

Федеральное агентство связи

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»

Выполнил студент

группы БФИ1902

Чернышов Дмитрий

(Вариант 19)

Москва 2021

Методы сортировки

Задание № 1:

- Создать Jupyter Notebook со следующим наименованием:
Lab1_Группа_ФИО
- Создать новую ячейку с помощью кнопки
- В созданной ячейке по указанной ниже форме заполните оглавление файла, заменив наименование группы и вписав свое ФИО
- С помощью кнопки запустите выполнение всех ячеек.
- После выполнения у вас должна отформатироваться ячейка с оглавлением и должен выполняться “Hello, World!”

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:

$m = 50$

$n = 50$

$\text{min_limit} = -250$

$\text{max_limit} = 1000 + (19)$

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.

Ниже будет представлен код данной лабораторной работы

```

import java.util.Scanner;

public class sorting {
    public static void main(String[] args) {
        System.out.println("Задание 1");
        System.out.println("Hello world!");
        sorting s = new sorting();
        s.Z2();
    }

    public void Z2() {
        System.out.println("Задание 2");
        Scanner sc = new Scanner(System.in);
        String m1 = sc.nextLine();
        String n1 = sc.nextLine();
        String min_lim1 = sc.nextLine();
        String max_lim1 = sc.nextLine();
        if (n1.equals(""))
            n1 = "50";
        if (m1.equals(""))
            m1 = "50";
        if (min_lim1.equals(""))
            min_lim1 = "-250";
        if (max_lim1.equals(""))
            max_lim1 = "1019";
        int n = Integer.parseInt(n1);
        int m = Integer.parseInt(m1);
        int min_lim = Integer.parseInt(min_lim1);
        int max_lim = Integer.parseInt(max_lim1);
        int[][] arr = new int[n][m];
        int max = 0, index = 0;
        int min = 0;
        for (int i = 0; i < n; i++) {
            System.out.print("\n");
            for (int j = 0; j < m; j++) {
                arr[i][j] = (int) ((Math.random() * (max_lim - min_lim)) +
min_lim);
                System.out.print(arr[i][j] + "\t");
            }
            System.out.println();
        }
        System.out.println("Задание 3");
        System.out.println("Выбором");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                min = arr[i][j]; // записываем в максимум
каждый элемент строки в матрице
                index = j; // находим его индекс
                for (int c = j + 1; c < m; c++) { // находим максимум и
индекс элемент
                    if (arr[i][c] < min) {
                        min = arr[i][c];
                        index = c;
                    }
                }
                if (j != index) { //если текущий элемент не
совпадает с максимальным
                    int zero = arr[i][j];
                    arr[i][j] = min; //меняем местами
                    arr[i][index] = zero;
                }
            }
        }
        for (int i = 0; i < n; i++) {

```

```

        for (int j = 0; j < m; j++) {
            System.out.print(arr[i][j] + "\t");
        }
        System.out.println();
    }
    System.out.println("Вставкой");
    //вставляет максимум сразу в нужное место
    for (int i = 0; i < n; i++) {
        for (int j = 1; j < m; j++) {
            for (int c = j; c > 0 && arr[i][c - 1] > arr[i][c]; c--) {
                //если предыдущий больше текущего
                int z = arr[i][c];
                arr[i][c] = arr[i][c - 1];
            }
            //меняем местами
            arr[i][c - 1] = z;
        }
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            System.out.print(arr[i][j] + "\t");
        }
        System.out.println();
    }
    System.out.println("Обменом");
    for (int i = 0; i < n; i++) {
        boolean needIteration = true;
        //конструкция для перехода к след строке
        while (needIteration) {
            needIteration = false;
            for (int j = 1; j < m; j++) {
                if (arr[i][j] < arr[i][j - 1]) {
                    //сравнивает соседние элементы и меняет местами если правый больше
                    int z = arr[i][j];
                    arr[i][j] = arr[i][j - 1];
                    arr[i][j - 1] = z;
                    needIteration = true;
                }
            }
        }
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            System.out.print(arr[i][j] + "\t"); //вывод массива
        }
        System.out.println();
    }
    System.out.println("Шелла");
    int d = m / 2; //шаг
    for (int i = 0; i < n; i++) {
        while (d > 0) { //если шаг больше нуля
            for (int j = 0; j < m - d; j++) {
                int q = j;
                while (q >= 0 && arr[i][q] > arr[i][q + d]) {
                    //меняет
                    int temp = arr[i][q];
                    arr[i][q] = arr[i][q + d];
                    arr[i][q + d] = temp;
                    q--;
                }
            }
            d = d / 2;
        }
    }
}

