Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе №8

по курсу «Логика и основы алгоритмизации в ИС»

на тему «Определение характеристик графов»

Выполнили

студенты группы 20ВВ2:

Тумасов Вадим

Портнов Никита

Принял:

д.т.н Митрохин М.А.

д.т.н Юрова О.В.

Пенза 2021

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

**Доп задание\*:**

1.Вывести степень каждой вершины

2.Найти изолированные,концевые и доминирующие вершины,используя матрицу инцидентности.

**Листинг:**

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <cstdlib>

#include <windows.h>

#include <conio.h>

#include <ctime>

#include <cmath>

#include <stack>

#include <queue>

using namespace std;

int max\_size = 5, i, j, start, num, mass\_ex[30], D = 0, R = 1000, perif, center, elem = 0, counter\_edge = 0, all\_edge = 0, deg, stepen;

int\* DIST = (int\*)malloc(max\_size \* sizeof(int));

int\* DIST2 = (int\*)malloc(max\_size \* sizeof(int));

queue<int> ex;

int st\_rebro;

int z[100];

int counter\_rebro, schetchik, schetchik2;

void BSFD(int st, int\*\* b, int num\_of\_elem, int\* vis) {

queue<int> Q;

int t;

Q.push(st);

vis[st] = 0;

while (!Q.empty())

{

t = Q.front();

Q.pop();

for (int i = 0; i < num\_of\_elem; i++)

{

if (b[t][i] > 0 && vis[i] == 1000)

{

vis[i] = vis[t] + b[t][i];

Q.push(i);

}

}

}

}

void BSFD\_rebro(int st\_rebro, int\*\* b, int num\_edge, int num\_height, int\* vis) {

queue<int> Q;

int t;

Q.push(st\_rebro);

vis[st\_rebro] = 0;

while (!Q.empty())

{

t = Q.front();

Q.pop();

for (int i = 0; i < num\_edge; i++)

{

if (b[t][i] > 0 && vis[t] == 1000)

{

for (int k = 0; k < num\_height; k++)

{

if (b[k][i] > 0 && vis[k] == 1000)

{

vis[k] = vis[k] + b[k][i];

Q.push(k);

}

}

}

}

}

}

void max\_elem(int\* mass) {

int max = 0, n;

for (n = 0; n < max\_size; n++) {

if (mass[n] > max && mass[n] != 1000)

max = mass[n];

}

ex.push(max);

}

int main() {

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

int\*\* a = new int\* [max\_size];

for (i = 0; i < max\_size; i++)

{

a[i] = new int[max\_size];

}

for (i = 0; i < max\_size; i++)

{

for (j = i + 1; j < max\_size; j++)

{

num = rand() % 8;

if (num <= 4)

{

a[i][j] = 0;

a[j][i] = a[i][j];

}

else

{

a[i][j] = (rand() % 6) + 1;

a[j][i] = a[i][j];

}

}

}

printf("\nНеориентированный граф:\n");

for (i = 0; i < max\_size; i++)

{

DIST[i] = 1000;

for (j = 0; j < max\_size; j++)

{

if (i == j)

a[i][j] = 0;

printf("%2d", a[i][j]);

}

printf("\n");

}

for (i = 0; i < max\_size; i++) {

BSFD(i, a, max\_size, DIST);

max\_elem(DIST);

printf("\nРасстояния из %d-й вершины: ", i + 1);

for (j = 0; j < max\_size; j++) {

printf("%4d ", DIST[j]);

DIST[j] = 1000;

}

}

printf("\n\nЭксицентритеты вершин:\n");

for (i = 0; i < max\_size; i++) {

if (!ex.empty()) {

mass\_ex[i] = ex.front();

printf("%d-й = %d\n", i + 1, mass\_ex[i]);

ex.pop();

}

}

for (i = 0; i < max\_size; i++) {

if (mass\_ex[i] > D)

D = mass\_ex[i];

if (mass\_ex[i] < R)

