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»

**Выполнили:**

студент группы 20ВВ3

Шмелёв Д.В.

**Приняли:**

Юрова О.В.

Митрохин М.А.

Пенза 2021

# Название

Определение характеристик графов

**Цель работы** – изучение характеристик графа и алгоритмов их определения

# Методические указания

**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 e(v i ) = d(G).

Минимальный из эксцентриситетов вершин графа называется его

радиусом и обозначается через r(G).

Вершина v i называется центральной, если её эксцентриситет равен

радиусу графа e(v i ) = r(G).

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.

# 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.

# Листинг

#define \_CRT\_SECURE\_NO\_WARNINGS

#define \_CRT\_NONSTDC\_NO\_WARNINGS

#define HEADER ("Лабораторная работа №8\nВыполнили: Шмелёв Д. и Пантюшов Е.\n\n")

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

#include <limits.h>

#include <queue>

using namespace std;

int\*\* createMatrix1(int n)

{

int\*\* M = (int\*\*)(malloc(n \* sizeof(int\*)));

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

{

M[i] = (int\*)(malloc(n \* sizeof(int)));

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

{

if (rand() % 100 < 50 || i == j)

{

M[i][j] = 0;

}

else

{

M[i][j] = rand() % n;

}

}

}

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

{

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

{

if (M[i][j] > 0)

{

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

}

}

}

return M;

}

int createEdge(int\*\* M1, int n)

{

int m = 0;

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

{

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

{

if (M1[i][j] > 0)

{

m = m + 1;

}

}

}

m = m / 2;

return m;

}

int\*\* createMatrix2(int\*\* M1, int n, int m)

{

int\*\* M = (int\*\*)(malloc(n \* sizeof(int\*)));

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

{

M[i] = (int\*)(malloc(m \* sizeof(int)));

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

{

M[i][j] = 0;

}

}

int k = 0;

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

{

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

{

if (M1[i][j] > 0 && i < j)

{

M[i][k] = M1[i][j];

M[j][k] = M1[i][j];

k++;

}

}

}

return M;

}

void printMatrix1(int\*\* M, int n)

{

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

{

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

{

printf(" %d ", M[i][j]);

}

printf("\n");

}

}

void printMatrix2(int\*\* M, int n, int m)

{

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

{

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

{

printf(" %d ", M[i][j]);

}

printf("\n");

}

}

void BFSD1(int\*\* M, int n, int\* dist, int x)

{

queue <int> Q;

Q.push(x);

dist[x] = 0;

while (!Q.empty())

{

x = Q.front();

Q.pop();

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

{

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

{

if (M[x][i] > 0 && dist[i] > dist[x] + M[x][i])

{

Q.push(i);

dist[i] = dist[x] + M[x][i];

}

}

}

}

}

void BFSD2(int\*\* M, int n, int m, int\* dist, int x)

{

queue <int> Q;

Q.push(x);

dist[x] = 0;

while (!Q.empty())

{

x = Q.front();

Q.pop();

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

{

if (M[x][i] > 0)

{

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

{

if (M[j][i] > 0 && dist[j] > dist[x] + M[j][i])

{

Q.push(j);

dist[j] = dist[x] + M[j][i];

}

}

}

}

}

}

void task\_1(int\*\* M, int n)

{

printf("Задание 1.\n\nПункт 1.\nМатрица смежности:\n");

printMatrix1(M, n);

int\* dist = (int\*)(malloc(n \* sizeof(int)));

int\*\* Mdist = (int\*\*)(malloc(n \* sizeof(int\*)));

int R = SHRT\_MAX, D = SHRT\_MIN;

printf("\nПункт 2.\nКратчайшие расстояния всех вершин:");

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

{

Mdist[i] = (int\*)(malloc(n \* sizeof(int)));

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

dist[j] = SHRT\_MAX;

BFSD1(M, n, dist, i);

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

{

if (dist[j] == SHRT\_MAX)

printf("\n%d -> %d: нет пути!", i, j);

else

printf("\n%d -> %d: %d", i, j, dist[j]);

if (dist[j] > D && dist[j] != SHRT\_MAX && j != i)

D = dist[j];

if (dist[j] < R && dist[j] != SHRT\_MAX && j != i)

R = dist[j];

Mdist[i][j] = dist[j];

}

}

printf("\n\nD = %d\nR = %d\n\n", D, R);

printf("Пункт 3.\nПодмножества периферийных вершин:\n");

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

{

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

{

if (Mdist[i][j] == D && i < j)

{

printf("%d <-> %d\n", i, j);

}

}

}

printf("\nПодмножества центральных вершин:\n");

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

{

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

{

if (Mdist[i][j] == R && i < j)

{

printf("%d <-> %d\n", i, j);

}

}

}

printf("\nПункт 4.\nИзолированные вершины:");

int\* s = (int\*)(malloc(n \* sizeof(int)));

