Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительной техники»

**Отчет**

По лабораторной работе №7

По курсу «Логика и основы алгоритмизации в инженерных задачах»

На тему «Поиск расстояний во взвешенном графе»

Выполнил студент гр.20ВВ3

Мирясов Н.А.

Проверили:

Митрохин М.А.

Юрова О.В.

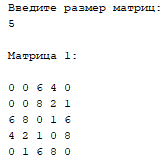
Пенза, 2021

**Цель:** изучить алгоритм поиска расстояний во взвешенном графе.

**Задание 1:**

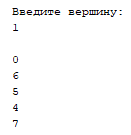
1. Сгенерировали (используя генератор случайных чисел) матрицу

смежности для неориентированного взвешенного графа G. Вывели матрицу на экран.



2. Для сгенерированного графа осуществили процедуру поиска

расстояний, реализованную в соответствии с приведенным выше описанием.

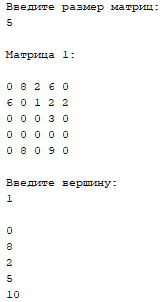


3.\* Сгенерировали (используя генератор случайных чисел) матрицу

смежности для ориентированного взвешенного графа G. Вывели матрицу

на экран и осуществили процедуру поиска расстояний, реализованную в

соответствии с приведенным выше описанием.



**Задание 2:**

1. Модернизировали программу так, чтобы получить возможность запуска программы с параметрами командной строки (см. описание ниже). В качестве параметра должны указываться тип графа (взвешенный или нет) и наличие ориентации его ребер (есть ориентация или нет).

**Листинг задания 1 - 2:**

package lb7;

import java.util.\*;

public class mission2 {

public static void main(String[] args) {

boolean orientation = false;

boolean weighted = false;

for (int i = 0; i < args.length; i++){

if (args[i].equals("1"))

orientation = true;

if (args[i].equals("2"))

weighted = true;

}

System.out.println("Enter the size of the matrices:");

Scanner in = new Scanner(System.in);

int n = in.nextInt();

int[][] matrix = new int[n][n];

rand(n, matrix, orientation, weighted);

output(n, matrix);

Queue<Integer> queue = new LinkedList<>();

int[] vis = new int[n];

for (int i = 0; i < n; i++)

vis[i] = 10000;

System.out.println();

System.out.println("Enter the vertex:");

Scanner in1 = new Scanner(System.in);

int v = in1.nextInt() - 1;

System.out.println();

BFSD(n, v, vis, queue, matrix);

for (int i = 0; i < n; i++)

System.out.println(vis[i] + " ");

}

public static void rand(int n, int[][] matrix, boolean orientation, boolean weighted) {

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

matrix[i][j] = (int) ((Math.random() \* 2) + 0);

if (!orientation)

matrix[j][i] = matrix[i][j]; // Отвечает за не ориентированность графа

if (i == j) {

matrix[i][j] = 0;

}

if (weighted)

if (matrix[i][j] == 1) {

matrix[i][j] = (int) (Math.random() \* 10);

if (!orientation)

matrix[j][i] = matrix[i][j]; // Отвечает за не ориентированность графа

}

}

}

}

public static void output(int n, int[][] matrix) {

System.out.println("\nMatrices:\n");

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

System.out.print(matrix[i][j] + " ");

}

System.out.println();

}

}

public static void BFSD(int n, int v, int[] vis, Queue<Integer> queue, int[][] matrix){

vis[v] = 0;

queue.offer(v);

while (queue.peek() != null){

int s = queue.poll();

//System.out.print(s + 1 + " ");

for (int i = 0; i < n; i++){

if (matrix[s][i] > 0 & vis[i] > vis[s] + matrix[s][i]){

queue.offer(i);

vis[i] = vis[s] + matrix[s][i];

}

}

}

}

}

**Вывод:** изучили алгоритм поиска расстояний во взвешенном графе с помощью обхода в ширину.