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 «Определение характеристик графов»

Выполнил студент гр.20ВВ3

Мирясов Н.А.

Проверили:

Митрохин М.А.

Юрова О.В.

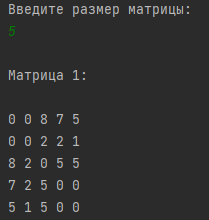
Пенза, 2021

**Цель:** научиться определять характеристике графов.

**Задание 1:**

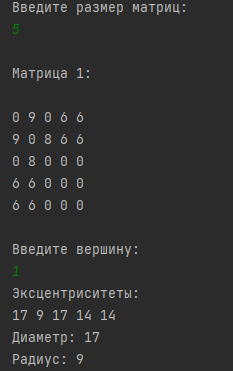
1. Сгенерировали (используя генератор случайных чисел) матрицу

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



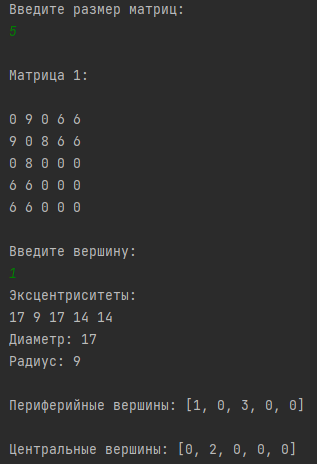
2. Определили радиус и диаметр графа G, используя матрицу смежности

графа.

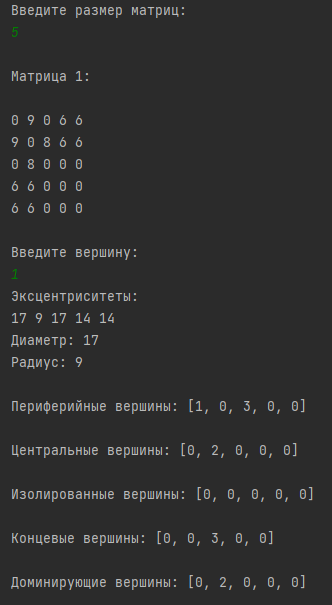


3. Определили подмножества периферийных и центральных вершин

графа G, используя матрицу смежности.

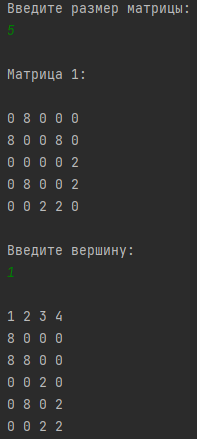


4. Нашли изолированные, концевые и доминирующие вершины.

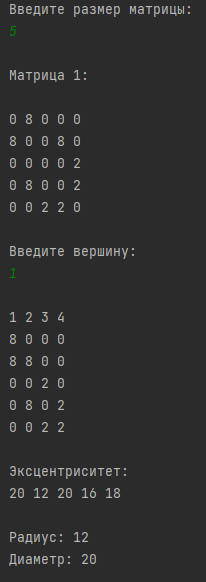


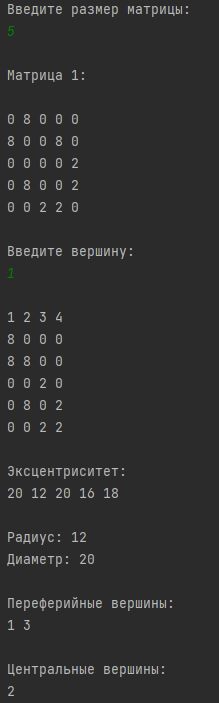
**Задание 2\*:**

1. Построили для графа G матрицу инцидентности.



1. Определили радиус и диаметр графа G, используя матрицу инцидентности графа.



1. Определите подмножества периферийных и центральных вершин графа G, используя матрицу инцидентности.
2. 

**Листинг задания 1:**

import java.util.\*;  
  
public class mission1 {  
 public static void main(String[] args) {  
 System.out.println("Введите размер матриц:");  
 Scanner in = new Scanner(System.in);  
 int n = in.nextInt();  
 int[][] matrix = new int[n][n];  
 int[][] Ex = new int[n][n];  
 int j = 0;  
 int Diametr = 0;  
 int radius = 10000;  
  
 rand(n, matrix);  
 output(n, matrix);  
  
  
  
