Пензенский государственный университет  
Кафедра «Вычислительной техники»

**Отчет**по лабораторной работе №8  
по дисциплине: «Логика и основы алгоритмизации в инженерных задачах»  
на тему: «Определение характеристик графов»

**Выполнил студент группы 19ВВ1:**

Балалаев А.А.

**Приняли:**

д.т.н. профессор Митрохин М. А.

к.т.н. Юрова О.В.

Пенза 2020.

**Цель работы:** научится находить эксцентриситет вершины, определять радиус и диаметр графа, находить изолированные, концевые и доминирующие вершины.

**Теория**

Если G - граф, содержащий непустое множество n вершин V и

множество ребер E и d(v i , v j ) – расстояние между двумя произвольными

вершинами v i  и v j , тогда для фиксированной вершины v величина



где v, v j V и j = 1…n называется эксцентриситетом вершины v i .

Другими словами 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.

**Код программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <time.h>

#include <iostream>

#include <locale.h>

#include <queue>

#include <malloc.h>

#include <windows.h>

using namespace std;

void rand\_Zap\_vz\_nor(int\*\* Matrix, int n)

{

srand(time(NULL));

printf("Взвешаный неорентированный граф\n");

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

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

{

if (i == j)

{

Matrix[i][j] = 0;

}

if (i < j) {

if (rand() % 100 > 50)

Matrix[i][j] = rand() % 10;

else

Matrix[i][j] = 0;

Matrix[j][i] = Matrix[i][j];

}

}

}

void dsit\_0(int\* Matrix, int n)

{

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

Matrix[i] = 65536;

}

void BFSD\_vz(int\*\* Matrix, int\* dist, int n, int v)

{

queue <int> Q;

Q.push(v);

dist[v] = 0;

while (Q.empty() == false)

{

v = Q.front();

Q.pop();

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

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

{

if ((Matrix[v][i] != 0) && (dist[i] > (dist[v] + Matrix[v][i])))

{

Q.push(i);

dist[i] = dist[v] + Matrix[v][i];

}

}

}

printf("\n");

}

void print\_G(int\*\* Matrix, int n)

{

printf(" ");

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

printf("%3d", i + 1);

printf("\n\n");

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

{

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

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

{

printf("%3d", Matrix[i][j]);

}

printf("\n");

}

printf("\n");

}

int main()

{

setlocale(LC\_ALL, "Rus");

int max = 0, min;

int N = 1;

scanf("%d", &N);

int\*\* G = (int\*\*)malloc(N \* sizeof(int\*));

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

{

G[i] = (int\*)malloc(N \* sizeof(int));

}

int\* DIST, \*eks, \*step;

DIST = (int\*)malloc(N \* sizeof(int));

eks = (int\*)malloc(N \* sizeof(int));

step = (int\*)malloc(N \* sizeof(int));

rand\_Zap\_vz\_nor(G, N);

print\_G(G, N);

dsit\_0(DIST, N);

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

{

max = 0;

dsit\_0(DIST, N);

printf("от вершины %d\n", i + 1);

BFSD\_vz(G, DIST, N, i);

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

{

if (DIST[j] != 65536)

{

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

if (DIST[j] > max)

max = DIST[j];

}

else

printf("нет прохода \t");

}

eks[i] = max;

printf("\n");

}

printf("\n");

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

printf("%d \t", eks[j]);

min = eks[1];

max = eks[1];

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

{

if (eks[i] > max)

max = eks[i];

if ((eks[i] < min) && (eks[i] != 0))

min = eks[i];

}

printf("\n\nДиаметр графа=%d \t радиус графа=%d\n\n", max, min);

printf("\n\nПериферийные вершины\n\n");

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

if (eks[i] == max)

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

printf("\n\nЦентральные вершины\n\n");

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

if (eks[i] == min)

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

printf("\n\n");

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

{

step[i] = 0;

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

{

if (G[i][j] != 0)

step[i]++;

}

switch (step[i])

{

case 0: printf("\n\n%d вершина-изолированая\n\n", i + 1);

break;

case 1:printf("\n\n%d вершина-концевая\n\n", i + 1);

break;

default:

if (step[i] == (N-1))

printf("\n\n%d вершина-доминирующая\n\n", i + 1);

}

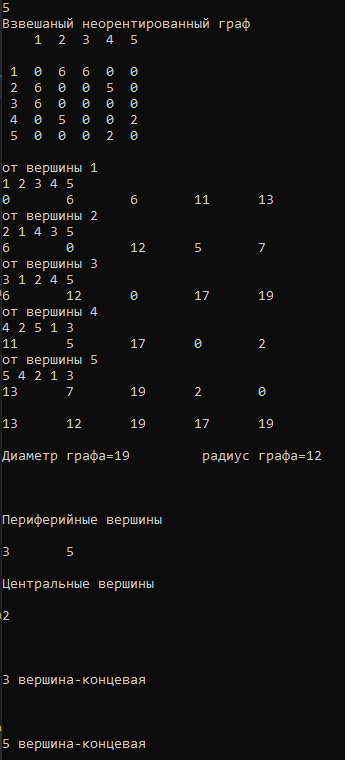
}

getchar();

getchar();

}

**Результат работы кода**



**Вывод:** научился находить эксцентриситет вершины, определять радиус и диаметр графа, находить изолированные, концевые и доминирующие вершины.