```

```

    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            System.out.print(arr[i][j] + "\t");
        }
        System.out.println();
    }
    System.out.println("Пирамидальная");
    int[] arr1 = new int[m];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            arr1[j] = arr[i][j];
        }

        heapSort(arr1);

        for (int l = 0; l < m; l++) {
            System.out.print(arr1[l] + "\t");
        }
        System.out.println();
    }
    System.out.println("Быстрая сортировка");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            arr1[j] = arr[i][j];
        }

        quickSort(arr1, 0, m - 1);

        for (int l = 0; l < m; l++) {
            System.out.print(arr1[l] + "\t");
        }
        System.out.println();
    }
    System.out.println("Турнирная сортировка");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            arr1[j] = arr[i][j];
        }

        Sort(arr1);

        for (int l = 0; l < m; l++) {
            System.out.print(arr1[l] + "\t");
        }
        System.out.println();
    }
}

static void heapify(int[] array, int length, int i) {
    int leftChild = 2 * i + 1; //позиции
    //дочерних элементов
    int rightChild = 2 * i + 2;
    int largest = i; //индекс самого
    //большого изначально записываем как текущий

    // если левый дочерний больше родительского
    if (leftChild < length && array[leftChild] > array[largest]) {
        largest = leftChild;
    }

    // если правый дочерний больше родительского
    if (rightChild < length && array[rightChild] > array[largest]) {

```

```

        largest = rightChild;
    }

    // если должна произойти замена
    if (largest != i) {
        int temp = array[i];
        array[i] = array[largest];
        array[largest] = temp;
        heapify(array, length, largest);
    }
}

public static void heapSort(int[] array) {
    if (array.length == 0) return;
    int length = array.length;
    // Построение кучи (перегруппируем массив)
    for (int i = length / 2 - 1; i >= 0; i--)
        heapify(array, length, i);
    // Один за другим извлекаем элементы из кучи
    for (int i = length - 1; i >= 0; i--) {
        // Перемещаем текущий корень в конец
        int temp = array[0];
        array[0] = array[i];
        array[i] = temp;
        // Вызываем процедуру heapify на уменьшенной куче
        heapify(array, i, 0);
    }
}

static int partition(int[] array, int begin, int end) {
    int pivot = end;

    int counter = begin;
    for (int i = begin; i < end; i++) {
        if (array[i] < array[pivot]) { //перемещает
            значения меньше чем pivot в левую от него часть, большие в правую
            int temp = array[counter];
            array[counter] = array[i];
            array[i] = temp;
            counter++;
        }
    }
    int temp = array[pivot];
    array[pivot] = array[counter];
    array[counter] = temp;

    return counter;
}

public static void quickSort(int[] array, int begin, int end) {
    if (end <= begin) return;
    int pivot = partition(array, begin, end);
    //выбираем опорный элемент
    quickSort(array, begin, pivot - 1); //левая часть
    quickSort(array, pivot + 1, end); //правая часть
}

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

    public Node() {
    }
}

```

```

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

public 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 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);
    }
}

```

```

    }
}
}

```

Далее будет представлен результат лабораторной работе на рисунках 1-4.

```

Run: sorting
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
Задание 1
Hello world!
Задание 2
13
10
-3
5
-1 0 -3 0 1 -3 -4 1 0 -1
3 1 1 -1 -1 4 0 1 -1 3
0 4 4 2 0 1 -3 -4 -2 -3
-2 -4 -4 0 2 -3 -3 -3 -3 -4
-1 0 4 -1 0 -2 -1 -1 -3 3
-1 -4 -1 -2 -2 4 3 -3 -3 -1
3 -4 -1 4 4 -3 2 3 -1 4
2 -3 2 0 4 4 2 0 3 2
1 1 -3 1 0 0 -4 4 0 1
-4 4 -1 0 4 0 -1 -4 0 0
Задание 3
Выбором
-4 -3 -3 -1 -1 0 0 0 1 1
-1 -1 -1 0 1 1 1 3 3 4
-4 -3 -3 -2 0 0 1 2 4 4
-4 -4 -4 -3 -3 -3 -3 -2 0 2
-3 -2 -1 -1 -1 0 0 3 4
-4 -3 -3 -2 -2 -1 -1 -1 3 4
-4 -3 -1 -1 2 3 3 4 4 4
-3 0 0 2 2 2 2 3 4 4
-4 -3 0 0 0 1 1 1 1 4

