DA$ ./a.out | ./solution
Количество элементов 100000
Время поразрядной сортировки: 65778us
Время сортировки stl: 131133us
kirill@kirill-VirtualBox:~/Рабочий стол/DA$ ./a.out | ./solution
Количество элементов 10000
Время поразрядной сортировки: 8465us
Время сортировки stl: 11255us
```

Как видно, поразрядная сортировка со сложностью O(n) работает быстрее, чем std::stable_sort со сложностью O(nlogn).

5 Выводы

Выполнив первую лабораторную работу по курсу «Дискретный анализ», я научился писать класс вектора, используя утилиты new и delete, писать поразрядную сортировку, с использованием стабильной сортировки подсчётом, и писать генератор тестов. Также я узнал, как создавать шаблоны в классе и объединять их с программой.

Список литературы

[1] *Сортировка подсчётом — Википедия.*
URL: `http://ru.wikipedia.org/wiki/Сортировка_подсчётом` (дата обращения: 01.10.2020).

[2] *Генератор случайных чисел.*
URL: `http://cppstudio.com/post/339/`(дата обращения: 05.10.2020).