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(ка) гр. 9381 _____ Шахин Н.С

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

Реализовать алгоритм пасьянской сортировки.

Задание.

Вариант 22.

Пасьянсная сортировка.

Описание работы алгоритма.

Этап 1. Разложение по стопкам.

Раскладываем числа в несколько стеков, таким образом, чтобы в каждом стеке числа представляли из себя упорядоченную последовательность. Первое число — начало первого стека. В этот стек перекладываем числа по очереди, до тех пор, пока очередное перекладываемое число меньше, чем верхнее в стеке. Обрабатывается не весь стек, а только верхнее число в нем, которое положили последним. Если текущее число больше, чем минимальное в стеке, значит, придётся создавать новый стек, текущее число открывает новый стек. Важна очерёдность стеков. Если их количество уже более одного, то очередное число помещается не в последний стек, а в самый левый стек, в который можно поместить. В итоге числа разложены в несколько стеков. В каждом стеке числа представляют из себя убывающую последовательность, вверху — наименьшее число.

Этап 2. Нижний ряд.

Сдвигаем доступные верхние числа и помещаем их в новый стек в порядке от самой правок к самой левой. Этот стек будет отсортирован по возрастанию. Далее используем только числа, которые находятся в этом сетке, назовём его нижний ряд. Назовём вернее число в стеке – текущее число. Текущее число возвращаем в массив, а на его место добавляем наименьшее число из верхушек стеков. Если такого числа нет то, добавляем в массив число из нижнего ряда, повторяем эти действия до тех пор, пока все стеки не станут пустыми.

Достоинства и недостатки алгоритма.

Пасьянсная сортировка является подвидом сортировки вставками. Сортировка вставками является устойчивой сортировкой.

Достоинства сортировки:

Сохранение взаимного расположения равных элементов

Сложность n*log(n).

Недостатки сортировки:

Для обеспечения устойчивости требуется дополнительное время и память.

Функции.

template<typename T>

void patienceSort(std::vector<T>&array, int size, int(*comparator)(const T&, const T&)) – функция, сортирующая массив. В функцию передаются: std::vector<T>&array - массив элементов типа T по ссылке, int size – количество элементов в массиве, int(*comparator)(const T&, const T&) – указатель на функцию компаратор. Функция компаратор возвращает 1, если левый элемент больше правого, -1 если левый элемент меньше правого, 0 – элементы равны.

template <typename T>

T minimum(std::vector<T>& vec, int(*comparator)(const T&, const T&)) – функция для нахождения минимума в массиве. В функцию передаются: std::vector<T>& vec - массив элементов типа T по ссылке, int(*comparator)(const T&, const T&) – указатель на функцию компаратор. Функция компаратор возвращает 1, если левый элемент больше правого, -1 если левый элемент меньше правого, 0 – элементы равны. Функция возвращает минимальный элемент массива типа T.

Тестирование программы.

№	Входные данные	Результат
1	3 1 4 7 8 2	----------PART ONE---------- ---------making stacks--------- ------------------------------ push first elem of array to the stack First stack top: 3 ------------------------------ Check all elements of array and push them to stacks -- Array elem 1 1 <= Stack(0) top: 3 Array elem pushed to the Stack(1) -- -- Array elem 4 4 > tops of all stacks Create new stack and add it to the right new stack top: 4 -- -- Array elem 7 7 > tops of all stacks Create new stack and add it to the right new stack top: 7 -- -- Array elem 8 8 > tops of all stacks Create new stack and add it to the right new stack top: 8 -- -- Array elem 2 2 <= Stack(1) top: 4 Array elem pushed to the Stack(2) -- ----------PART TWO---------- ---------bottom row--------- ----making sorted array----

