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: И. А. Мариничев
Преподаватель: Н. С. Капралов
Группа: М8О-208Б

Дата: 15.10.20
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: 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

Программе подаются входные данные через стандартный поток ввода и, как следствие, весьма удобно считывать их циклом **while** (пока значение может быть прочитано, продолжать цикл). Предусматривается работа с неизвестным количеством данных. Для алгоритма сортировки подсчетом необходимо знать верхнюю границу. Мы находим ее в цикле считывания данных. Далее требуется написать реализацию алгоритма сортировки подсчётом.
Основная идея сортировки подсчетом заключается в том, чтобы для каждого входного элемента x определить количество элементов, которые меньше x [1].
Собственно сам алгоритм сортировки принимает на вход вектор **unsortedVector**, пары «ключ-значение» которого необходимо отсортировать по порядку возрастания ключей, и максимальный ключ **max**. Также в коде сортировки присутствуют 2 дополнительных массива:
- массив для временной работы **count[0...max]**,
- массив **result[0...unsortedVector.Size()]** для хранения отсортированных выходных данных.

2 Исходный код

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

Кроме того, так как мы не знаем количество входных данных, напишем динамический массив - вектор **TVector**, в который будем помещать наши пары **TPair**.

Листинг (реализация методов опущена)

```
#include <iostream>
#include <cassert>
#include <algorithm>

const int SIZE = 65;

struct TPair {
  unsigned short key;
  char value[SIZE];

  TPair();
  TPair(int i, char *str);
  void Fill(char ch = '\0');
};

template <typename T>
class TVector {
public:
  using TValueType = T;
  using TIterator = TValueType *;

  TVector();
  TVector(unsigned int size);
  unsigned int Size() const;
  bool Empty() const;
  TIterator Begin() const;
  TIterator End() const;
  template <typename U>
  friend void Swap(TVector<U> &first, TVector<U> &second);
  TVector &operator=(TVector other);
  ~TVector();
  TValueType &operator[](int index) const;
  void PushBack(TValueType &value);
  template <typename U>
  friend TVector<U> CountingSort(const TVector<U> &unsortedVector, unsigned int max);

private:
```

```
38    int storageSize;
39    int alreadyUsed;
40    TValueType *storage;
41  };
42
43  template <typename U>
44  TVector<U> CountingSort(const TVector<U> &unsortedVector, unsigned int max) {
45    TVector<unsigned int> count{max + 1};
46    for (register unsigned int i = 0; i <= max; i++) {
47      count[i] = 0;
48    }
49    for (register unsigned int i = 0; i < unsortedVector.Size(); ++i) {
50      ++count[unsortedVector[i].key];
51    }
52    for (register unsigned int i = 1; i <= max; i++) {
53      count[i] += count[i-1];
54    }
55    TVector<U> result{unsortedVector.Size()};
56    for (register int i = unsortedVector.Size() - 1; i >= 0; i--) {
57      result[--count[unsortedVector[i].key]] = unsortedVector[i];
58    }
59
60    return result;
61  }
62
63  int main() {
64    std::ios::sync_with_stdio(false);
65    std::cin.tie(nullptr);
66
67    TVector<TPair> elems;
68    TPair temp;
69    unsigned int maxKey = 0;
70    while(std::cin >> temp.key >> temp.value) {
71      elems.PushBack(temp);
72      if (temp.key > maxKey) {
73        maxKey = temp.key;
74      }
75      temp.Fill();
76    }
77    TVector<TPair> sortedVector = CountingSort(elems, maxKey);
78    for (size_t i = 0; i < sortedVector.Size(); ++i) {
79      std::cout << sortedVector[i].key << '\t' << sortedVector[i].value << std::endl;
80    }
81    return 0;
82  }
```

Таблица методов и функций

1-1.cpp	
Тип данных	Значение
struct TPair	Структура для хранения пары "ключ-значение"
template <typename T> class TVector	Вектор для хранения структур **TPair**
Функция	Значение
TPair()	Конструктор по умолчанию структуры **TPair**
TPair(int i, char *str)	Конструктор структуры **TPair**
void Fill(char ch = '\0')	Заполнение массива **value[]**
TVector()	Конструктор по умолчанию класса **TVector**
TVector(unsigned int size)	Конструктор класса **TVector**
unsigned int Size() const	Размер вектора
bool Empty() const	Проверка на пустоту вектора
TIterator Begin() const	Доступ к началу вектора
TIterator End() const	Доступ к концу вектора
template<typename U> friend void Swap(TVector<U> &first, TVector<U> &second)	Обмен местами двух векторов
TVector &operator=(TVector other)	Перегрузка оператора присваивания для класса **TVector**
~ TVector()	Деструктор класса **TVector**
TValueType &operator[](int index) const	Перегрузка оператора [] для класса **TVector**
void PushBack(TValueType &value)	Добавить элемент в конец вектора
template <typename U> friend TVector<U> CountingSort(const TVector<U> &unsortedVector, unsigned int max)	Функция сортировки подсчётом
int main()	Главная функция, в которой происходит чтение данных, вызов функции сортировки и вывод.