R = mass\_ex[i];

}

if (D > 0)

{

printf("Диаметр графа = %d\n", D);

}

if (R > 0)

{

printf("Радиус графа = %d", R);

}

printf("\nПерифирийные вершины: ");

for (i = 0; i < max\_size; i++) {

if (mass\_ex[i] == D)

printf("%d ", i + 1);

}

printf("\nЦентральные вершины: ");

for (i = 0; i < max\_size; i++) {

if (mass\_ex[i] == R && R != 0)

printf("%d ", i + 1);

}

//изолированные, концевые и доминирующие вершины

printf("\n\nКонцевые вершины:");

for (i = 0; i < max\_size; i++)

{

for (j = 0; j < max\_size; j++)

{

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

elem++;

}

if (elem == 1)

printf("%d ", i + 1);

elem = 0;

}

printf("\nИзолированные вершины:");

for (i = 0; i < max\_size; i++)

{

for (j = 0; j < max\_size; j++)

{

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

{

elem++;

}

}

if (elem == 0)

printf("%d ", i + 1);

elem = 0;

}

printf("\nДоминирующие вершины:");

for (i = 0; i < max\_size; i++)

{

for (j = 0; j < max\_size; j++)

{

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

{

elem++;

}

}

if (elem == max\_size - 1)

printf("%d ", i + 1);

elem = 0;

}

cout << endl;

cout << endl;

cout << endl;

cout << "Матрица инцидентности:" << endl;

int q, w;

counter\_edge = 0;

all\_edge = 0;

for (q = 0; q < max\_size; q++)

{

for (w = 0; w < (max\_size + q - max\_size + 1); w++)

{

if (a[q][w] > 0 && a[q][w] != 1000)

{

all\_edge++;

}

}

}

int\*\* x = new int\* [max\_size];

for (i = 0; i < max\_size; i++)

{

x[i] = new int[all\_edge];

}

for (i = 0; i < max\_size; i++)

{

for (j = 0; j < all\_edge; j++)

{

x[i][j] = 0;

}

}

for (q = 0; q < max\_size; q++)

{

for (w = q; w < max\_size; w++)

{

if (a[q][w] > 0 && a[q][w] != 1000)

{

x[q][counter\_edge] = a[q][w];

x[w][counter\_edge] = a[q][w];

counter\_edge++;

}

}

}

for (q = 0; q < max\_size; q++)

{

for (w = 0; w < all\_edge; w++)

{

printf("%2d", x[q][w]);

}

printf("\n");

}

for (i = 0; i < max\_size; i++)

{

DIST2[i] = 1000;

}

for (i = 0; i < max\_size; i++) {

BSFD(i, a, max\_size, DIST2);

max\_elem(DIST2);

printf("\nРасстояния из %d-й вершины: ", i + 1);

for (j = 0; j < max\_size; j++) {

printf("%4d ", DIST2[j]);

DIST2[j] = 1000;

}

}

printf("\n\nЭксицентритеты вершин:\n");

for (i = 0; i < max\_size; i++) {

if (!ex.empty()) {

mass\_ex[i] = ex.front();

printf("%d-й = %d\n", i + 1, mass\_ex[i]);

ex.pop();

}

}

for (i = 0; i < max\_size; i++) {

if (mass\_ex[i] > D)

D = mass\_ex[i];

if (mass\_ex[i] < R)

R = mass\_ex[i];

}

if (D > 0)

{

printf("Диаметр графа = %d\n", D);

}

if (R > 0)

{

printf("Радиус графа = %d", R);

}

//printf("Диаметр графа = %d\nРадиус графа = %d", D, R);

cout << endl << endl;

printf("\nПерифирийные вершины: ");

for (i = 0; i < max\_size; i++) {

if (mass\_ex[i] == D)

printf("%d ", i + 1);

}

printf("\nЦентральные вершины: ");

for (i = 0; i < max\_size; i++) {

if (mass\_ex[i] == R && R != 0)

printf("%d ", i + 1);

