

Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации
Ордена Трудового Красного Знамени федеральное государственное
бюджетное образовательное учреждение высшего образования
“Московский технический университет связи и информатики”

Кафедра: "Математическая кибернетика и информационные
технологии"

Лабораторная работа №1 по дисциплине “Структуры и алгоритмы
обработки данных” по теме “Методы сортировки”

Выполнил студент
группы БФИ1902
вариант №9
Крутиков С.С.

г. Москва, 2021

Содержание

1	Задание на лабораторную работу	3
2	Ход выполнения лабораторной работы	3
2.1	Листинги программы	3
2.2	Результаты выполнения программы.....	10
Вывод	12

1 Задание на лабораторную работу

Задание на лабораторную работу представлено на рисунке 1.

Задание №2:

Написать генератор случайных матриц(многомерных), который принимает опциональные параметры **m**, **n**, **min_limit**, **max_limit**, где **m** и **n** указывают размер матрицы, а **min_lim** и **max_lim** - минимальное и максимальное значение для генерируемого числа . По умолчанию при отсутствии параметров принимать следующие значения:

$$m = 50$$

$$n = 50$$

$$\text{min_limit} = -250$$

$$\text{max_limit} = 1000 + (\text{номер своего варианта})$$

Задание №3:

Реализовать методы сортировки строк числовой матрицы в соответствии с заданием. Оценить время работы каждого алгоритма сортировки и сравнить его со временем стандартной функции сортировки. Испытания проводить на сгенерированных матрицах.

Методы:

Выбором	Вставкой	Обменом	Шелла	Турнирная	Быстрая сортировка	Пирамидальная
---------	----------	---------	-------	-----------	--------------------	---------------

Рисунок 1 – Задание на лабораторную работу

2 Ход выполнения лабораторной работы

2.1 Листинги программы

Код в классе Main представлен на листинге 1.

Листинг 1 – Код в классе Main

```
import java.util.Scanner;
import java.util.Arrays;

public class Main{

    public static int[][] mat;
    public static int m, n, min_limit, max_limit;

    public static void main(String[] args) {
        System.out.println("ЛАБОРАТОРНАЯ РАБОТА №1\nВыполнил студент группы
ВФИ1902 Крутиков Степан Сергеевич");
        hello();
    }
}
```

```

        mat_gen();
        selectionSort();
        insertionSort();
        exchangeSort();
        shellSort();
        heapSort();
        quickSort();
        tournamentSort();
    }

    public static void hello(){
        System.out.println();
        System.out.println("@Задание 1@");

        System.out.println("Hello, World!");
    }

    public static void mat_gen(){
        System.out.println();
        System.out.println("@Задание 2");

        m = 50;
        n = 50;
        min_limit = -250;
        max_limit = 1000 + 9;
        String var = "";

        Scanner sc = new Scanner(System.in);

        System.out.println("Введите значение m или отбейте enter");
        var = sc.nextLine();
        if (!var.equals(""))
            m = Integer.parseInt(var);

        System.out.println("Введите значение n или отбейте enter");
        var = sc.nextLine();
        if (!var.equals(""))
            n = Integer.parseInt(var);

        System.out.println("Введите значение min_limit или отбейте enter");
        var = sc.nextLine();
        if (!var.equals(""))
            min_limit = Integer.parseInt(var);

        System.out.println("Введите значение max_limit или отбейте enter");
        var = sc.nextLine();
        if (!var.equals(""))
            max_limit = Integer.parseInt(var);

        System.out.println("\n Source array");
        mat = new int[m][n];
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                mat[i][j] = (int) (Math.random() * (max_limit - min_limit)) +
min_limit;
                System.out.print(mat[i][j] + " ");
            }
            System.out.println("\n");
        }
    }

    public static void output_mat(int [][] mat){
        System.out.println();
    }

