Пензенский государственный университет

Кафедра "Вычислительной техники"

**Отчёт**

По лабораторной №8

по дисциплине "Л и ОА в ИТ"

на тему "Определение характеристик графов"

***Выполнили студенты группы 19вв1:***

*Григорьева Д. Д.*

*Мереняшева Е. А.*

***Приняли:***

*Митрохин М. А.*

*Юрова О. В.*

Пенза 2020

# Цель работы: определить характеристики графов.

**Лабораторное задание:**

Задание 1

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного взвешенного графа G. Выведите матрицу на экран.

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

3. Определите подмножества периферийных и центральных вершин графа G, используя матрицу смежности.

4. Найдите изолированные, концевые и доминирующие вершины.

**Теоретическая часть:**

Эксцентриситет вершины – расстояние до наиболее удаленной вершины графа.

Максимальный эксцентриситет среди эксцентриситетов всех вершин графа называется диаметром графа G и обозначается через D(G).

Вершина v i называется периферийной, если её эксцентриситет равен диаметру графа.

Минимальный из эксцентриситетов вершин графа называется его радиусом и обозначается через r(G).

Вершина v i называется центральной, если её эксцентриситет равен радиусу графа.

Множество всех центральных вершин графа называется его центром.

Граф G может иметь единственную центральную вершину или несколько центральных вершин.

Степенью вершины графа G называется число инцидентных ей ребер. Степень вершины v i обозначается через deg (v, i ).

Вершина v i со степенью 0 называется изолированной, со степенью 1 – концевой.

Вершина графа, смежная с каждой другой его вершиной, называется доминирующей.

**Практическая часть:**

**Листинг:**

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

class Program

{

static void Izolir(int[,] M,int size)

{

int vertexIz;

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

{

vertexIz = 0;

for (int j = 0; j < size; j++)

{

vertexIz = vertexIz + M[i, j];

}

if (vertexIz == 0)

{

Console.WriteLine("Вершина " + i + " является изолированной");

}

}

}

static void Concev(int[,] M, int size)

{

int vertexCon;

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

{

vertexCon = 0;

for (int j = 0; j < size; j++)

{

if (M[i, j] != 0) {

vertexCon++;

}

}

if (vertexCon == 1)

{

Console.WriteLine("Вершина " + i + " является концевой");

}

}

}

static void Domin(int[,] M, int size)

{

int vertexDom;

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

{

vertexDom = 0;

for (int j = 0; j < size; j++)

{

if (M[i, j] != 0)

{

vertexDom++;

}

}

if (vertexDom == size-1)

{

Console.WriteLine("Вершина " + i + " является доминирующей");

}

}

}

static void BFSD(int v, int[,] matrix, int[] DIST, int size)

{

Queue<int> queue = new Queue<int>(); //Создаем новую очередь

queue.Enqueue(v); //Помещаем v в очередь

DIST[v] = 0;

Console.Write("Рез.обх.: ");

while (queue.Count != 0)

{

v = queue.Dequeue();//Удаляем первый элемент из очереди

Console.Write(v);

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

{

if (matrix[v, i] > 0 && DIST[i]>DIST[v]+ matrix[v,i])

{

queue.Enqueue(i); //Помещаем i в очередь

DIST[i] = DIST[v] + matrix[v, i];

}

}

}

Console.Write(" ");

}

static void Main(string[] args)

{

Random random = new Random();

Console.Write("Введите размерность матрицы:");

int size = Convert.ToInt32(Console.ReadLine());

int[,] M = new int[size, size];

int[] DIST = new int[size];

int r=1000; //Радиус (минимальный из эксцентриситетов вершин)

int[] exs = new int[size];

int ex = 0;

int d = 0;//Диаметр (максимальный эксцентриситет среди эксцентриситетов всех вершин)

for (int i = 1; i < size; i++)

{

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

{

M[i, j] = random.Next(10);

if (random.NextDouble() < 0.8)

