Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра: Информационные технологии и автоматизированные системы

Дисциплина: «Математические методы теории систем»

Лабораторная работа № 2

на тему: «Описание потоков информации в системах»

Выполнил: студент группы АСУ8-23-1м

Зиятдинов Расиль Артурович

Проверил: ст. преп. кафедры ИТАС

Тютюных Артём Александрович

Пермь 2023

**ЗАДАНИЕ**

**Задача 1.** Схеме движения оперативной отчетности в подсистеме оперативного управления производством соответствует информационный граф, представленный на рис. 1. Необходимо формально выявить все свойства данного информационного графа. Реализовать решение задачи на одном из языков программирования. Привести результаты работы программы.

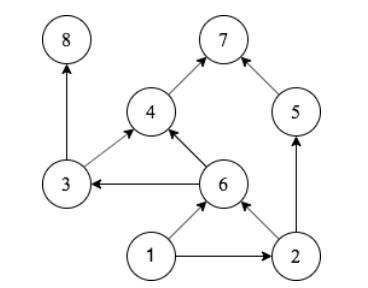


Рисунок 1 – Схема движения документации в подсистеме

оперативного управления производством

**Задача 2.** Выявить все свойства одного информационного графа из предложенных ниже вариантов, используя разработанную программу. Привести результаты работы программы.