		Stacks tops: 3 4 _ _ Bottom row : 1 2 7 8 Left elem of bottom row: 1 goes to array --- --- Find new left element is stacks tops No element in stacks tops less then 2 Stacks tops: 3 4 _ _ Bottom row : _ 2 7 8 Left elem of bottom row: 2 goes to array --- --- Find new left element is stacks tops 3 4 less then 7 min is: 3 3 goes to the bottom row Stacks tops: _ 4 _ _ Bottom row : _ 3 7 8 Left elem of bottom row: 3 goes to array --- --- Find new left element is stacks tops 4 less then 7 min is: 4 4 goes to the bottom row Stacks tops: _ _ _ _ Bottom row : _ 4 7 8 Left elem of bottom row: 4 goes to array --- --- Find new left element is stacks tops No element in stacks tops less then 7 Stacks tops: _ _ _ _ Bottom row : _ _ 7 8 Left elem of bottom row: 7 goes to array --- --- Find new left element is stacks tops No element in stacks tops less then 8 Stacks tops: _ _ _ _ Bottom row : _ _ _ 8 Left elem of bottom row: 8 goes to array --- array: 3 1 4 7 8 2

		patience sorted array: 1 2 3 4 7 8 sorted array: 1 2 3 4 7 8 ARRAYS ARE EQUAL
2	6 32 24 20 54 8 44 26 1 10 22 34 12 16 18 45	array: 6 32 24 20 54 8 44 26 1 10 22 34 12 16 18 45 patience sorted array: 1 6 8 10 12 16 18 20 22 24 26 32 34 44 45 54 sorted array: 1 6 8 10 12 16 18 20 22 24 26 32 34 44 45 54 ARRAYS ARE EQUAL
3	1 2 3 a 5 b c 7 8	error not integer in array
4	37 4 38 40 33 3 1 16 31 20 17 54 25 39 53 35 11 10	array: 37 4 38 40 33 3 1 16 31 20 17 54 25 39 53 35 11 10 patience sorted array: 1 3 4 10 11 16 17 20 25 31 33 35 37 38 39 40 53 54 sorted array: 1 3 4 10 11 16 17 20 25 31 33 35 37 38 39 40 53 54 ARRAYS ARE EQUAL

Вывод.

Была реализована пасьянсная сортировка на языке C++.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main.cpp

```
#include "InOut.h"
#include "Patience_Sort.h"

int comparator(const int& a, const int& b){
    if ( a > b)
        return 1;
    if (a == b)
        return 0;
    if ( a < b)
        return -1;
    return 0;
}



int main(int argc, char** argv) {
    std::vector<int> arr;
    if(makeArr(argc, argv, arr)){
        return 1;
    }
    std::vector<int> copy = arr;
    int size = arr.size();
    for(int i = 0; i < size; i++){
        cout<<arr[i]<<" ";
    }
    cout<<endl;
    patienceSort(arr, size, comparator);
    cout<<"array:                  ";
    for(int i = 0; i < size; i++){
        cout<<copy[i]<<" ";
    }
    cout<<endl;
    sort(copy.begin(), copy.end());
    std::string output;
    cout<<"patience sorted array: ";
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
        output += std::to_string(arr[i]) + " ";
    }
    cout << endl;
    cout<<"sorted array:           ";
    for (int i = 0; i < size; i++) {
        cout << copy[i] << " ";
    }
    cout << endl;
    if (argc > 1) {
        ofstream outfile(optarg, ios::app);
        outfile << output;
    }
    if(arr == copy) {
        cout<<"\033[1;32mARRAYS ARE EQUAL\033[0m\n";
```

```
    } else cout<<"ERROR"<<endl;
    return 0;
}
```

Файл InOut.h

```
#ifndef AISD_LB3_INOUT_H
#define AISD_LB3_INOUT_H

#include"structs.h"

int makeArr(int argc, char** argv, std::vector<int>& arr);
int getNum(string& input);
void skip (string& str, int n = 1);

#endif //AISD_LB3_INOUT_H
```

