МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Алгоритмы и структуры данных»
Тема: Сортировки

Студентка гр. 9304 _____ Рослова Л.С

Преподаватель _____ Филатов А.Ю

Санкт-Петербург
2020

Цель работы.

Изучить методы сортировок в массиве. Применить усовершенствованные методы сортировок на практике.

Задание.

8) Быстрая сортировка, рекурсивная реализация. Во время сортировки массив должен быть в состоянии : элементы < x, не отсортированные элементы, элементы >= x.

Выполнение работы.

Программа принимает на вход строку со значениями, которая подается функции *convert* для преобразования в вектор целочисленных значений, данные не относящиеся к цифрам будут проигнорированы. Вектор значений *arrDigit* подается функции *customQsort* для дальнейшей сортировки, если вектор пуст — программа завершает свое выполнение. Сортировка осуществляется с помощью рекурсивной лямбды, которая в качестве аргументов принимает крайние индексы сортируемого массива. Выход из рекурсии осуществляется при пересечении индексов.

Разработанный программный код см. в приложении А.

Формат входных и выходных данных.

На вход программе подается строка с целочисленными значениями.

Программа должна рекурсивно отсортировать массив чисел придерживаясь правила – < x, не отсортированные элементы, элементы >= x.

Тестирование.

Для проведения тестирования был написан bash-скрипт ./script .Скрипт запускает программу где в качестве входных аргументов служат заранее подготовленные файлы, расположенные в папке ./Tests

```
Test 3:
Start = 7 7 1 7 4 2 8
7 7 1 7 4 2 8   ->      Swap 2[5] and 7[0]      ->      2 7 1 7 4 7 8
2 7 1 7 4 7 8   ->      Swap 1[2] and 2[0]      ->      1 7 2 7 4 7 8
1 7 2 7 4 7 8   ->      Swap 2[2] and 7[1]      ->      1 2 7 7 4 7 8
1 2 7 7 4 7 8   ->      Swap 4[4] and 7[2]      ->      1 2 4 7 7 7 8
1 2 4 7 7 7 8   ->      Swap 7[4] and 7[3]      ->      1 2 4 7 7 7 8
1 2 4 7 7 7 8   ->      Swap 7[5] and 7[3]      ->      1 2 4 7 7 7 8
1 2 4 7 7 7 8   ->      Swap 7[4] and 7[3]      ->      1 2 4 7 7 7 8
1 2 4 7 7 7 8   ->      Swap 7[5] and 7[4]      ->      1 2 4 7 7 7 8
1 2 4 7 7 7 8   ->      Swap 7[4] and 7[3]      ->      1 2 4 7 7 7 8
1 2 4 7 7 7 8   ->      Swap 7[5] and 7[4]      ->      1 2 4 7 7 7 8
Result: 1 2 4 7 7 7 8
```

Рисунок 1 — Часть вывода скрипта.

Выводы.

Был изучен принцип быстрой сортировки. Создана программа реализующая быструю сортировку на языке C++, сложность в среднем состовляет O(n log n). Основной проблемой является возможность переполнения стека из-за рекурсивного обхода.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <vector>

#include "customQsort.h"


int main(){

std::string strValue {};
getline(std::cin, strValue);

std::vector<int> arrDigit = convert(strValue);

if(!arrDigit.empty()){
customQsort(arrDigit);
}

std::cout << "Result: ";
for(const auto &str : arrDigit){
std::cout << str << ' ';
}
std::cout << std::endl;

return 0;
}
```

Название файла: csutomQsort.h

```
#pragma once

#include <vector>
#include <string>

void customQsort(std::vector<int> &arrDigit);
std::vector<int> convert(std::string &strValue);
```

Название файла: csutomQsort.cpp

```
#include "customQsort.h"
#include <iostream>
```

```
std::vector<int> convert(std::string &strValue){
    std::vector<int> arrDigit {};
    strValue.c_str();
    bool flag1 = 0;
    bool flag2 = 0;
    for(size_t i = 0; strValue[i] != '\0'; i++){
        if(isdigit(strValue[i])){
            if(flag1){
                continue;
            }else{
                arrDigit.emplace_back(atoi(&strValue[i]));
                flag1 = 1;
            }
        }else if(strValue[i] == '-'){
            flag2 = 1;
        }else{
            flag1 = 0;
            if(flag2){
                arrDigit[arrDigit.size() - 1] -= (arrDigit.back() * 2);
                flag2 = !flag2;
            }
        }
    }
    if(!arrDigit.size()){
        arrDigit.clear();
    }

    return arrDigit;
}



void customQsort(std::vector<int> &arrDigit){

    size_t l = 0;                       // left index
    size_t r = arrDigit.size() - 1;     // right index

    auto PR = [&arrDigit](int l, int r, auto&& PR){

        int left = l;
        int right = r;
        int base = arrDigit[(l + r) / 2];

        if(l >= r){
            return;
        }

        while(left < right){

            while(arrDigit[left] < base){
                ++left;
            }
            while(arrDigit[right] > base){
                --right;
            }
            if(left <= right){
                if(left != right){
```

```
                    for(const auto &str : arrDigit){
                        std::cout << str << ' ';
                    }
                    std::swap(arrDigit[left], arrDigit[right]);
                     std::cout << "\t->\tSwap " << arrDigit[left] << '['
<< right << ']' << " and " << arrDigit[right] << '[' << left << ']' << "\
t->\t";
                    for(const auto &str : arrDigit){
                        std::cout << str << ' ';
                    }
                    std::cout << '\n';
                }
                ++left;
                --right;
            }

            PR(l, right, PR);
            PR(left, r, PR);
        }
    };

    PR(l, r, PR);

}
```

Название файла: script

```
#!/bin/bash
for n in {1..7}
do
arg=$(cat Tests/test$n.txt)
echo -e "\nTest $n:"
echo "Start = $arg"
./lab4 < Tests/test$n.txt
done
```