```

```

        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(mat[i][j] + " ");
            }
            System.out.println("\n");
        }
    }

    public static void selectionSort(){ // Сортировка выбором
        int[][] sel_mat = new int[m][n];
        for (int i = 0; i < m; i++) {
            System.arraycopy(mat[i], 0, sel_mat[i], 0, n);
        }

        for (int i = 0; i < m; i++)
            for (int j = 0; j < n; j++){
                int min = mat[i][j]; // Минимум определяем как текущий
элемент
                int index = j; // Запоминаем индекс условного минимума

                for (int k = j + 1; k < n; k++) // Цикл для прохода по
массиву (начиная со следующего элемента) в поисках элемента, который меньше
условного минимума
                    if (mat[i][k] < min){
                        min = mat[i][k];
                        index = k;
                    }

                if (j != index) { // Если новый минимум был найден, то меняем
местами текущий элемент и найденный минимум
                    int z = mat[i][j];
                    mat[i][j] = mat[i][index];
                    mat[i][index] = z;
                }
            }

        System.out.println("\n Selection sort");
        output_mat(sel_mat);
    }

    public static void insertionSort() { // Сортировка вставкой
        int[][] ins_mat = new int[m][n];
        for (int i = 0; i < m; i++) {
            System.arraycopy(mat[i], 0, ins_mat[i], 0, n);
        }

        for (int i = 0; i < m; i++) {
            for (int j = 1; j < n; j++) { // Цикл для прохода по
неотсортированной части массива
                for (int k = j; k >= 1; k--) { // Цикл для прохода по
отсортированной части массива
                    if (ins_mat[i][k] < ins_mat[i][k - 1]) { // Поиск места
для вставки текущего элемента
                        int z = ins_mat[i][k];
                        ins_mat[i][k] = ins_mat[i][k - 1];
                        ins_mat[i][k - 1] = z;
                    }
                }
                else // Если нету элемента, меньше текущего, то текущий
остаётся на своем месте
                    break;
            }
        }
    }
}

```

```

        System.out.println("\n Insertion sort");
        output_mat(ins_mat);
    }

    public static void exchangeSort(){
        int[][] ex_mat = new int[m][n];
        for (int i = 0; i < m; i++) {
            System.arraycopy(mat[i], 0, ex_mat[i], 0, n);
        }

        for (int i = 0; i < m; i++) {
            boolean needIteration = true;
            while (needIteration) {
                needIteration = false;
                for (int j = 1; j < n; j++) { // Проход по массиву, в каждой
итерации которого максимальный элемент будет оказываться справа
                    if (ex_mat[i][j] < ex_mat[i][j - 1]) {
                        int z = ex_mat[i][j];
                        ex_mat[i][j] = ex_mat[i][j - 1];
                        ex_mat[i][j - 1] = z;
                        needIteration = true;
                    }
                }
            }
        }

        System.out.println("\n Exchange sort");
        output_mat(ex_mat);
    }

    public static void shellSort(){
        int[][] shell_mat = new int[m][n];
        for (int i = 0; i < m; i++) {
            System.arraycopy(mat[i], 0, shell_mat[i], 0, n);
        }

        int h = 1; // Описывается переменная шага
        while (h*3 < m) // Находится максимальное значение для кнотовской
последовательности
            h = h * 3 + 1;

        while(h >= 1) { // Вызывается метод сортировки с h, равной
максимальному значению кнотовской последовательности, а после в h
записывается предыдущий элемент кнотовской последовательности
            hSort(shell_mat, h);
            h = h/3;
        }

        System.out.println("\n Shell sort");
        output_mat(shell_mat);
    }

    public static void hSort(int[][] arr, int h) {
        for (int i = 0; i < m; i++) {
            for (int j = h; j < n; j++) {
                for (int k = j; k >= h; k = k - h) { // Выполняется
пузырьковой сортировка, но с шагом h
                    if (arr[i][k] < arr[i][k - h]) {
                        int z = arr[i][k];
                        arr[i][k] = arr[i][k - h];
                        arr[i][k - h] = z;
                    }
                }
            }
        }
    }

```

```

        else
            break;
    }
}

}

}

public static void heapSort(){
    int[][] heap_mat = new int[m][n];
    for (int i = 0; i < m; i++) {
        System.arraycopy(mat[i], 0, heap_mat[i], 0, n);
    }

    for (int i = 0; i < m; i++)
        sort(heap_mat, i);

    System.out.println("\n Heap sort");
    output_mat(heap_mat);
}

public static void sort(int[][] arr, int i)
{
    // Построение кучи (перегруппируем массив)
    for (int j = n / 2 - 1; j >= 0; j--)
        heapify(arr, n, i, j);

    // Один за другим извлекаем элементы из кучи
    for (int j=n-1; j>=0; j--)
    {
        // Перемещаем текущий корень в конец
        int temp = arr[i][0];
        arr[i][0] = arr[i][j];
        arr[i][j] = temp;

        // Вызываем процедуру heapify на уменьшенной куче
        heapify(arr, j, i, 0);
    }
}

// Процедура для преобразования в двоичную кучу поддерева с корневым узлом i, что является
// индексом в arr[]. n - размер кучи
public static void heapify(int[][] arr, int n, int i, int j)
{
    int largest = j; // Инициализируем наибольший элемент как корень
    int l = 2*j + 1; // левый = 2*i + 1
    int r = 2*j + 2; // правый = 2*i + 2

    // Если левый дочерний элемент больше корня
    if (l < n && arr[i][l] > arr[i][largest])
        largest = l;

    // Если правый дочерний элемент больше, чем самый большой элемент на
    // данный момент
    if (r < n && arr[i][r] > arr[i][largest])
        largest = r;
    // Если самый большой элемент не корень
    if (largest != j)
    {
        int swap = arr[i][j];
        arr[i][j] = arr[i][largest];
        arr[i][largest] = swap;
    }
}