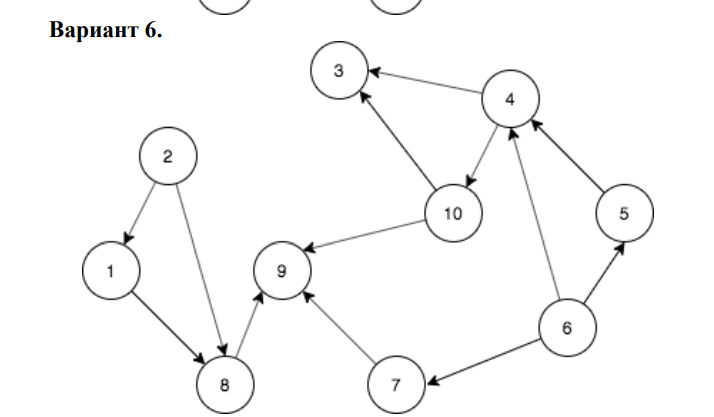
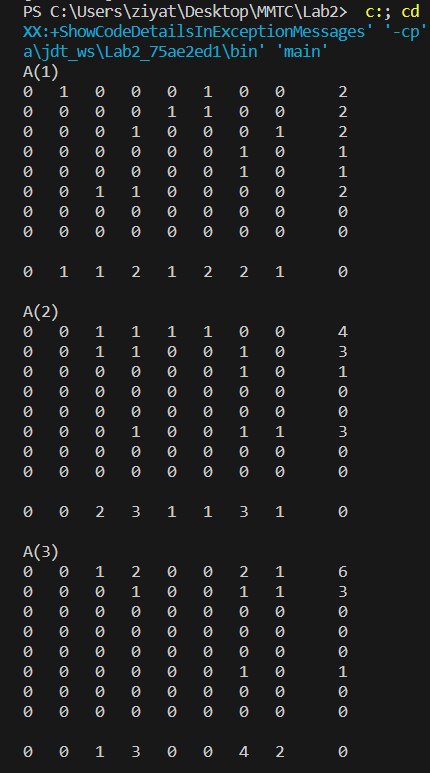
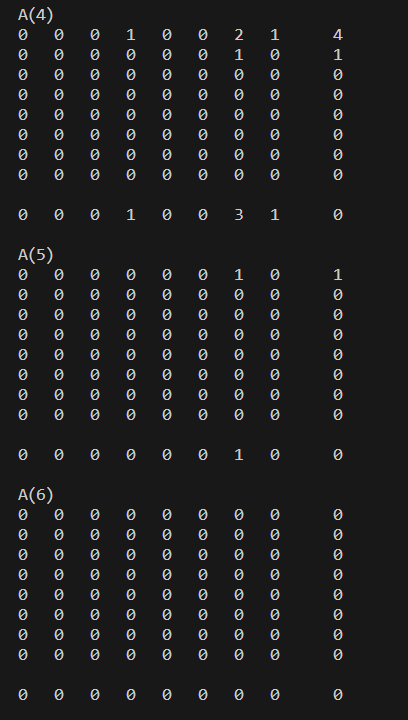
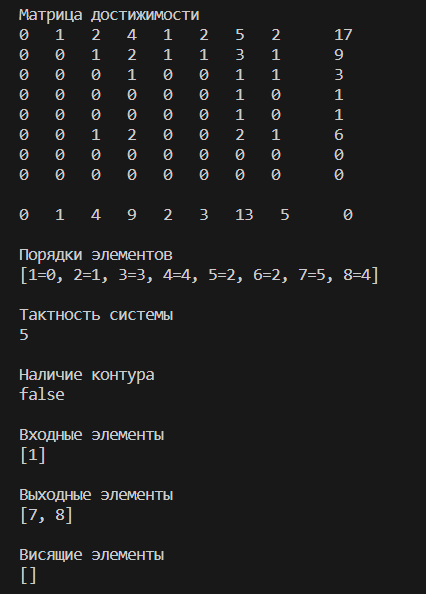
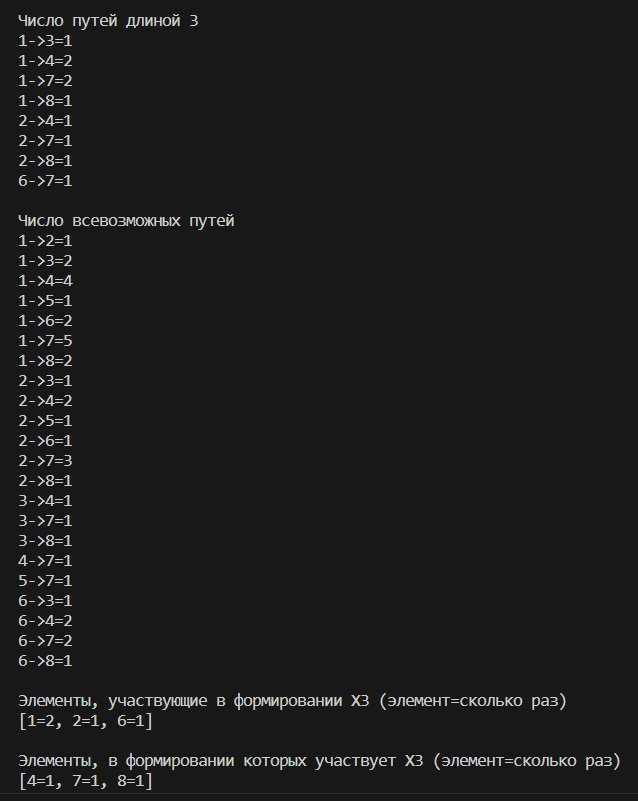
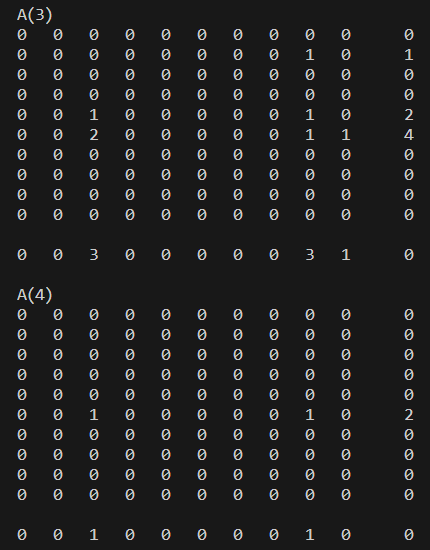
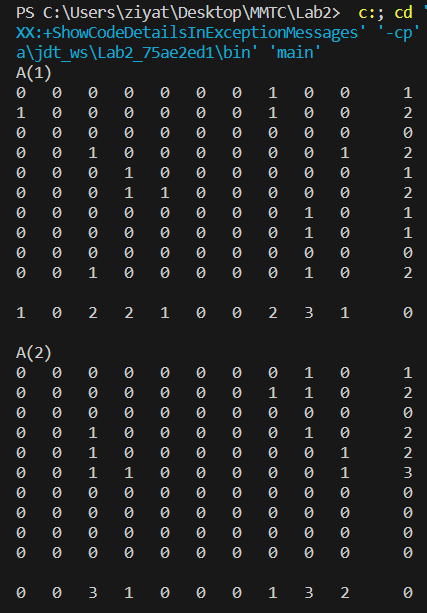
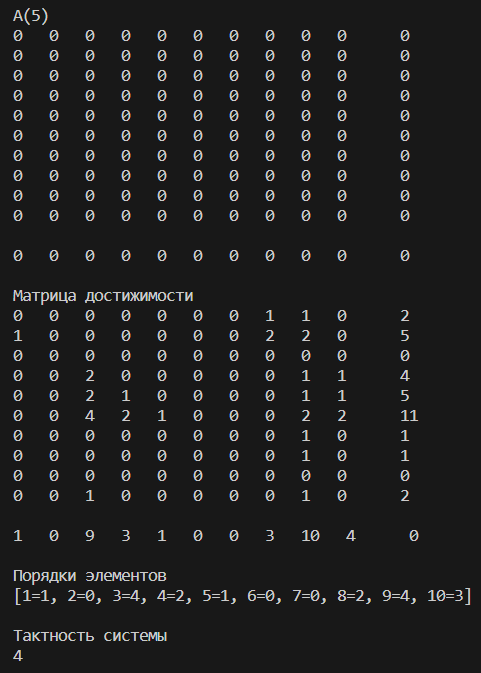
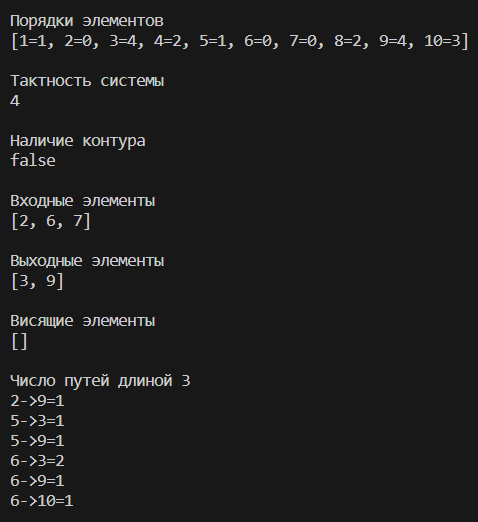


