Министерство образование Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №8

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Определение характеристик графов»

Выполнила:

студентка группы 20ВВ1:

Зацепилина Е.С.

Приняли:

Акифьев И.В.

Юрова О.В.

Пенза 2021 г.

**Цель работы**: освоить методы определения основных характеристик графа в его различных представлениях.

**Задание на лабораторную работу:**

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.

**Ход выполнения задания:**

Для выполнения задания на лабораторную работу были написана программа 8 на языке С++ в среде Visual Studio 2019. Полный текст программы смотрите в п. «Листинг».

Генерация матрицы смежности:

for (int i = 0; i < m1.size(); i++) {

for (int j = 0; j < m1[i].size(); j++) {

if (i == j) {

m1[i][j] = 0;

break;

}

else {

int z = rand() % 101;

if (z < 30) { m1[i][j] = 0; }

else { m1[i][j] = z; }

m1[j][i] = m1[i][j];

}

}

}

Генерация матрицы инцидентности:

kol = 0;

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

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

if (i == j) break;

else {

if (m1[i][j] > 0) kol++;

}

}

}

m2.resize(n);

for (int i = 0; i < m2.size(); i++) {

m2.at(i).resize(kol);

}

kol = 0;

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

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

if (i == j) break;

else {

if (m1[i][j] > 0) {

m2[i][kol] = m1[i][j];

m2[j][kol] = m1[i][j];

kol++;

}

}

}

}

Поиск расстояний в матрице смежности:

void BFSD(vector <vector <int>> gr, int v) {

queue <unsigned int> q = {};

q.push(v);

int vc = v;

DIST[vc][v] = 0;

while (!q.empty()) {

v = q.front();

q.pop();

for (unsigned int i = 0; i < gr[v].size(); i++) {

if (gr[v][i] > 0 && DIST[vc][i] > DIST[vc][v] + gr[v][i]) {

q.push(i);

DIST[vc][i] = DIST[vc][v] + gr[v][i];

}

}

}

}

Поиск расстояний в матрице инцидентности:

void DI(vector <vector <int>> gr, int v) {

queue <unsigned int> q = {};

q.push(v);

int vc = v;

DIST\_inc[vc][v] = 0;

while (!q.empty()) {

v = q.front();

q.pop();

vector <int> subv(gr.size());

for (unsigned int i = 0; i < subv.size(); i++) subv[i] = 0;

for (unsigned int i = 0; i < gr[v].size(); i++) {

if (gr[v][i] > 0) {

for (unsigned int j = 0; j < gr.size(); j++) {

if (gr[j][i] > 0) {

subv[j] = gr[j][i];

}

}

}

}

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

if (subv[i] > 0 && DIST\_inc[vc][i] > DIST\_inc[vc][v] + subv[i]) {

q.push(i);

DIST\_inc[vc][i] = DIST\_inc[vc][v] + subv[i];

}

}

}

}

Поиск основных характеристик на основе сформированной матрицы DIST (аналогично для DIST\_inc):

printf("\nЭксцентриситеты в графе: ");

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

printf("%-4d", exc[i]);

}

printf("\n");

int min = exc[0];

int max = exc[0];

for (int i = 1; i < exc.size(); i++) {

if (exc[i] < min) min = exc[i];

if (exc[i] > max) max = exc[i];

}

printf("\nРадиус: %d\n", min);

printf("\nДиаметр: %d\n", max);

printf("\nПереферийные вершины графа: ");

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

if (exc[i] == min) printf("%-2d", i + 1);

}

printf("\n");

printf("\nЦентральные вершины графа: ");

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

if (exc[i] == max) printf("%-2d", i + 1);

}

printf("\n");

int kol;

printf("\nИзолированные вершины графа: ");

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

kol = 0;

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

if (m1[i][j] > 0) kol++;

}

if (kol == 0) printf("%-2d", i + 1);

}

printf("-\n");

printf("\nКонцевые вершины графа: ");

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

kol = 0;

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

if (m1[i][j] > 0) kol++;

}

if (kol == 1) printf("%-2d", i + 1);

}

printf("-\n");

printf("\nДоминирующие вершины графа: ");

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

kol = 0;

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

if (m1[i][j] > 0) kol++;

}

if (kol == n - 1) printf("%-2d", i + 1);

}

printf("-\n");

**Вывод:** в результате выполнения лабораторной работы были получены навыки определения основных характеристик графа.

**Листинг:**

**8.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <locale.h>

#include <windows.h>

#include <vector>

#include <queue>

using namespace std;

vector <vector<int>> DIST;

vector <vector<int>> DIST\_inc;

void vivod\_m(vector <vector<int>> arr) {//Функция вывода одной матрицы

cout << (" |");

for (int i = 0; i < arr[0].size(); i++) {

printf("%-4d|", i + 1);

}

cout << ("\n");

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

printf("|");

printf("%-2d|", i + 1);

for (int j = 0; j < arr[i].size(); j++) {

printf("%-4d|", arr[i][j]);

}

printf("\n");

}

}

void BFSD(vector <vector <int>> gr, int v) {

queue <unsigned int> q = {};

q.push(v);

int vc = v;