 System.out.println();  
 System.out.println("Введите вершину:");  
 Scanner in1 = new Scanner(System.in);  
 int v = in1.nextInt() - 1;  
  
  
 for (; v < matrix.length; v++){  
 int[] vis = new int[n];  
 for (int i = 0; i < vis.length; i++)  
 vis[i] = 10000;  
 Queue<Integer> queue = new LinkedList<>();  
 BFSD(n, v, vis, queue, matrix);  
  
 for (int i = 0; i < vis.length; i++){  
 if (vis[i] == 10000)  
 vis[i] = 0;  
 }  
  
 for (int i = 0; i < vis.length; i++){  
 Ex[j][i] = vis[i];  
 }  
  
 j++;  
 int Max = 0;  
 for (int i = 0; i < vis.length; i++){  
 if (vis[i] > Max)  
 Max = vis[i];  
 }  
 if (Max > Diametr & Max != 0)  
 Diametr = Max;  
 if (Max < radius & Max != 0){  
 radius = Max;  
 }  
 }  
  
 System.out.println("Эксцентриситеты: ");  
 int[] Excentr = new int[n];  
 for (int i = 0; i < Ex.length; i++) {  
 int Max = 0;  
 for (int k = 0; k < Ex[i].length; k++){  
 if (Ex[i][k] > Max)  
 Max = Ex[i][k];  
 Excentr[i] = Max;  
 }  
 System.out.print(Excentr[i] + " ");  
 }  
 System.out.println();  
 System.out.println("Диаметр: " + Diametr);  
 System.out.println("Радиус: " + radius);  
  
 mission1\_3(n, Ex, Diametr, radius, Excentr);  
 mission1\_4(n, matrix);  
 }  
  
 private static void mission1\_3(int n, int[][] Ex, int Diametr, int radius, int[] Excentr) {  
 int[] per = new int[n];  
 int[] middle = new int[n];  
  
 for (int i = 0; i < Ex.length; i++){  
 if (Excentr[i] == Diametr)  
 per[i] = i + 1;  
 if (Excentr[i] == radius){  
 middle[i] = i + 1;  
 }  
 }  
  
 System.out.println("\nПериферийные вершины: " + Arrays.toString(per));  
 System.out.println("\nЦентральные вершины: " + Arrays.toString(middle));  
 }  
  
 private static void mission1\_4(int n, int[][] matrix) {  
 int[] Isolirovannie = new int[n];  
 Isolirovannie(n, matrix, Isolirovannie);  
 int[] Conc = new int[n];  
 Conc(n, matrix, Conc);  
 int[] Domination = new int[n];  
 Domination(n, matrix, Domination);  
  
 System.out.println("\nИзолированные вершины: " + Arrays.toString(Isolirovannie));  
 System.out.println("\nКонцевые вершины: " + Arrays.toString(Conc));  
 System.out.println("\nДоминирующие вершины: " + Arrays.toString(Domination));  
 }  
  
 public static void rand(int n, int[][] matrix) {  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 matrix[i][j] = (int) ((Math.random() \* 2) + 0);  
 matrix[j][i] = matrix[i][j];  
  
 if (i == j) {  
 matrix[i][j] = 0;  
 }  
 if (matrix[i][j] == 1) {  
 matrix[i][j] = (int) (Math.random() \* 10);  
 matrix[j][i] = matrix[i][j];  
 }  
 }  
 }  
 }  
  
 public static void output(int n, int[][] matrix) {  
 System.out.println("\nМатрица 1:\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];  
 }  
 }  
 }  
 }  
  
 public static void Isolirovannie(int n, int[][] matrix, int[] Isolirovannie){  
 for (int i = 0; i < matrix.length; i++){  
 int count = 0;  
 for (int j = 0; j < matrix[i].length; j++){  
 count++;  
 if (matrix[i][j] != 0)  
 break;  
 if (matrix[i][j] == 0 & count == matrix.length){  
 Isolirovannie[i] = i + 1;  
 }  
 }  
 }  
 }  
  
 public static void Conc(int n, int[][] matrix, int[] Conc){  
 for (int i = 0; i < matrix.length; i++){  
 int count = 0;  
 for (int j = 0; j < matrix[i].length; j++){  
 if (matrix[i][j] != 0)  
 count++;  
 if (j == matrix.length - 1 & count == 1){  
 Conc[i] = i + 1;  
 }  
 }  
 }  
 }  
  
 public static void Domination(int n, int[][] matrix, int[] Domination){  
 for (int i = 0; i < matrix.length; i++){  
 int count = 0;  
 for (int j = 0; j < matrix[i].length; j++){  
 if (matrix[i][j] != 0)  
 count++;  
 if (j == matrix.length - 1 & count == matrix.length - 1){  
 Domination[i] = i + 1;  
 }  
 }  
 }  
 }  
}

**Листинг задания 2:**

import java.util.\*;  
  
public class mission2 {  
 public static void main(String[] args) {  
 System.*out*.println("Введите размер матрицы:");  
 Scanner in = new Scanner(System.*in*);  
 int n = in.nextInt();  
 int[][] matrix = new int[n][n];  
 int[][] Ex = new int[n][n];  
 int j = 0;  
 int Diametr = 0;  
 int radius = 10000;  
  