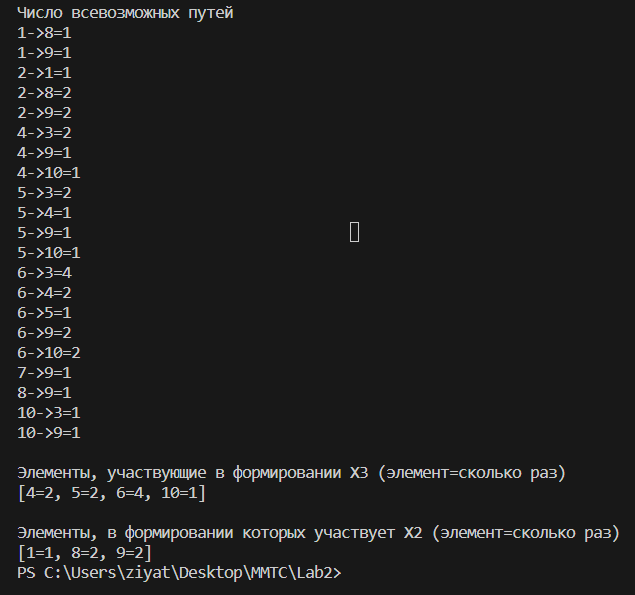
Рисунок 2 – Задание для варианта 6

**ВЫПОЛНЕНИЕ РАБОТЫ**

** Задача 1.**

****

**Задача 2.**

****

**ПРИЛОЖЕНИЕ А**

Листинг 1 – Код программы на языке программирования Java

import java.util.\*;

import java.io.\*;

class Graph {

private List<int[][]> matrices;

private int[][] reachabilityMatrix;

private List<Integer> inputVertices;

private List<Integer> outputVertices;

private List<Integer> hangingVertices;

private HashMap<Integer, Integer> vertexOrders;

public Graph(int[][] matrix) {

matrices = new ArrayList<>();

vertexOrders = new HashMap<>();

inputVertices = new ArrayList<>();

outputVertices = new ArrayList<>();

hangingVertices = new ArrayList<>();

matrices.add(matrix);