}

for (q = 0; q < max\_size; q++)

{

for (w = 0; w < all\_edge; w++)

{

if (x[q][w] > 0)

{

stepen++;

}

}

z[q] = stepen;

printf("\n");

cout << "Степень вершины" << " " << q << " " << "равна";

printf("%2d", stepen);

stepen = 0;

}

printf("\n\nКонцевые вершины:");

for (q = 0; q < max\_size; q++)

{

for (w = 0; w < all\_edge; w++)

{

if (x[q][w] > 0)

elem++;

}

if (elem == 1)

printf("%d ", q + 1);

elem = 0;

}

printf("\nИзолированные вершины:");

for (q = 0; q < max\_size; q++)

{

for (w = 0; w < all\_edge; w++)

{

if (x[q][w] > 0)

{

elem++;

}

}

if (elem == 0)

printf("%d ", q + 1);

elem = 0;

}

printf("\nДоминирующие вершины:");

for (q = 0; q < max\_size; q++)

{

for (w = 0; w < all\_edge; w++)

{

if (x[q][w] > 0)

{

elem++;

}

}

if (elem == max\_size - 1)

printf("%d ", q + 1);

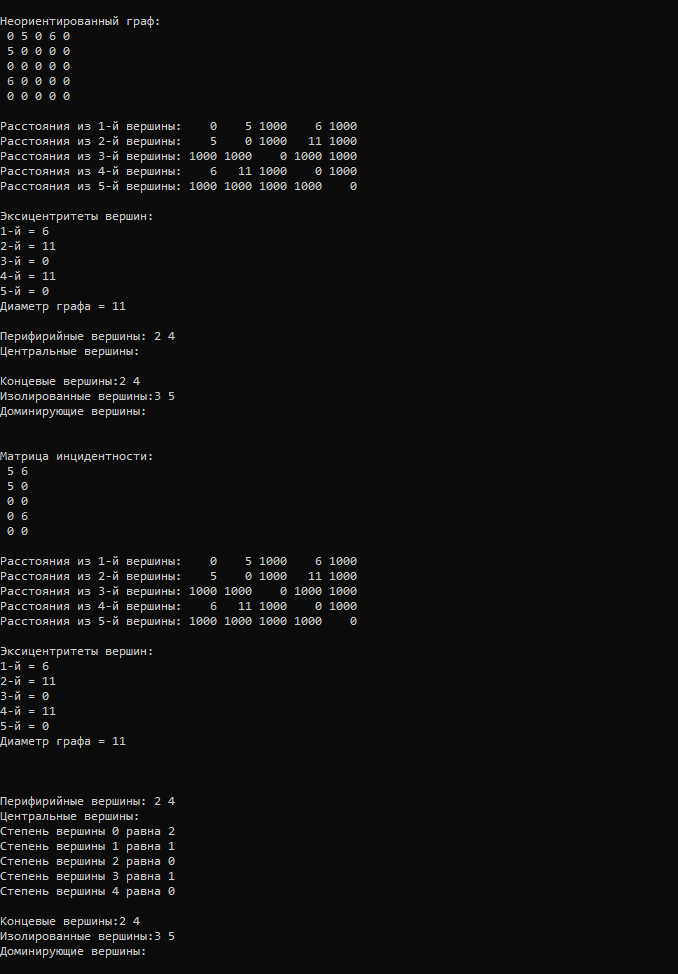
elem = 0;

}

\_getch();

}

**Результат работы программы:**



**Вывод**:в ходе выполнения данной лабораторной работы мы углубили свои знания о графах,познали теоретический материал об эксцентриситетах вершин,диаметре,радиусе графа и также разновидностях вершин(перифирийные,центральные,концевые,изолированные,доминирующие) и реализовали полученную информацию в нашей программе.Также научились организовывать матрицу инцидентности и работать с ней.