DIST[vc][v] = 0;

while (!q.empty()) {

v = q.front();

q.pop();

for (unsigned int i = 0; i < gr[v].size(); i++) {

if (gr[v][i] > 0 && DIST[vc][i] > DIST[vc][v] + gr[v][i]) {

q.push(i);

DIST[vc][i] = DIST[vc][v] + gr[v][i];

}

}

}

}

void DI(vector <vector <int>> gr, int v) {

queue <unsigned int> q = {};

q.push(v);

int vc = v;

DIST\_inc[vc][v] = 0;

while (!q.empty()) {

v = q.front();

q.pop();

vector <int> subv(gr.size());

for (unsigned int i = 0; i < subv.size(); i++) subv[i] = 0;

for (unsigned int i = 0; i < gr[v].size(); i++) {

if (gr[v][i] > 0) {

for (unsigned int j = 0; j < gr.size(); j++) {

if (gr[j][i] > 0) {

subv[j] = gr[j][i];

}

}

}

}

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

if (subv[i] > 0 && DIST\_inc[vc][i] > DIST\_inc[vc][v] + subv[i]) {

q.push(i);

DIST\_inc[vc][i] = DIST\_inc[vc][v] + subv[i];

}

}

}

}

int main(void)

{

char\* locale = setlocale(LC\_ALL, "");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

COORD coord;

SetConsoleDisplayMode(GetStdHandle(STD\_OUTPUT\_HANDLE), CONSOLE\_FULLSCREEN\_MODE, &coord);

vector < vector <int> > m1, m2;

int n;

printf("Введите кол-во вершин: ");

cin >> n;

m1.resize(n);

for (int i = 0; i < m1.size(); i++) {

m1.at(i).resize(n);

}

DIST.resize(n);

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

DIST.at(i).resize(n);

}

srand(time(NULL));

for (int i = 0; i < m1.size(); i++) {

for (int j = 0; j < m1[i].size(); j++) {

if (i == j) {

m1[i][j] = 0;

break;

}

else {

int z = rand() % 101;

if (z < 30) { m1[i][j] = 0; }

else { m1[i][j] = z; }

m1[j][i] = m1[i][j];

}

}

}

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

for (int j = 0; j < DIST[i].size(); j++) {

DIST[i][j] = INT\_MAX;

}

}

printf("\nМатрица смежности графа\n\n");

vivod\_m(m1);

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

BFSD(m1, i);

}

printf("\nРасстояния в графе\n\n");

vivod\_m(DIST);

vector <int> exc(n);

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

exc[i] = 0;

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

if (exc[i] < DIST[i][j]) exc[i] = DIST[i][j];

}

}

printf("\nЭксцентриситеты в графе: ");

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

printf("%-4d", exc[i]);

}

printf("\n");

int min = exc[0];

int max = exc[0];

for (int i = 1; i < exc.size(); i++) {

if (exc[i] < min) min = exc[i];

if (exc[i] > max) max = exc[i];

}

printf("\nРадиус: %d\n", min);

printf("\nДиаметр: %d\n", max);

printf("\nПереферийные вершины графа: ");

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

if (exc[i] == min) printf("%-2d", i + 1);

}

printf("\n");

printf("\nЦентральные вершины графа: ");

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

if (exc[i] == max) printf("%-2d", i + 1);

}

printf("\n");

int kol;

printf("\nИзолированные вершины графа: ");

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

kol = 0;

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

if (m1[i][j] > 0) kol++;

}

if (kol == 0) printf("%-2d", i + 1);

}

printf("-\n");

printf("\nКонцевые вершины графа: ");

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

kol = 0;

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

if (m1[i][j] > 0) kol++;

}

if (kol == 1) printf("%-2d", i + 1);

}

printf("-\n");

printf("\nДоминирующие вершины графа: ");

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

kol = 0;

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

if (m1[i][j] > 0) kol++;

}

if (kol == n - 1) printf("%-2d", i + 1);

}

printf("-\n");

kol = 0;

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

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

if (i == j) break;

else {

if (m1[i][j] > 0) kol++;

}

}

}

m2.resize(n);

for (int i = 0; i < m2.size(); i++) {

m2.at(i).resize(kol);

}

kol = 0;

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

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

if (i == j) break;

else {

if (m1[i][j] > 0) {

m2[i][kol] = m1[i][j];

m2[j][kol] = m1[i][j];

kol++;

}

}

}

}

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

printf("\nМатрица инцидентности графа\n\n");

vivod\_m(m2);

DIST\_inc.resize(n);

for (int i = 0; i < DIST\_inc.size(); i++) {

DIST\_inc.at(i).resize(n);

}

for (int i = 0; i < DIST\_inc.size(); i++) {

for (int j = 0; j < DIST\_inc[i].size(); j++) {

DIST\_inc[i][j] = INT\_MAX;

}

}

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

DI(m2, i);

}

printf("\nРасстояния в графе\n\n");

vivod\_m(DIST\_inc);

vector <int> exc\_inc(n);

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

exc\_inc[i] = 0;