}

public List<int[][]> getMatrices() {

return matrices;

}

public HashMap<Integer, Integer> getVertexOrders() {

return vertexOrders;

}

public int[][] getReachabilityMatrix() {

return reachabilityMatrix;

}

private Boolean isZeroMatrix(int[][] matrix) {

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

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

if (matrix[i][j] != 0) {

return false;

}

}

}

return true;

}

public int[] sumArrays(int[] arr1, int[] arr2) {

int[] arr3 = new int[arr1.length];

for (int i = 0; i < arr1.length; i++) {

arr3[i] = arr1[i] + arr2[i];

}

return arr3;

}

public void calculateReachabilityMatrix() {

reachabilityMatrix = new int[matrices.get(0).length][matrices.get(0)[0].length];

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

for (int[][] matrix : matrices) {

reachabilityMatrix[i] = sumArrays(reachabilityMatrix[i], matrix[i]);

}

}

public void exponentiation() {

int k = 0;

int[][] startMatrix = matrices.get(k);

while (!isZeroMatrix(matrices.get(k))) {

int[][] newMatrix = new int[startMatrix.length][startMatrix[0].length];

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

for (int j = 0; j < startMatrix[i].length; j++) {

if (startMatrix[i][j] != 0) {

newMatrix[i] = sumArrays(newMatrix[i], matrices.get(k)[j]);

}

}

}

matrices.add(newMatrix);

k++;

}

}

public void calculateSigma() {

for (int[][] matrix : matrices) {

for (int i = 0; i < matrix.length - 1; i++) {

for (int j = 0; j < matrix[0].length - 1; j++) {

matrix[i][matrix[i].length - 1] += matrix[i][j];

matrix[matrix.length - 1][j] += matrix[i][j];

}

}

}

}

public void calculateVertexOrders() {

int[] arr = matrices.get(0)[matrices.get(0).length - 1];

for (int j = 0; j < arr.length - 1; j++) {

if (arr[j] == 0) {

vertexOrders.put(j + 1, 0);

}

}

for (int i = 0; i < matrices.size() - 1; i++) {

int[] arr1 = matrices.get(i)[matrices.get(0).length - 1];

int[] arr2 = matrices.get(i + 1)[matrices.get(0).length - 1];

for (int j = 0; j < arr1.length - 1; j++) {

if (arr1[j] > 0 && arr2[j] == 0) {

vertexOrders.put(j + 1, i + 1);

}

}

}

System.out.println(vertexOrders.entrySet());

}

public Boolean checkKontur() {

for (int[][] matrix : matrices) {

for (int i = 0; i < matrix.length - 1; i++) {

for (int j = 0; j < matrix[0].length - 1; j++) {

if (matrix[i][j] != 0 && i == j) {

return true;

}

}

}

}

return false;

}

public List<Integer> getInputVerteces() {

for (int i = 0; i < matrices.get(0).length - 1; i++) {

if (matrices.get(0)[matrices.get(0).length - 1][i] == 0) {

inputVertices.add(i + 1);

}

}

return inputVertices;

}

public List<Integer> getOutputVerteces() {

for (int i = 0; i < matrices.get(0).length - 1; i++) {

if (matrices.get(0)[i][matrices.get(0).length - 1] == 0) {

outputVertices.add(i + 1);

}

}

return outputVertices;

}

public List<Integer> getHangingVerteces() {

for (int i = 0; i < matrices.get(0).length - 1; i++) {

if (matrices.get(0)[i][matrices.get(0).length - 1] == matrices.get(0)[matrices.get(0).length - 1][i]

&& matrices.get(0)[matrices.get(0).length - 1][i] == 0) {

hangingVertices.add(i + 1);

}

}

return hangingVertices;