```

```

        // Рекурсивно преобразуем в двоичную кучу затронутое поддереву
        heapify(arr, n, i, largest);
    }
}

public static void quickSort(){
    int[][] quick_mat = new int[m][n];
    for (int i = 0; i < m; i++) {
        System.arraycopy(mat[i], 0, quick_mat[i], 0, n);
    }

    for (int i = 0; i < m; i++)
        quickSort(quick_mat, 0, n - 1, i);

    System.out.println("\n Quick sort");
    output_mat(quick_mat);
}

public static void quickSort(int[][] array, int low, int high, int i) {
    if (array.length == 0)
        return;//завершить выполнение если длина массива равна 0

    if (low >= high)
        return;//завершить выполнение если уже нечего делить

    // выбрать опорный элемент
    int middle = low + (high - low) / 2;
    int opora = array[i][middle];

    // разделить на подмассивы, который больше и меньше опорного элемента
    int ilow = low, jhigh = high;
    while (ilow <= jhigh) {
        while (array[i][ilow] < opora) {
            ilow++;
        }

        while (array[i][jhigh] > opora) {
            jhigh--;
        }

        if (ilow <= jhigh) { //меняем местами
            int temp = array[i][ilow];
            array[i][ilow] = array[i][jhigh];
            array[i][jhigh] = temp;
            ilow++;
            jhigh--;
        }
    }

    // вызов рекурсии для сортировки левой и правой части
    if (low < jhigh)
        quickSort(array, low, jhigh, i);

    if (high > ilow)
        quickSort(array, ilow, high, i);
}

public static void tournamentSort(){
    System.out.println("\n Tournament sort");
    int[] arr1 = new int[n];
    for (int i = 0; i < m; i++) {

```



```

        System.arraycopy(mat[i], 0, arr1, 0, n);

        Sort(arr1);

        System.out.println("\n" + Arrays.toString(arr1));
    }
}

private static class Node
{
    public int data;
    public int id;

    public Node()
    {

    }

    public Node(int _data, int _id)//
    {
        data = _data;
        id = _id;
    }
}

public static void Adjust(Node[] data, int idx)
{
    while(idx != 0)
    {
        if(idx % 2 == 1)
        {
            if(data[idx].data < data[idx + 1].data)
            {
                data[(idx - 1)/2] = data[idx];
            }
            else
            {
                data[(idx-1)/2] = data[idx + 1];
            }
            idx = (idx - 1)/2;
        }
        else
        {
            if(data[idx-1].data < data[idx].data)
            {
                data[idx/2 - 1] = data[idx-1];
            }
            else
            {
                data[idx/2 - 1] = data[idx];
            }
            idx = (idx/2 - 1);
        }
    }
}

public static void Sort(int[] data)
{
    int nNodes = 1;
    int nTreeSize;
    while(nNodes < data.length)
    {
        nNodes *= 2;
    }
}

```

```

    }
    nTreeSize = 2 * nNodes - 1;

    Node[] nodes = new Node[nTreeSize];

    int i, j;
    int idx;
    for( i = nNodes - 1; i < nTreeSize; i++)
    {
        idx = i - (nNodes - 1);
        if(idx < data.length)
        {
            nodes[i] = new Node(data[idx], i);
        }
        else
        {
            nodes[i] = new Node(Integer.MAX_VALUE, -1);
        }
    }

    for( i = nNodes - 2; i >= 0; i--)
    {
        nodes[i] = new Node();
        if(nodes[i * 2 + 1].data < nodes[i * 2 + 2].data)
        {
            nodes[i] = nodes[i*2 + 1];
        }
        else
        {
            nodes[i] = nodes[i*2 + 2];
        }
    }
    for( i = 0; i < data.length; i++)
    {
        data[i] = nodes[0].data;
        nodes[nodes[0].id].data = Integer.MAX_VALUE;
        Adjust(nodes, nodes[0].id);
    }
}
}