 *rand*(n, matrix);  
 *output*(n, matrix);  
  
 System.*out*.println();  
 System.*out*.println("Введите вершину:");  
 Scanner in1 = new Scanner(System.*in*);  
 int v = in1.nextInt() - 1;  
  
 int[][] incidenceMatrix = new int[n][n\*n];  
 *IncidenceMatrix*(n, matrix, incidenceMatrix);  
 }  
  
 public static void rand(int n, int[][] matrix) {  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 matrix[i][j] = (int) ((Math.*random*() \* 2) + 0);  
 matrix[j][i] = matrix[i][j];  
  
 if (i == j) {  
 matrix[i][j] = 0;  
 }  
 if (matrix[i][j] == 1) {  
 matrix[i][j] = (int) (Math.*random*() \* 10);  
 matrix[j][i] = matrix[i][j];  
 }  
 }  
 }  
 }  
  
 public static void output(int n, int[][] matrix) {  
 System.*out*.println("\nМатрица 1:\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 IncidenceMatrix(int n, int[][] matrix, int[][] incidenceMatrix){  
 var s = 1;  
 var num = 0;  
 var x = 0;  
 var d = -1;  
 var r = 10000;  
 int[] extr = new int[n];  
  
  
  
 for (int i = 0; i < n; i++){  
 for(int j = 0; j < n; j++){  
 incidenceMatrix[i][j] = -1;  
 }  
 }  
  
 for (int i = 0; i < n; i++){  
 for(int j = s; j < n; j++){  
 if (matrix[i][j] > 0){  
 incidenceMatrix[i][j] = num;  
 x++;  
 num++;  
 }  
 }  
 s++;  
 }  
  
 s = 1;  
 for (int i = 0; i < n; i++){  
 for(int j = s; j < n; j++){  
 incidenceMatrix[j][i] = incidenceMatrix[i][j];  
 }  
 s++;  
 }  
  
 int[][] M1 = new int[n][x];  
 for (int i = 0; i < n; i++){  
 for(int j = 0; j < x; j++){  
 M1[i][j] = 0;  
 }  
 }  
  
 for (int i = 0; i < n; i++){  
 for(int j = 0; j < n; j++){  
 if (incidenceMatrix[i][j] > - 1){  
 M1[i][incidenceMatrix[i][j]] = matrix[i][j];  
 }  
 }  
 }  
  
 System.*out*.println();  
 for(int i = 0; i < x; i++){  
 System.*out*.print(i + 1 + " ");  
 }  
  
 for(int i = 0; i < n; i++){  
 System.*out*.println();  
 for(int j = 0; j < x; j++){  
 System.*out*.print(M1[i][j] + " ");  
 }  
 }  
  
  
 int[] dist = new int[n];  
 for(int i = 0; i < n; i++){  
 extr[i] = -1;  
 }  
  
 for (int i = 0; i < n; i++){  
 for (int k = 0; k < n; k++){  
 dist[k] = 9999;  
 }  
 *BFSDMIX*(i, dist, M1, n, x);  
 for (int j = 0; j < n; j++)  
 {  
 if (extr[i] < dist[j] && dist[j] != 9999)  
 extr[i] = dist[j];  
 }  
 }  
  
 System.*out*.println("\n\nЭксцентриситет: ");  
 for (int i = 0; i < n; i++){  
 System.*out*.print(extr[i] + " ");  
 }  
  
  
 for (int i = 0; i < n; i ++){  
 if (d < extr[i])  
 d = extr[i];  
 if (r > extr[i] && extr[i] != 0)  
 r = extr[i];  
 }  
  
 System.*out*.println("\n\nРадиус: " + r);  
 System.*out*.println("Диаметр: " + d);;  
 System.*out*.println("\nПереферийные вершины: ");  
 for (int i = 0; i < n; i++){  
 if (extr[i] == d)  
 System.*out*.print(i + 1 + " ");  
 }  
  
 System.*out*.println("\n\nЦентральные вершины: ");  
 for (int i = 0; i < n; i++){  
 if (extr[i] == r)  
 System.*out*.print(i + 1 + " ");  
 }  
 }  
  
 public static void BFSDMIX(int x, int[] dist, int[][] Arr, int n, int p){  
 Queue<Integer> queue = new LinkedList<>();  
 queue.offer(x);  
 dist[x] = 0;  
 while (queue.peek() != null){  
 x = queue.poll();  
 for (int i = 0; i < p; i++){  
 if (Arr[x][i] > 0){  
 for (int j = 0; j < n; j++){  
 if(Arr[j][i] > 0 && dist[j] > dist[x] + Arr[j][i]){  
 queue.offer(j);  
 dist[j] = dist[x] + Arr[j][i];  
 }  
 }  
 }  
 }  
 }  
 }  
}

**Вывод:** научились определять характеристики графов.