3 Консоль

```
ivan@Laptop-IM:/mnt/c/Users/Иван/projects/da_labs/da_lab1$ cat Makefile
CC=g++
FLAGS=-std=c++14 -pedantic -Wall -Wextra -Wno-unused-variable
OUTPUT=lab1

all: 1-1.cpp
$(CC) $(FLAGS) 1-1.cpp -o $(OUTPUT)
clean:
rm *.o $(OUTPUT)
ivan@Laptop-IM:/mnt/c/Users/Иван/projects/da_labs/da_lab1$ make
g++ -std=c++14 -pedantic -Wall -Wextra -Wno-unused-variable 1-1.cpp -o lab1
ivan@Laptop-IM:/mnt/c/Users/Иван/projects/da_labs/da_lab1$ cat test1
0 n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3naatt
65535 n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3naat
0 n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3naa
65535 n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3na
ivan@Laptop-IM:/mnt/c/Users/Иван/projects/da_labs/da_lab1$ ./lab1 <test1
0       n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3naatt
0       n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3naa
65535   n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3naat
65535   n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3na
```

4 Тест производительности

Тест производительности представляет из себя следующее: моя реализация сортировки подсчетом сравнивается с **std::stable_sort()**. Тест состоит из 1000 строк. Время выводится микросекундах. Для замерки времени использовалась бибилиотека **chrono**.

Тесты создавались с помощью программы на языке Python:

```python
import sys
import random
import string

TEST_COUNT = 8

def get_random_string():
    length = random.randint(0, 64)
    random_list = [ random.choice(string.ascii_letters) for _ in range(length) ]
    return "".join(random_list)

def main():
    if len(sys.argv) != 2:
        print(f"Usage: {sys.argv[0]} <test directory>")
        sys.exit(1)

    test_dir = sys.argv[1]

    lines = [0]
    lines.extend([ 10 ** i for i in range(TEST_COUNT) ])

    for enum, test_count in enumerate(range(1, TEST_COUNT+1)):
        test = []
        answer = []

        line_count = lines[enum]
        for _ in range(line_count):
            key = random.randint(0, 65535)
            value = get_random_string()
            test.append((key, value))

        test_name = "{}/{:02d}".format(test_dir, test_count)
        with open(f'{test_name}.t', 'w') as ftest:
            for key, value in test:
                ftest.write(f'{key} {value}\n')

        answer = sorted(test, key=lambda x: x[0])
        with open(f'{test_name}.a', 'w') as ftest:
            for key, value in answer:
                ftest.write(f'{key} {value}\n')
main()
```

```
ivan@Laptop-IM:/mnt/c/Users/Иван/projects/da_labs/da_lab1$ python3 generator.py
tests
ivan@Laptop-IM:/mnt/c/Users/Иван/projects/da_labs/da_lab1$ ls
01.t  03.t  05.t  07.t  1-1.cpp  benchmark_stl.cpp  tests       vector.hpp
02.t  04.t  06.t  08.t  Makefile  benchmark.cpp  generator.py       vector.cpp
ivan@Laptop-IM:/mnt/c/Users/Иван/projects/da_labs/da_lab1$ g++ benchmark_stl.cpp
-o benchstl
ivan@Laptop-IM:/mnt/c/Users/Иван/projects/da_labs/da_lab1$ g++ benchmark.cpp
-o bench
ivan@Laptop-IM:/mnt/c/Users/Иван/projects/da_labs/da_lab1$ ./bench <05.t
Count of lines is 1000
Counting sort time: 585us
ivan@Laptop-IM:/mnt/c/Users/Иван/projects/da_labs/da_lab1$ ./benchstl <05.t
Counting sort time: 585us
ivan@Laptop-IM:/mnt/c/Users/Иван/projects/da_labs/da_lab1$ ./benchstl <05.t
Counting sort time: 585us
ivan@Laptop-IM:/mnt/c/Users/Иван/projects/da_labs/da_lab1$ ./benchstl <05.t
Count of lines is 1000
STL stable sort time: 4us
```

Как видно, что **std::stable_sort()** выиграл у моей реализации, так как в моей реализации происходит создание вектора и операция копирования из одного вектора в другой, которые требует отдельного времени.

5 Выводы

Выполнив первую лабораторную работу по курсу «Дискретный анализ», я научился применять теоретические знания о сортировках за линейное время, а именно о сортировке подсчетом, на практике. Смог написать свою реализацию вектора, которая может понадобиться при выполнении дальнейших лабораторных работ. Получил опыт написания генератора тестов, а также написал свой первый benchmark, замеряющий время работы определенной реализации алгоритма. А также улучшил свои навыки в отлаживании программы при работе с чекером.

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] *Символы и строки в C++*
URL: `http://cppstudio.com/post/437/` (дата обращения: 09.10.2020).

[3] *Templates in C++*
URL: `https://www.geeksforgeeks.org/templates-cpp/` (дата обращения: 10.10.2020).

[4] *How to implement our own Vector Class in C++?*
URL: `https://www.geeksforgeeks.org/how-to-implement-our-own-vector-class-in-c/` (дата обращения: 12.10.2020).

[5] *Chrono in C++*
URL: `https://www.geeksforgeeks.org/chrono-in-c/` (дата обращения: 14.10.2020).