**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

по дисциплине «Алгоритмы и структуры данных»

на тему: «Линейные динамические структуры»

Выполнил: студент гр. ИП-31

Казутин П. Н.

Принял: ст. преподаватель

Косинов Г. П.

Гомель 2021

**Цель работы:** овладеть навыками алгоритмизации и программирования задач, в которых используются графы.

**Практическая часть:**

9. Реализовать алгоритм Дейкстры для нахождения кратчайшего пути

между двумя заданными вершинами графа.

**Main.java**

package by.ktsin;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

System.out.print("Enter matrix size: ");

int size = in.nextInt();

Random rnd = new Random();

int[][] matrix = new int[size][size];

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

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

matrix[i][j] = rnd.nextInt(2);

matrix[j][i] = matrix[i][j];

}

}

System.out.println("Matrix:");

printMatrix(matrix);

System.out.print("Enter start point: ");

int start = inputInBounds(0, matrix.length - 1);

System.out.print("Enter finish point: ");

int finish = inputInBounds(0, matrix.length - 1);

int[] path = searchPath(matrix, start, finish);

System.out.print("Path: ");

for (int p : path) {

System.out.printf("->%3d", p);

}

}

public static void printMatrix(int[][] matrix) {

//header

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

System.out.printf("%3d", i);

}

System.out.println("");

int rowN = 1;

for (int[] row : matrix) {

System.out.printf("%3d", rowN);

for (int col : row) {

System.out.printf("%3d", col);

}

System.out.print("\n");

rowN++;

}

}

public static int inputInBounds(int from, int to) {

int res = 0;

Scanner in = new Scanner(System.in);

do {

System.out.printf("Enter int between %d and %d: ", from, to);

res = in.nextInt();

} while (res < from || res > to);

return res;

}

public static int[] searchPath(int[][] matrix, int start, int finish) {

int[] d = new int[matrix.length]; // минимальное расстояние

int[] v = new int[matrix.length]; // посещенные вершины

Stack<Integer> path = new Stack<>();

int temp, minindex, min;

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

d[i] = Integer.MAX\_VALUE;

v[i] = 1;

}

d[start] = 0;

do {

minindex = Integer.MAX\_VALUE;

min = Integer.MAX\_VALUE;

for (int i = 0; i < matrix.length; i++) { // Если вершину ещё не обошли и вес меньше min

if ((v[i] == 1) && (d[i] < min)) { // Переприсваиваем значения

min = d[i];

minindex = i;

}

}

// Добавляем найденный минимальный вес

// к текущему весу вершины

// и сравниваем с текущим минимальным весом вершины

if (minindex != Integer.MAX\_VALUE) {

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

if (matrix[minindex][i] > 0) {

temp = min + matrix[minindex][i];

if (temp < d[i]) {

d[i] = temp;

}

}

}

v[minindex] = 0;

}

} while (minindex < Integer.MAX\_VALUE);

int a = 0;

System.out.printf("Shortest path between %d and %d is %d\n", start + 1, finish + 1, d[finish]);

if (d[finish] > 0) {

// добавляем соседние вершины пока не появится искомая в списке вершин

// искомая = старт, начальная = финиш

//path.push(finish);

// Восстановление пути

int[] ver = new int[matrix.length]; // массив посещенных вершин

ver[0] = finish; // начальный элемент - конечная вершина

int k = 1; // индекс предыдущей вершины

int weight = d[finish]; // вес конечной вершины

while (finish != start) // пока не дошли до начальной вершины

{

for (int i = 0; i < matrix.length; i++) {// просматриваем все вершины

if (matrix[i][finish] != 0) // если связь есть

{

int tmp = weight - matrix[i][finish]; // определяем вес пути из предыдущей вершины

if (tmp == d[i]) // если вес совпал с рассчитанным

{ // значит из этой вершины и был переход

weight = tmp; // сохраняем новый вес

finish = i; // сохраняем предыдущую вершину

ver[k] = i; // и записываем ее в массив

k++;

}

}

}

}

return Arrays.stream(ver).limit(k).toArray();

} else

return new int[1];

}

}

Результат выполнения:

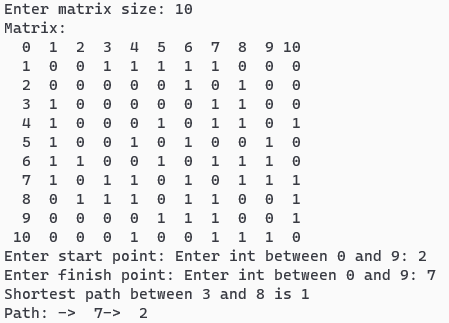


Рисунок 1 – Результат выполнения программы Main.java

**Вывод**: в ходе выполнения лабораторной работы овладеть навыками алгоритмизации и программирования задач, в которых используются графы.