```

Рисунок 1 – Результат работы

The screenshot shows an IDE window titled "Lab1" with a project named "sorting". The main editor displays the output of a Java program, which is a sorting algorithm. The output is organized into three sections: "Задание 3", "Выбором", and "Вставкой". Each section shows the initial array and the state of the array after each step of the sorting process. The array is: -4 -3 -3 -1 -1 0 0 0 1 1. The output shows the array after each step of the sorting process, with the sorted elements in bold. The output is as follows:

```
Задание 3
Выбором
-4 -3 -3 -1 -1 0 0 0 1 1
-1 -1 -1 0 1 1 1 3 3 4
-4 -3 -3 -2 0 0 1 2 4 4
-4 -4 -4 -3 -3 -3 -3 -2 0 2
-3 -2 -1 -1 -1 -1 0 0 3 4
-4 -3 -3 -2 -2 -1 -1 -1 3 4
-4 -3 -1 -1 2 3 3 4 4 4
-3 0 0 2 2 2 2 3 4 4
-4 -3 0 0 0 1 1 1 1 4
-4 -4 -1 -1 0 0 0 0 4 4

Вставкой
-4 -3 -3 -1 -1 0 0 0 1 1
-1 -1 -1 0 1 1 1 3 3 4
-4 -3 -3 -2 0 0 1 2 4 4
-4 -4 -4 -3 -3 -3 -3 -2 0 2
-3 -2 -1 -1 -1 -1 0 0 3 4
-4 -3 -3 -2 -2 -1 -1 -1 3 4
-4 -3 -1 -1 2 3 3 4 4 4
-3 0 0 2 2 2 2 3 4 4
-4 -3 0 0 0 1 1 1 1 4
-4 -4 -1 -1 0 0 0 0 4 4

Обменом
-4 -3 -3 -1 -1 0 0 0 1 1
-1 -1 -1 0 1 1 1 3 3 4
-4 -3 -3 -2 0 0 1 2 4 4
-4 -4 -4 -3 -3 -3 -3 -2 0 2
-3 -2 -1 -1 -1 -1 0 0 3 4
-4 -3 -3 -2 -2 -1 -1 -1 3 4
-4 -3 -1 -1 2 3 3 4 4 4
-3 0 0 2 2 2 2 3 4 4
-4 -3 0 0 0 1 1 1 1 4
-4 -4 -1 -1 0 0 0 0 4 4

Шелла
-4 -3 -3 -1 -1 0 0 0 1 1
-1 -1 -1 0 1 1 1 3 3 4
-4 -3 -3 -2 0 0 1 2 4 4
-4 -4 -4 -3 -3 -3 -3 -2 0 2
-3 -2 -1 -1 -1 -1 0 0 3 4
-4 -3 -3 -2 -2 -1 -1 -1 3 4
-4 -3 -1 -1 2 3 3 4 4 4
-3 0 0 2 2 2 2 3 4 4
-4 -3 0 0 0 1 1 1 1 4
-4 -4 -1 -1 0 0 0 0 4 4
```

