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

Студент гр. 9304 _____ Цаплин И.В.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург

2020

Цель работы.

Ознакомиться с алгоритмами сортировки. Реализовать один из алгоритмов сортировки на языке программирования C++.

Задание.

Вариант 6.

Реализовать алгоритм бинго-сортировки.

Описание алгоритма работы.

Программа принимает на вход строку, в которой записана последовательность чисел для сортировки. Затем строка с помощью функции checkString() проверяется на корректость. Если строка некорректна, выводится сообщение об этом, программа завершается. Если строка корректна, значения чисел записываются в вектор. Вектор копируется для дальнейшей проверки правильности сортировки. Затем вектор сортируется алгоритмом бинго-сортировки с помощью функции bingoSort(). Полученный вектор выводится в стандартный поток вывода. Затем копия вектора сортируется функцией std::sort. Полученный вектор сравнивается с результатом работы функции bingoSort().

Формат входных и выходных данных.

Программа принимает на вход строку, состоящую из целых чисел, разделенных любым количеством пробелов.

Программа выводит состояние массива на каждом шаге алгоритма и пояснения действий алгоритма.

Затем программа выводит на экран полученный после сортировки вектор и результат его сравнения с вектором, полученным после работы функции std::sort.

Описание основных структур данных и функций.

Функция print() - печатает вектор.

Функция bingoSort() - реализует алгоритм бинго-сортировки.

Функция принимает ссылку на вектор. При первом проходе функция находит максимальное значение. Индекс maxIndex указывает на последний

элемент. Если последний элемент является максимальным, индекс maxIndex уменьшается.

Затем в цикле while, пока значение maxIndex больше нуля, максимальные значения переносятся в конец вектора на каждом шаге алгоритма. Одновременно с этим ищется максимальное значение для следующего шага. В конце каждого шага значение maxIndex уменьшается до тех пор, пока maxIndex не указывает на максимальный элемент или равно нулю.

Функция checkString() - проверяет, что строка корректна.

Функция принимает ссылку на строку. Затем функция проверяет, что строка состоит из последовательностей цифр, перечисленных через пробел. Исключением является знак минуса, который стоит после пробела, и после которого стоит хотя бы одна цифра.

Разработанный код см. в приложении А.

Тестирование.

Программа проводит тестирование самостоятельно, сравнивая результат своей работы с результатом работы функции std::sort.

Для проведения тестирования был написан bash-script tests_script, который запускает программу с определенными входными данными и выводит результат работы в стандартный поток вывода.

Результаты тестирования см. в приложении Б.

Выводы.

Были изучены алгоритмы сортировки. Был реализован алгоритм бинго-сортировки на языке программирования C++.

Была разработана программа, преобразующая строку в вектор чисел и сортирующая полученный вектор с помощью алгоритма бинго-сортировки.

Сложность алгоритма бинго-сортировки составляет $O(n^2/2)$.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab4.cpp

```
#include <iostream>
#include <sstream>
#include <vector>
#include <string>
#include <algorithm>

template<typename T>
void print(std::vector<T>& vec) {
    for (int it : vec) {
        std::cout << it << ' ';
    }
}

template<typename T>
void bingoSort(std::vector<T>& data){
    int maxIndex = size(data)-1;
    T nextMax = data[maxIndex];
    for (int i = maxIndex - 1; i > -1; i--){
        if (data[i] > nextMax) {
            nextMax = data[i];
        }
    }

    while((maxIndex > 0) && (data[maxIndex] == nextMax)){
        maxIndex--;
    }

    T curMax;
    std::cout <<  "Original data: ";
    print(data);
    std::cout <<  "\n";
    std::cout << "First max: " << nextMax << "\n";
    int iterationCounter = 1;
    while(maxIndex > 0){
        curMax = nextMax;
        nextMax = data[maxIndex];
        std::cout << "Iteration: " << iterationCounter << "\
n";
        for(int i = maxIndex - 1; i > -1; i--){
            if (data[i] == curMax){
                std::swap(data[i], data[maxIndex]);
                std::cout << "Swapped " << data[i] << " and "
<< data[maxIndex] << ":\t";
                print(data);
                std::cout << "\n";
                maxIndex--;
```

```
                }else{
                    if (data[i] > nextMax){
                        nextMax = data[i];
                        std::cout << "Found new next max: " <<
data[i] << "\n";
                    }
                }
            }
            while((maxIndex > 0) && (data[maxIndex] == nextMax)){
                maxIndex--;
            }
        iterationCounter++;
    }
}

bool checkString(std::string& str){
    auto iterator = str.cbegin();
    while(iterator != str.cend()){
        if(*iterator == '-'){
            iterator++;
        }
        if(!isdigit(*iterator)){
            return false;
        }

        while(isdigit(*iterator)){
            iterator++;
        }

        if((*iterator != ' ') && (iterator != str.cend())){
            return false;
        }

        while(*iterator == ' '){
            iterator++;
        }

    }
    return true;
```

```
}

int main() {
    std::string inString{};
    std::getline(std::cin, inString);
    std::cout <<  "String: " << inString << "\n";
    if(!checkString(inString) || inString.empty()){
        std::cout << "String is not correct\n";
        return 0;
    }
    std::stringstream iss(inString);
    int number;
    std::vector<int> vec;
    while ( iss >> number )
        vec.push_back(number);
    std::vector<int> copyVec = vec;
    bingoSort(vec);
    std::cout << "\nResult: ";
    print(vec);
    std::cout << "\n";
    std::sort(copyVec.begin(), copyVec.end());
    if (std::equal(vec.cbegin(), vec.cend(),copyVec.cbegin())){
        std::cout << "Compare test with std::sort passed\n";
    }else{
        std::cout << "Compare test with std::sort failed\n";
    }
    return 0;
}
```

Название файла: tests_script

```
#! /bin/bash
printf "\nRunning tests...\n\n"
for n in {1..8}
do
    printf "Test$n:\n"
    ./lab4 < "./Tests/test$n.txt"
```

```
    printf "\n"
done
```

Название файла: Makefile

```
lab4: Source/lab4.cpp
	g++ -Wall -std=c++17 Source/lab4.cpp -o lab4

run_tests: lab4
	./tests_script
```

ПРИЛОЖЕНИЕ Б
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

№	Входные данные	Выходные данные	Комментарии
1	9 8 7 6 5 4 3 2 1 0	0 1 2 3 4 5 6 7 8 9	Числа в порядке убывания
2	5 5 5 5 5 5 5 5 5 5	5 5 5 5 5 5 5 5 5 5	Псоледовательность из одного числа
3	-1 -2 -3 -4 -5 -6 -7 -8 -9 -10	-10 -9 -8 -7 -6 -5 -4 -3 -2 -1	Отрицательные числа в порядке убывания
4	1 -2 3 -4 5 -6 7 -8 9 -10	-10 -8 -6 -4 -2 1 3 5 7 9	Последовательность, содержащая отрицательные и положительные числа
5	2398 -9428 24 29342 -2342 9230 -440 23 103 10	-9428 -2342 -440 10 23 24 103 2398 9230 29342	Случайная последовательность
6	94238 93 240 9234 0923042 402 934 094 023940 2349	93 94 240 402 934 2349 9234 23940 94238 923042	Разное количество пробелов
7	--23 322 342 321 123 312 245	String is not correct	Некорректная строка
8	1 2 3 4 5 6 7 8a9 0	String is not correct	Некорректная строка