Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЁТ**

по лабораторной работе №8

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Определение характеристик графов»

Выполнили:

студенты группы 20ВВ2

Киреев Б.П.

Верховский М.В.

Лукин В.Д.

Приняли:

д.т.н. Митрохин М.А.

к.т.н. доцент Юрова О.В.

Пенза 2021

Название: Определение характеристик графов.

Цель работы: Научиться определять 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.

**Задание 2\***

1. Постройте для графа G матрицу инцидентности.

2. Определите радиус и диаметр графа G, используя матрицу

инцидентности графа.

3. Определите подмножества периферийных и центральных вершин

графа G, используя матрицу инцидентности.

**Задание 1,2:**

Листинг:

Файл Lab8Dop.cpp

#include <iostream>

#include <time.h>

#include <malloc.h>

#include <locale.h>

#include <queue>

#include <ctime>

#include <conio.h>

#include <climits> //Для предельного значения int

#include <string>

using namespace std;

int i, num = 0, pogr = 0;

int\*\* vis = NULL, n, \*\* p = NULL;

int\*\* z = NULL, \*\* vis2 = NULL, RebroScore = 0, ZapScore = 0;

//Поиск пути до всех вершин

void BFSD(int v, int\*\* p) {

for (int i = 0; i < n; i++) { vis[0][i] = INT\_MAX; }

queue <int> q;

q.push(v);

vis[0][v] = 0;

while (!q.empty()) {

v = q.front();

q.pop();

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

if (vis[0][i] > vis[0][v] + p[i][v] && p[i][v] != 0) {

q.push(i);

vis[0][i] = vis[0][v] + p[i][v];

}

}

}

cout << endl;

}

//Size2 - кол-во путей

void BFSDINCED(int v, int\*\* p, int Size1, int Size2) {

int Saver = 0;

for (int i = 0; i < n; i++) { vis2[0][i] = INT\_MAX; }

queue <int> q;

q.push(v);

vis2[0][v] = 0;

while (!q.empty()) {

v = q.front();

q.pop();

//cout << endl;

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

Saver = 0;

for (int j = 0; j < Size2; j++) {

if (z[i][j] == z[v][j] && z[i][j] != 0) { Saver = z[i][j]; }

if (i == v) { Saver = 0; }

}

//cout << " " << Saver;

if (vis2[0][i] > vis2[0][v] + Saver && Saver != 0) {

q.push(i);

vis2[0][i] = vis2[0][v] + Saver;

}

}

}

//cout << endl;

}

//Поиск эксцентрисететов вершин в матрице смежности и их степеней (Матрица, размер матрицы, матрица для записи эксцентриситета и степени)

void ExcentricitetFind(int\*\* M, int Size, int\*\* vis) {

int Exc = 0, Step = 0;

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

Exc = 0, Step = 0;

BFSD(i, p);

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

Exc = max(Exc, vis[0][j]);

if (p[i][j] != 0) { Step++; }

}

vis[1][i] = Exc;

vis[2][i] = Step;

cout << "Вершина " << i + 1 << " Эксцентриситет: " << Exc << " Степень: " << Step << endl;

}

}

//Поиск радиуса и диаметра графа // Поиск центральных и периферийных вершин

void DiametrandRadiusFind(int\* mas, int Size) {

int Diametr = 0, Radius = INT\_MAX;

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

if (mas[i] > Diametr) { Diametr = mas[i]; }

if (mas[i] < Radius) { Radius = mas[i]; }

}

cout << "Диаметр графа равен " << Diametr << endl;

cout << "Радиус графа равен " << Radius << endl;

cout << endl;

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

if (mas[i] == Diametr) { cout << "Вершина " << i + 1 << " является периферийной" << endl; }

if (mas[i] == Radius) { cout << "Вершина " << i + 1 << " является центральной" << endl; }

}

}

//Поиск эксцентрисететов вершин в матрице инцидентности и их степеней (Матрица, размер матрицы(вершины), размер матрицы(пути), матрица для записи эксцентриситета и степени)

void ExcentricitetFindINCED(int\*\* M, int Size1, int Size2, int\*\* visX) {

int Exc = 0, Step = 0;

for (int i = 0; i < Size1; i++) {

Exc = 0, Step = 0;

BFSDINCED(i, z, n, RebroScore);

for (int j = 0; j < Size1; j++) {

Exc = max(Exc, vis2[0][j]);

}

Exc = vis[1][i];

cout << "Вершина " << i + 1 << " Эксцентриситет: " << Exc << endl;