{

M[i, j] = 0;

}

M[j, i] = M[i, j];

}

}

Console.WriteLine("Сгененрированная матрица:\t");

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

{

for (int j = 0; j < size; j++)

{

Console.Write($"{M[i, j]}, \t");

}

Console.WriteLine();

}

Console.WriteLine();

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

{

Console.Write("Верш. " + i + " ");

for (int j = 0; j < size; j++)

{

DIST[j] = 1000;

}

BFSD(i, M, DIST, size);

Console.Write("Расст. от верш. " + i + ": ");

for (int t = 0; t < size; t++)

{

Console.Write(DIST[t] + " ");

if (ex < DIST[t]&& DIST[t]!=1000) {

ex = DIST[t];

}

}

exs[i] = ex;

Console.Write(" exs(M)= " + exs[i]);

Console.WriteLine();

ex = 0;

}

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

{

if (exs[i] < r)

{

r = exs[i];

}

if (exs[i] > d)

{

d = exs[i];

}

}

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

if (exs[i] == d)

{

Console.Write(" Верш. " + i + " явл. периф.\n");

}

if (exs[i] == r)

{

Console.Write(" Верш. " + i + " явл. цент.\n");

}

}

if (r == 1000)

{

Console.Write(" r(M)= " + 0);

}

else

{

Console.Write(" r(M)= " + r);

}

Console.Write(" d(M)= " + d);

Console.WriteLine();

Izolir(M,size);

Concev(M, size);

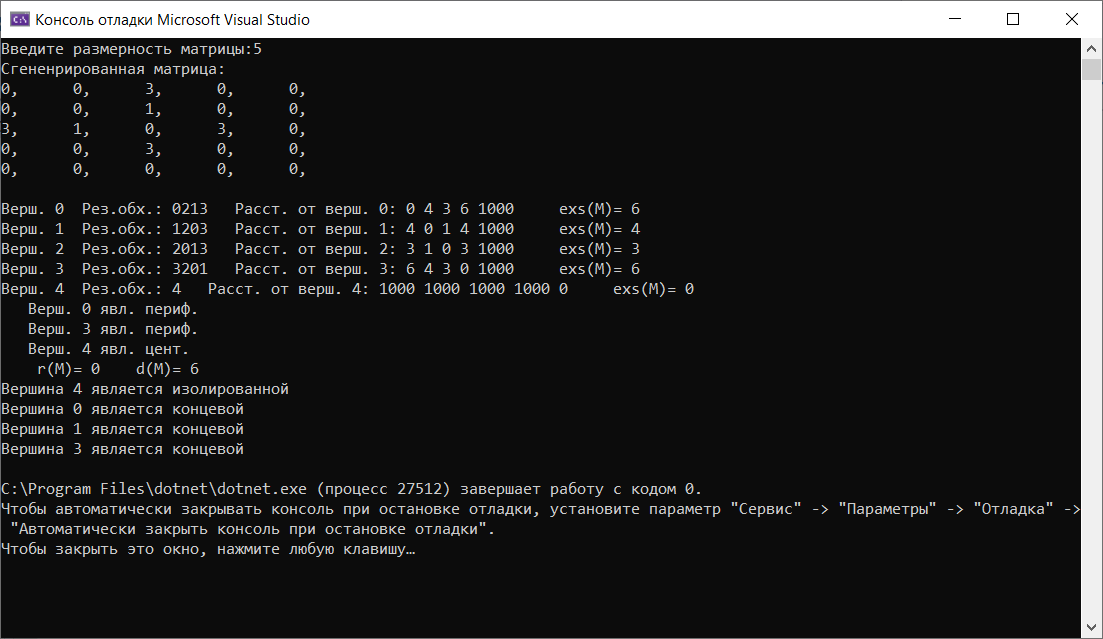
Domin(M, size);

}

}

}

}}

**Результат работы программы:**

**Вывод:** научились определять характеристики графов.