}

public void getWaysNumberByLenght(int l) {

for (int i = 0; i < matrices.get(0).length - 1; i++) {

for (int j = 0; j < matrices.get(0)[0].length - 1; j++) {

if (matrices.get(l - 1)[i][j] != 0) {

System.out.printf("%s->%d=%d%n", i + 1, j + 1, matrices.get(l - 1)[i][j]);

}

}

}

}

public void getAllWaysNumber() {

for (int i = 0; i < reachabilityMatrix.length - 1; i++) {

for (int j = 0; j < reachabilityMatrix[0].length - 1; j++) {

if (reachabilityMatrix[i][j] != 0) {

System.out.printf("%s->%d=%d%n", i + 1, j + 1, reachabilityMatrix[i][j]);

}

}

}

}

public HashMap<Integer, Integer> getIncludeElements(int x) {

HashMap<Integer, Integer> result = new HashMap<>();

for (int i = 0; i < reachabilityMatrix.length - 1; i++) {

if (reachabilityMatrix[i][x-1] != 0) {

result.put(i + 1, reachabilityMatrix[i][x-1]);

}

}

return result;

}

public HashMap<Integer, Integer> getExcludeElements(int x) {

HashMap<Integer, Integer> result = new HashMap<>();

for (int i = 0; i < reachabilityMatrix.length - 1; i++) {

if (reachabilityMatrix[x-1][i] != 0) {

result.put(i + 1, reachabilityMatrix[x-1][i]);

}

}

return result;

}

public void printAllMatrices() {

int i = 1;

for (int[][] matrix : matrices) {

System.out.printf("A(%s)%n", i++);

print(matrix);

System.out.println();

}

}

public void print(int[][] matrix) {

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

if (i == matrix.length - 1) {

System.out.println();

}

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

if (j == matrix[i].length - 1) {

System.out.print(" ");

}

System.out.printf("%s ", matrix[i][j]);

}

System.out.println();

}

}

}

public class main {

public static void main(String[] args) throws FileNotFoundException {

int[][] matrix = readMatrix("graph0.txt");

Graph graph = new Graph(matrix);

graph.exponentiation();

graph.calculateSigma();

graph.printAllMatrices();

graph.calculateReachabilityMatrix();

System.out.printf("Матрица достижимости%n");

graph.print(graph.getReachabilityMatrix());

System.out.println();

System.out.printf("Порядки элементов%n");

graph.calculateVertexOrders();

System.out.println();

System.out.printf("Тактность системы%n");

System.out.printf("%s%n",

Collections.max(graph.getVertexOrders().entrySet(), Map.Entry.comparingByValue()).getValue());

System.out.println();

System.out.printf("Наличие контура%n");

System.out.println(graph.checkKontur());

System.out.println();

System.out.printf("Входные элементы%n");

System.out.println(graph.getInputVerteces());

System.out.println();

System.out.printf("Выходные элементы%n");

System.out.println(graph.getOutputVerteces());

System.out.println();

System.out.printf("Висящие элементы%n");

System.out.println(graph.getHangingVerteces());

System.out.println();

int l = 3;

System.out.printf("Число путей длиной %s%n", l);

graph.getWaysNumberByLenght(l);

System.out.println();

System.out.printf("Число всевозможных путей%n");

graph.getAllWaysNumber();

System.out.println();

int x = 3;

System.out.printf("Элементы, участвующие в формировании X%s (элемент=сколько раз)%n", x);

System.out.println(graph.getIncludeElements(x).entrySet());

System.out.println();

int x2 = 3;

System.out.printf("Элементы, в формировании которых участвует X%s (элемент=сколько раз)%n", x2);

System.out.println(graph.getExcludeElements(x2).entrySet());

}

public static int[][] readMatrix(String filename) throws FileNotFoundException {

int rows = 0;

String line = "";

Scanner inFile = new Scanner(new File(filename));

while (inFile.hasNextLine()) {

rows++;

line = inFile.nextLine();

}

int cols = line.length();

inFile = new Scanner(new File(filename));

int[][] matrix = new int[rows + 1][cols + 1];

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

line = inFile.nextLine();

for (int k = 0; k < cols; k++) {

matrix[i][k] = line.charAt(k) - '0';

}

}

return matrix;

}

}