}

}

int main() {

setlocale(LC\_ALL, "Rus");

srand(time(0));

cout << "Введите размерность графа:";

cin >> n;

cout.precision(5 \* n);

p = (int\*\*)calloc(n, sizeof(int));//Выделяем место под матрицу смежности

z = (int\*\*)calloc(n, sizeof(int)); //Выделяем место под матрицу инцидентности

vis = (int\*\*)calloc(3, sizeof(int)); //Выделяем место под массив с данными о вершинах(Смежность)

vis2 = (int\*\*)calloc(3, sizeof(int)); //Выделяем место под массив с данными о вершинах(Инцидентность)

for (i = 0; i < n; i++) { p[i] = (int\*)calloc(n, sizeof(int)); }

for (i = 0; i < 3; i++) {

vis[i] = (int\*)calloc(n, sizeof(int));

vis2[i] = (int\*)calloc(n, sizeof(int));

}

for (int i = 0; i < n; i++) { //Генерация путей и их веса

for (int m = pogr; m < n; m++) {

p[i][m] = 0 + rand() % 11;

p[m][i] = p[i][m];

p[i][i] = 0;

if (p[i][m] != 0) { RebroScore++; } //Считаем количество рёбер в графе

}

pogr++;

}

for (i = 0; i < n; i++) { z[i] = (int\*)calloc(RebroScore, sizeof(int)); } //Создаем в первичном массиве вторичный, получая матрицу (n X RebroScore)

pogr = 0;

for (int i = 0; i < n; i++) { //Заполняем матрицу инцидентности по матрице смежности

for (int m = pogr; m < n; m++) {

if (p[i][m] != 0) {

z[i][ZapScore] = p[i][m];

z[m][ZapScore] = p[i][m];

ZapScore++;

}

}

pogr++;

}

//Для матрицы смежности//Задание 1//

cout << "\nМатрица смежности:" << endl; //Вывод матрицы

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

for (int m = 0; m < n; m++) { cout << p[i][m] << " "; }

cout << endl;

}

cout << endl;

ExcentricitetFind(p, n, vis);

cout << endl;

DiametrandRadiusFind(vis[1], n); //Анализируем массив с эксцентриситетами на радиус и диаметр графа

cout << endl;

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

if (vis[2][i] == 0) { cout << "Вершина " << i + 1 << " является изолированной" << endl; }

if (vis[2][i] == 1) { cout << "Вершина " << i + 1 << " является концевой" << endl; }

if (vis[2][i] == n - 1) { cout << "Вершина " << i + 1 << " является доминирующей" << endl; }

}

////

//Для матрицы инцидентности//Задание 2//

cout << "\nМатрица инцедентности:" << endl; //Вывод матрицы

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

for (int m = 0; m < RebroScore; m++) { cout << z[i][m] << " "; }

cout << endl;

}

cout << endl;

ExcentricitetFindINCED(z, n, RebroScore, vis2);

cout << endl;

DiametrandRadiusFind(vis[1], n);

cout << endl;

\_getch();

return 0;

}

Пояснительный текст к программе:

Данная программа определяет радиус и диаметр взвешенного неориентированного графа, определяет подмножества периферийных и центральных вершин графа, используя матрицу смежности, находит изолированные, концевые и доминирующие вершины. Определяет радиус и диаметр графа, подмножества периферийных и центральных вершин графа, используя матрицу инцидентности графа.

Результаты работы программы:

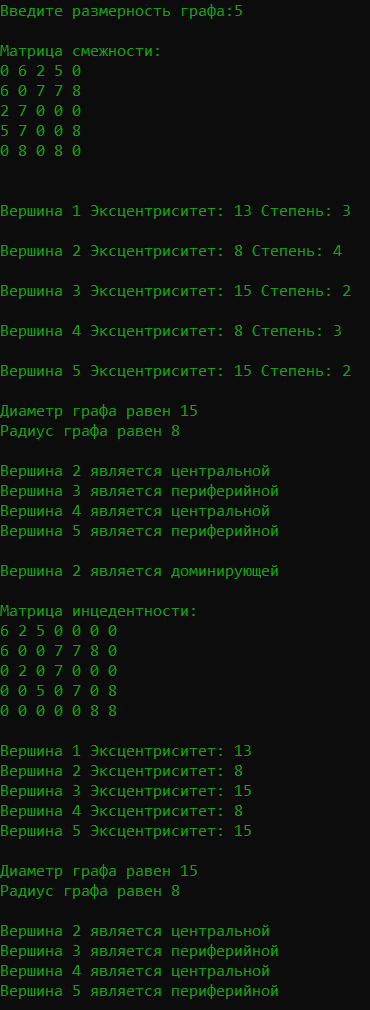


Рисунок №1 – Результат работы программы.

Вывод: При выполнении данной лабораторной работы мы определили характеристики графов.