Файл InOut.cpp

```
#include "InOut.h"

int makeArr(int argc, char** argv, std::vector<int>& arr){
    if(argc == 1) {
        cout<<"Write a statement: ";
        string res;
        getline(cin, res);
        while (!res.empty()){
            if(!(isdigit(res[0]) || res[0] == '-')){
                cout<<" error not integer in array\n";
                arr.clear();
                return 1;
            }
            int n = getNum(res);
            arr.push_back(n);
            skip(res,1);
        }
        return 0;
    }
    int option = 0;
    while ((option = getopt(argc,argv,"hf:"))!=-1){
        switch (option) {
            case 'h': cout<<"If you want read from file use flag -
f<filename>\n";return 0;
            case 'f': cout<<"read from file - "<<optarg<<endl;
                ifstream infile(optarg);
                if (!infile) {
                    cout << "> File can't be open!" << endl;
                    return 1;
                }
                string str;
                string num;
                getline(infile, str);
                while (!str.empty()){
                    if(!(isdigit(str[0]) || str[0] == '-')){
                        cout<<"not integer in array\n";
                        arr.clear();
```

```
                        return 1;
                    }
                    int n = getNum(str);
                    arr.push_back(n);
                    skip(str,1);
                }
                return 0;
        }
    }
    return 0;
}

int getNum(string& str){
    string strNum;
    while (isdigit(str[0]) || (strNum.length() == 0 && str[0] == '-'))
{
        strNum += str[0];
        skip(str, 1);
    }
    return stoi(strNum);
}

void skip(string& str, int n){
    if (str.length() >= n) {
        str = str.substr(n);
    }
}
```

Файл PatienceSort.h

```
#ifndef AISD_LB4_PATIENCE_SORT_H
#define AISD_LB4_PATIENCE_SORT_H
#include "structs.h"

template <typename T>
T minimum(std::vector<T>& vec, int(*comparator)(const T&, const T&)){
    int size = vec.size();
        if (size == 1)
            return vec[0];
        T min = vec[0];
        for (int i = 1; i < size; i++) {
            if (comparator(vec[i], min) == -1) {
                min = vec[i];
            }
        }
        return min;
}

template<typename T>
void        patienceSort(std::vector<T>&array,        int        size,
int(*comparator)(const T&, const T&)) {
    cout << "\033[1;31m----------PART ONE----------\033[0m\n"
            "\033[1;31m---------making      stacks---------\033[0m\n"<<
endl;
    std::vector<std::stack<T>> stacksArray;
    std::stack<T> buf;
    stacksArray.push_back(buf);
```

```
    stacksArray[0].push(array[0]); // добавляем первый элемет массива в
самый левый стэк
    cout << "-------------------------------\n"
            "push first elem of array to the stack\n"
            "First stack top: " << stacksArray[0].top() <<
         "\n--------------------------------" << endl;
    bool pushFlag = false; // флаг отвечающий за то добавился элемент в
стек или надо создавать новый.
    int arrStackSize = stacksArray.size();
    cout << "Check all elements  of array and push them to stacks" <<
endl;
    for (int i = 1; i < size; i++) {
        for (int j = 0; j < arrStackSize; j++) { // проверяем все стэки
            // если текущий элемент меньше чем значение в стэке
            if  (comparator(array[i],  stacksArray[j].top())  ==  -1  ||
comparator(array[i], stacksArray[j].top()) == 0) {
                cout << "\n-----------------------------------------\n"
                     << "Array elem " << array[i] << "\n" << array[i]
                     <<  "  <=  "  <<  "Stack("  <<  j  <<  ")  top:  "  <<
stacksArray[j].top()
                     << "\n" << "Array elem pushed to the Stack(" << j+1
<< ")"
                     << "\n-----------------------------------------
" << endl;
                stacksArray[j].push(array[i]);
                pushFlag = true;
                break;
            } else continue;
        }
        if (!pushFlag) {
            std::stack<T> buffer; // создаём новый стэк и добавляем его
вправо.
            stacksArray.push_back(buffer);
            arrStackSize = stacksArray.size();
            stacksArray[arrStackSize - 1].push(array[i]);
            cout  <<  "\n-----------------------------------------
\n"
                 << "Array elem " << array[i] << "\n" << array[i] << "
> tops of all stacks\n"
                 << "Create new stack and add it to the right\n" << "new
stack top: "
                 << stacksArray[arrStackSize - 1].top()
                 <<  "\n-----------------------------------------" <<
endl;
            pushFlag = false;
        } else {
            pushFlag = false;
            continue;
        }
    }

/*////////////////////////////////////////////////////////////////////
////////////////////////////////*/
    cout << "\n\033[1;31m----------PART TWO----------\033[0m\n"
            "\033[1;31m---------bottom row---------\033[0m\n"
            "\033[1;31m----making sorted array----\033[0m\n";
    int k = 0; // положение в изначальном массиве.
    std::vector<T> demonsrateVector;
```

```
    for (int i = 0; i < arrStackSize; i++) {
        demonsrateVector.push_back(stacksArray[i].top());
    }
    std::stack<T> buttomRow;
    //создаём нижний ряд.
    for (int i = arrStackSize - 1; i >= 0; i--) {
        buttomRow.push(stacksArray[i].top());
        stacksArray[i].pop();
    }
    while (k < size) {
        if(!buttomRow.empty()) {
            //вывод верхнего ряда стэков и нижнего ряда для обработки
            cout<<"Stacks tops: ";
            for (int i = 0; i < arrStackSize; i++) {
                if(!stacksArray[i].empty())
                    cout << stacksArray[i].top() << " ";
                else cout<<"_ ";
            }
            cout<<"\nBottom row : ";
            for(int j = 0; j < arrStackSize - buttomRow.size(); j++)
                cout<<"_ ";
            for (int i = 0; i < buttomRow.size(); i++) {
                cout<<demonsrateVector[i]<< " ";
            }
            cout<<"\n";
            cout<< "Left elem of bottom row: "<< buttomRow.top()<< "
goes to array"
            <<"\n----------------------------------------"<<endl;

/*////////////////////////////////////////////////////////////////
///*/
            array[k++] = buttomRow.top(); // добавление элемента в
отсортированный массив
            buttomRow.pop();
            std::vector<T> vec;
            if(buttomRow.empty()){ // если в нижнем ряду нет элементов
то добавляем наименьший из стэков
                for (int s = 0; s < arrStackSize; s++) {
                    if (!stacksArray[s].empty()){
                        vec.push_back(stacksArray[s].top());        //
добавляем в массив все верхние элементы стэков
                    }
                }
                if(vec.size() != 0) {
                    T min = minimum(vec, comparator); // находим минимум
в массиве
                    vec.clear();
                    for (int s = 0; s < arrStackSize; s++) {
                        if (!stacksArray[s].empty()) {
                            if (stacksArray[s].top() == min) {
                                buttomRow.push(stacksArray[s].top());
// добавляем в нижний ряд наименьший элемент из стэков
                                stacksArray[s].pop();
                                break;
                            }
                        }
                    }
                }
```

```
            }
            if(!buttomRow.empty())  {  //  поиск  нового  элемента  для
ниэнего ряда.
                cout << "-----------------------------------------
-\n"
                        "Find new left element is stacks tops\n";
                int len = arrStackSize - buttomRow.size();
                for (int s = 0; s < len; s++) {
                    if (!stacksArray[s].empty()) {
                        if (stacksArray[s].top() < buttomRow.top()) {
                            vec.push_back(stacksArray[s].top());     //
добавляем в массив все верхние элементы стэков
                        }
                    }
                }
                if (vec.size() != 0) {
                    for (int i = 0; i < vec.size(); i++) {
                        cout << vec[i] << " ";
                    }
                    cout << " less then " << buttomRow.top() << "\n";
                    T min = minimum(vec, comparator); // находим минимум
в массиве
                    cout << "min is: " << min << endl;
                    vec.clear();
                    for (int s = 0; s < len; s++) {
                        if (!stacksArray[s].empty()) {
                            if (stacksArray[s].top() == min) {
                                cout  <<  min  <<  "  goes  to  the  bottom
row\n";
                                buttomRow.push(stacksArray[s].top());
// добавляем в нижний ряд наименьший элемент из стэков
                                stacksArray[s].pop();
                                break;
                            }
                        }
                    }
                } else {
                    cout << "No element in stacks tops less then " <<
buttomRow.top()<<endl;
                }
                //обновление массива для вывода промежуточных данных
                cout<<"\n";
                demonsrateVector.clear();
                int stSize = buttomRow.size();
                for (int j = 0; j < stSize; j++){
                    demonsrateVector.push_back(buttomRow.top());
                    buttomRow.pop();
                }
                for(int a = stSize-1; a>=0; a--){
                    buttomRow.push(demonsrateVector[a]);
                }
            }
        }
    }
}

#endif //AISD_LB4_PATIENCE_SORT_H
```