Рисунок 2 – Результат работы

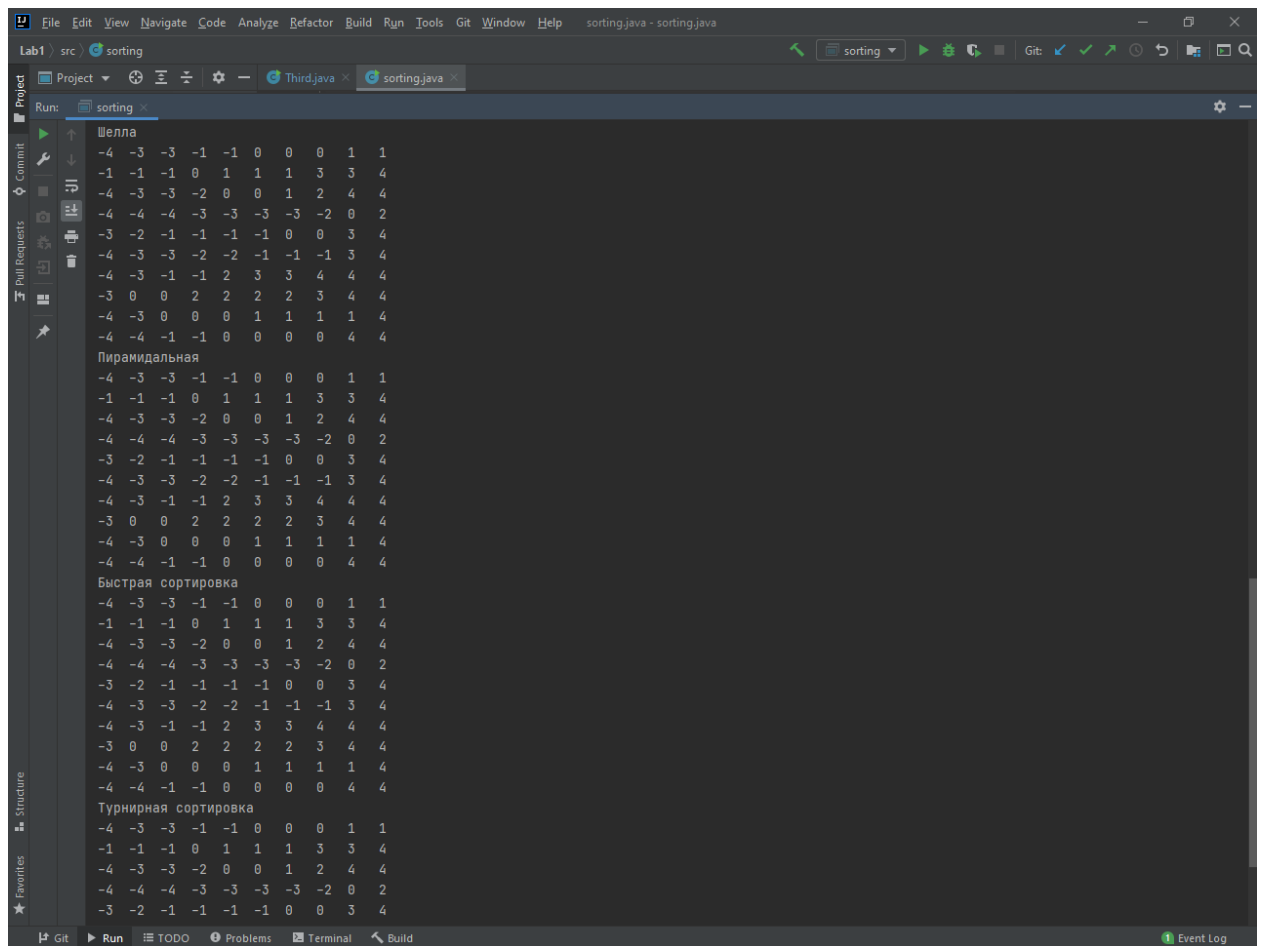


Рисунок 3 – Результат работы

```
Lab1 | src | sorting
Run: sorting
-3 0 0 2 2 2 2 3 4 4
-4 -3 0 0 0 1 1 1 1 4
-4 -4 -1 -1 0 0 0 0 4 4
Пирамидальная
-4 -3 -3 -1 -1 0 0 0 1 1
-1 -1 -1 0 1 1 1 3 3 4
-4 -3 -3 -2 0 0 1 2 4 4
-4 -4 -4 -3 -3 -3 -2 0 2
-3 -2 -1 -1 -1 -1 0 0 3 4
-4 -3 -3 -2 -2 -1 -1 -1 3 4
-4 -3 -1 -1 2 3 3 4 4 4
-3 0 0 2 2 2 2 3 4 4
-4 -3 0 0 0 1 1 1 1 4
-4 -4 -1 -1 0 0 0 0 4 4
Быстрая сортировка
-4 -3 -3 -1 -1 0 0 0 1 1
-1 -1 -1 0 1 1 1 3 3 4
-4 -3 -3 -2 0 0 1 2 4 4
-4 -4 -4 -3 -3 -3 -2 0 2
-3 -2 -1 -1 -1 -1 0 0 3 4
-4 -3 -3 -2 -2 -1 -1 -1 3 4
-4 -3 -1 -1 2 3 3 4 4 4
-3 0 0 2 2 2 2 3 4 4
-4 -3 0 0 0 1 1 1 1 4
-4 -4 -1 -1 0 0 0 0 4 4
Турнирная сортировка
-4 -3 -3 -1 -1 0 0 0 1 1
-1 -1 -1 0 1 1 1 3 3 4
-4 -3 -3 -2 0 0 1 2 4 4
-4 -4 -4 -3 -3 -3 -2 0 2
-3 -2 -1 -1 -1 -1 0 0 3 4
-4 -3 -3 -2 -2 -1 -1 -1 3 4
-4 -3 -1 -1 2 3 3 4 4 4
-3 0 0 2 2 2 2 3 4 4
-4 -3 0 0 0 1 1 1 1 4
-4 -4 -1 -1 0 0 0 0 4 4
Process finished with exit code 0
```

Рисунок 4 – Результат работы

Вывод

В данной лабораторной работе мы познакомились с основными методами сортировки и научились реализовывать их.