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

{

s[i] = 0;

}

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

{

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

{

if (M[i][j] > 0)

{

s[i] = s[i]++;

}

}

}

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

{

if (s[i] == 0)

{

printf(" %d ", i);

}

}

printf("\nКонцевые вершины:");

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

{

if (s[i] == 1)

{

printf(" %d ", i);

}

}

printf("\nДоминирующие вершины:");

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

{

if (s[i] == n - 1)

{

printf(" %d ", i);

}

}

}

void task\_2(int\*\* M, int n, int m)

{

printf("\n\nЗадание 2.\n\nПункт 1.\nМатрица инцидентности:\n");

printMatrix2(M, n, m);

int\* dist = (int\*)(malloc(n \* sizeof(int)));

int\*\* Mdist = (int\*\*)(malloc(n \* sizeof(int\*)));

int R = SHRT\_MAX, D = SHRT\_MIN;

printf("\nПункт 2.\nКратчайшие расстояния всех вершин:\n");

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

{

Mdist[i] = (int\*)(malloc(n \* sizeof(int)));

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

dist[j] = SHRT\_MAX;

BFSD2(M, n, m, dist, i);

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

{

if (dist[j] == SHRT\_MAX)

printf("\n%d -> %d: нет пути!", i, j);

else

printf("\n%d -> %d: %d", i, j, dist[j]);

if (dist[j] > D && dist[j] != SHRT\_MAX && j != i)

D = dist[j];

if (dist[j] < R && dist[j] != SHRT\_MAX && j != i)

R = dist[j];

Mdist[i][j] = dist[j];

}

}

printf("\n\nD = %d\nR = %d\n\n", D, R);

printf("Пункт 3.\nПодмножества периферийных вершин:\n");

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

{

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

{

if (Mdist[i][j] == D && i < j)

{

printf("%d <-> %d\n", i, j);

}

}

}

printf("\nПодмножества центральных вершин:\n");for (int i = 0; i < n; i++)

{

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

{

if (Mdist[i][j] == R && i < j)

{

printf("%d <-> %d\n", i, j);

}

}

}

}

int main()

{

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

printf(HEADER);

int n;

printf("Введите размерность матрицы: ");

scanf("%d", &n);

int\*\* M1 = createMatrix1(n);

task\_1(M1, n);

int m = createEdge(M1, n);

int\*\* M2 = createMatrix2(M1, n, m);

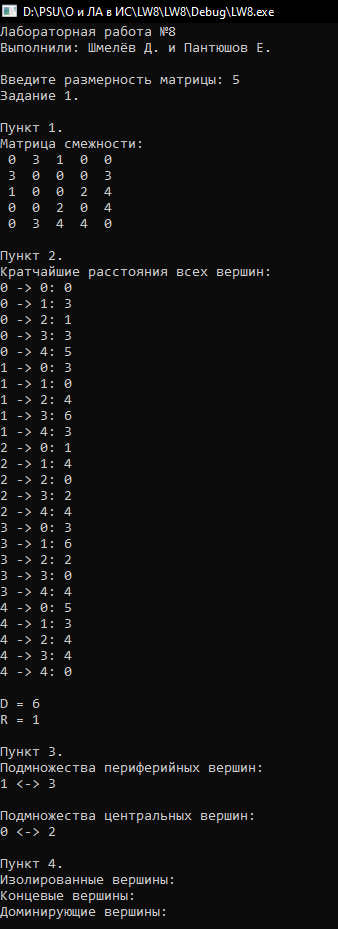
task\_2(M2, n, m);

system("PAUSE");

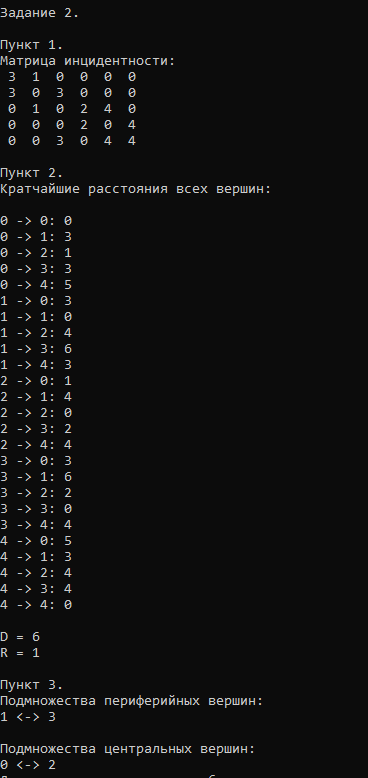
return 0;

}

# Результат работы программы



**Рисунок 1– Результат работы программы (задание 1)**



**Рисунок 2 - Результат работы программы (задание 2)**

# Вывод

# В ходе выполнения лабораторной работы была разработана программа, в который были реализованы алгоритмы нахождения характеристик графа.