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

if (exc\_inc[i] < DIST\_inc[i][j]) exc\_inc[i] = DIST\_inc[i][j];

}

}

printf("\nЭксцентриситеты в графе: ");

for (int i = 0; i < exc\_inc.size(); i++) {

printf("%-4d", exc\_inc[i]);

}

printf("\n");

min = exc\_inc[0];

max = exc\_inc[0];

for (int i = 1; i < exc.size(); i++) {

if (exc\_inc[i] < min) min = exc\_inc[i];

if (exc\_inc[i] > max) max = exc\_inc[i];

}

printf("\nРадиус: %d\n", min);

printf("\nДиаметр: %d\n", max);

printf("\nПереферийные вершины графа: ");

for (int i = 0; i < exc\_inc.size(); i++) {

if (exc\_inc[i] == min) printf("%-2d", i + 1);

}

printf("\n");

printf("\nЦентральные вершины графа: ");

for (int i = 0; i < exc\_inc.size(); i++) {

if (exc\_inc[i] == max) printf("%-2d", i + 1);

}

printf("\n");

printf("\n");

system("pause");

return 0;

}

**Результаты:**



Рисунок 1 – Результаты работы программы

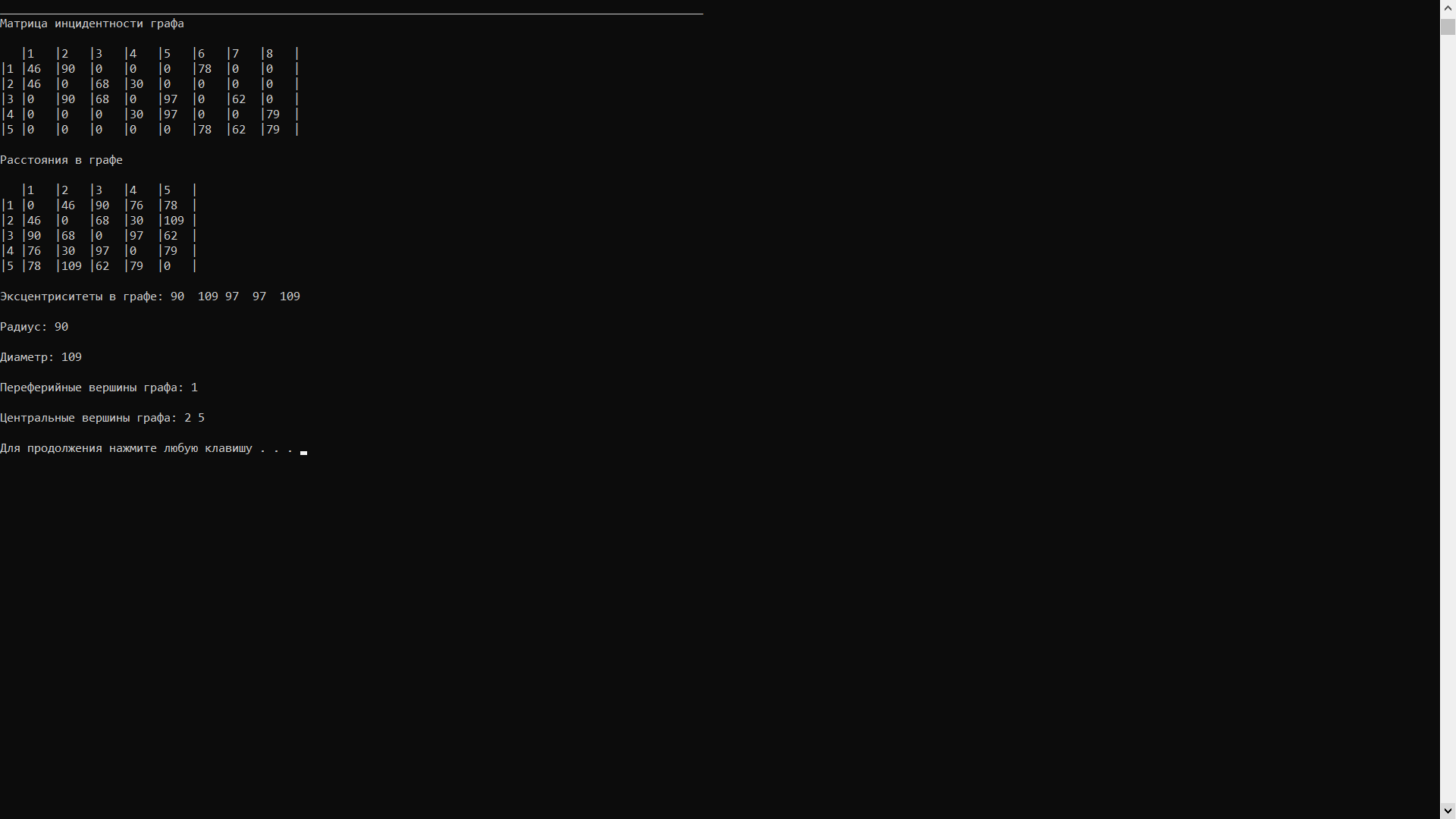


Рисунок 2 – Результаты работы программы