```

2.2 Результаты выполнения программы

После ввода всех необходимых данных для генерации массива программа выводит отсортированные массивы различными методами. Результат выполнения представлен на рисунках 2 и 3.

```
SIAOD, LAB 1 > src > Main > heapify
Project
Run: Main x
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
ЛАБОРАТОРНАЯ РАБОТА №1
Выполнил студент группы БФИ1902 Крутиков Степан Сергеевич

@Задание 1@
Hello, World!

@Задание 2
Введите значение m или отбейте enter
2
Введите значение n или отбейте enter
5
Введите значение min_limit или отбейте enter
8
Введите значение max_limit или отбейте enter
100

Source array
3 74 10 36 38

22 30 91 4 55

Selection sort

3 74 10 36 38

22 30 91 4 55

Insertion sort

3 10 36 38 74

4 22 30 55 91

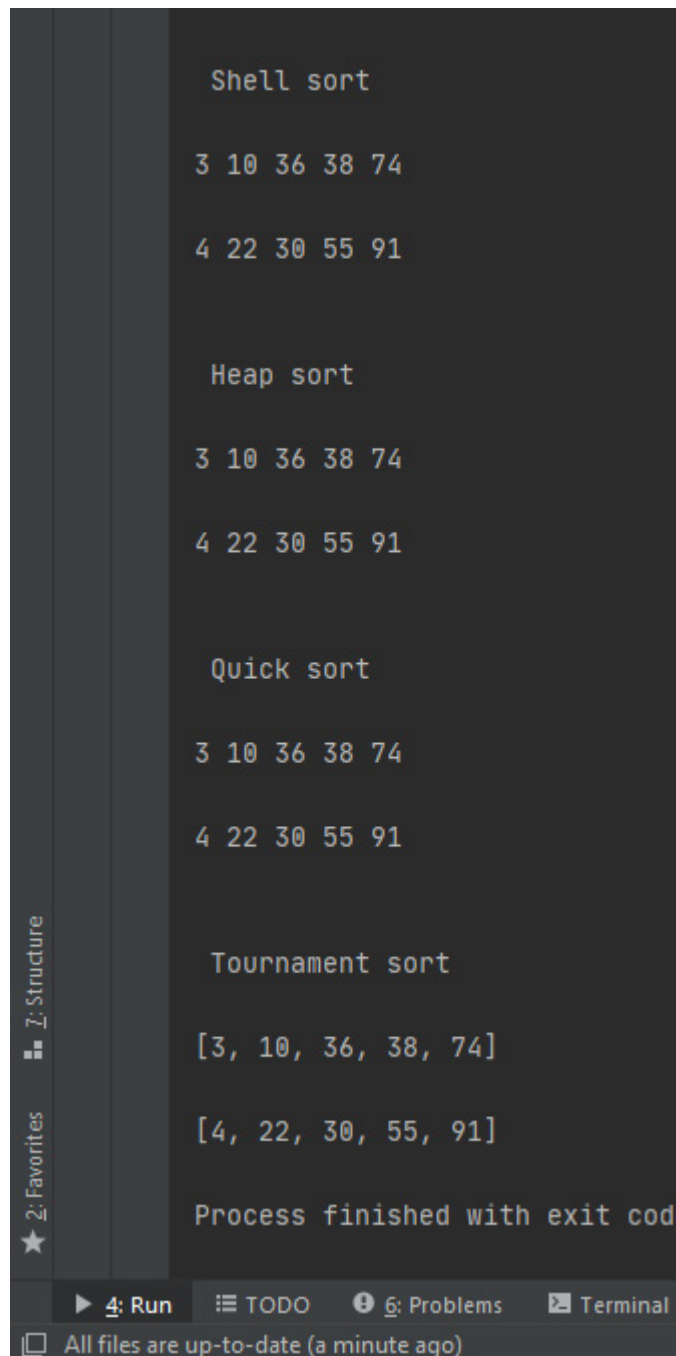
Exchange sort

3 10 36 38 74

4 22 30 55 91

4: Run 6: Problems Terminal Build
All files are up-to-date (a minute ago)
```

Рисунок 2 – Результат выполнения программы (Часть 1)



```
Shell sort

3 10 36 38 74

4 22 30 55 91


Heap sort

3 10 36 38 74

4 22 30 55 91


Quick sort

3 10 36 38 74

4 22 30 55 91


Tournament sort

[3, 10, 36, 38, 74]

[4, 22, 30, 55, 91]

Process finished with exit cod
```

Рисунок 3 – Результат выполнения программы (Часть 2)

Вывод

В результате выполнения данной лабораторной работы были получены навыки реализации основных алгоритмов сортировки.

Список использованных источников

- 1 ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления
- 2 ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления