ММИНОБРНАУКИ 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. 9384 _____ Мосин К.К.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Реализовать сортировку, сравнить с другими возможными.

Задание.

ВАРИАНТ 22.

Реализовать пасьянсную сортировку.

Выполнение работы.

Дан массив чисел. Он разбивается на кучки, реализованные на базе стека, следующим образом: наименьшие элементы должны расположиться как можно левее, но есть очередное число больше последнего в кучке, то оно смещается правее. В результате, со всех доступных(верхних) элементов создается отсортированный ряд, также основанный на базе стека. Так как наименьший элемент находится как можно левее, то созданный ряд уже отсортирован. Далее, интерес представляют доступный элемент из нижнего ряда и кучки, соответствующие порядковому номеру верхнего элемента в ряду и левее. Это позволяет не делать лишних проверок, тем самым сведя алгоритм к худшему времени O(nlog(n)). Пасьянсная сортировка основана на принципе сортировки вставками. Сравнения данной сортировки с другими представлены в табл. 1.

Таблица 1 – Сравнения сортировок

Название сортировки	Лучшее время	Худшее время
Пасьянсная сортировка	O(n)	O(nlog(n))
Сортировка пузырьком	O(n)	O(n^2)
Сортировка вставками	O(n)	O(n^2)

Тестирование.

Результаты тестирования представлены в табл. 2.

Таблица 2 – Результаты тестирования

Входные данные	Выходные данные
1	1
2 1 3	1 2 3
3 3 3 3 2	2 3 3 3 3

Выводы.

В ходе выполнения лабораторный работы была реализована пасьянсная сортировка, а также была проделана работа со стеком.

ПРИЛОЖЕНИЕ
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <cmath>
#include <ctime>
#include "stack.h"

void solitareSort(int *, size_t);

int main(){
    size_t count;
    int *array;
    std::string temp;
    std::cout << "Type C to write array in console or R to rand: ";
    std::cin >> temp;
    if (temp == "C"){
        std::cout << "Type count of array: ";
        std::cin >> count;
        array = new int[count];
        std::cout << "Type array: ";
        for (size_t i = 0; i < count; i++)
            std::cin >> array[i];
    }
    else if (temp == "R"){
        size_t remainder;
        std::cout << "Type count of array: ";
        std::cin >> count;
        std::cout << "Type max num in rand arr: ";
        std::cin >> remainder;
        array = new int[count];
        for (size_t i = 0; i < count; i++)
            array[i] = rand() % remainder - rand() % remainder;
    }

    std::cout << "Array: ";
    for (size_t i = 0; i < count; i++)
        std::cout << array[i] << ' ';
    std::cout << std::endl;

    clock_t start = clock();
    solitareSort(array, count);
    clock_t end = clock();

    std::cout << "Sort array: ";
    for (size_t i = 0; i < count; i++)
        std::cout << array[i] << ' ';
    std::cout << std::endl;

    std::cout << "TIME = " << (double)(end - start) / CLOCKS_PER_SEC << " s" << std::endl;
    delete[] array;
    return 0;
}

void solitareSort(int *array, size_t count){
    if (count < 2)
```

```
    return;

Stack<int> *heap = new Stack<int>[count];
Stack<int> row;
int size = 1;

heap[0].push(array[0]);
for (int i = 1; i < count; i++){
    for (int j = 0; j < size; j++){
        if (array[i] <= heap[j].top()){
            heap[j].push(array[i]);
            break;
        }
        else if (j == size - 1){
            heap[size++].push(array[i]);
            break;
        }
    }

    std::cout << "HEAP = ";
    for (int j = 0; j < size; j++)
        std::cout << heap[j].top() << " ";
    std::cout << std::endl;
}
std::cout << "HEAP COUNT = " << size << std::endl;
std::cout << "-----" << std::endl;

for (int i = size - 1; i >= 0; i--)
    row.push(heap[i].pop());

std::cout << "ROW top = " << row.top() << std::endl;
std::cout << "HEAP = ";
for (int j = 0; j < size; j++)
    if (!heap[j].empty())
        std::cout << heap[j].top() << " ";
std::cout << std::endl;
std::cout << "-----" << std::endl;

array[0] = row.pop();
int removed = 1;
for (int i = 1; i < count; i++){
    int min, index = 0;
    for (int i = 0; i < size; i++){
        if (!heap[i].empty()){
            min = heap[i].top();
            index = i;
            break;
        }
        else{
            index = -1;
        }
    }

    if (index != -1){
        for (int j = 0; j < removed; j++){
            if (!heap[j].empty() && heap[j].top() <= min){
                min = heap[j].top();
                index = j;
            }
```

```
            }
            std::cout << "MIN = " << min << " INDEX = " << index << std::endl;
        }

        if (!row.empty()){
            if (min <= row.top() && index != -1)
                row.push(heap[index].pop());
            else
                removed++;
        }
        else{
            row.push(heap[index].pop());
        }

        std::cout << "ROW top = " << row.top() << std::endl;
        array[i] = row.pop();
    }

    delete[] heap;
}
```

Название файла: stack.h

```
#ifndef STACK_H
#define STACK_H

#include <cstring>

template <typename T>
class Stack
{
public:
    Stack(){
        this->data = new T[this->memory];
    }

    ~Stack(){
        delete this->data;
    }

    void push(T value){
        if (count == memory){
            memory *= 2;
            T *temp = new T[memory];
            std::memcpy(temp, data, memory * sizeof(T));
            delete this->data;
            data = temp;
        }
        this->data[this->count] = value;
        this->count++;
    }

    T pop(){
        if (!this->empty())
            return this->data[--this->count];
        else
            std::cout << "Stack is empty" << std::endl;
```

```
        return (T)NULL;
    }

    T top(){
        if (!this->empty())
            return this->data[this->count - 1];
        else
            std::cout << "Stack is empty" << std::endl;

        return (T)NULL;
    }

    bool empty(){
        return this->count == 0;
    }

private:
    T *data;
    size_t count = 0;
    size_t memory = 10;
};

#endif
```