Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №8

по курсу «Л и ОА в ИЗ»

на тему «Определение характеристик графов»

**Выполнили:**

студентом группы 19ВВ3

Земляков В.Д.

**Принял:**

Митрохин М.А.

Пенза 2020

**Цель работы:**

Изучить способы определения характеристик графов.

**Ход работы:**

Эксцентриситет вершины – расстояние до наиболее удаленной вершины графа.

Максимальный эксцентриситет среди эксцентриситетов всех вершин

графа называется диаметром графа.

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.

**Листинг:**

**Header.h:**

#pragma once

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <conio.h>

#include<locale.h>

#include <String.h>

#include <time.h>

#include <stack>

#include <queue>

#include <iostream>

#include <limits.h>

**LAB7.cpp:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "Header.h"

void output(int\*\* Array, int N) {

printf("\n");

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

{

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

{

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

}

printf("\n");

}

printf("\n");

}

void GenWeighed(int\*\* Array, int N) {

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

{

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

{

int z = rand() % 100;

if (z > 80) {

Array[i][j] = rand() % N;

}

else {

Array[i][j] = 0;

}

Array[j][i] = Array[i][j];

if (i == j) {

Array[i][j] = 0;

}

}

}

output(Array, N);

}

void ClearDistanceWeighed(int\* Dist, int N) {

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

Dist[i] = 1000;

}

}

void BFSWeighed(int\*\* Array, int v, int\* DIST, int N) {

std::queue <int> Q;

Q.push(v);

DIST[v] = 0;

while (!Q.empty()) {

v = Q.front();

Q.pop();

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

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

Q.push(i);

DIST[i] = DIST[v] + Array[v][i];

}

}

}

}

void BFSIN(int\*\* Array, int\* DIST, int i, int count, int N) {

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

if (Array[i][j] != 0) {

for (int k = 0; k < N; k++) {

if ((Array[k][j] != 0) && (DIST[k] == 1000)) {

DIST[k] = DIST[i] + Array[k][j];

BFSIN(Array, DIST, i, count, N);

}

}

}

}

}

void transfer(int \*\*ArrayM1,int \*\*ArrayM2,int N,int count) {

int index;

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

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

ArrayM2[i][j] = 0;

}

}

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

for (int j = i+1; j < N; j++) {

if (ArrayM1[i][j] != 0) {

ArrayM2[i][index] = ArrayM2[j][index]=ArrayM1[i][j];

index++;

}

}

}

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

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

printf("%d\t", ArrayM2[i][j]);

}

printf("\n");

}

}

void operations(int\*\* ArrayM1, int\* eks,int N, int\* DIST, int R, int D, int count ) {

printf("Матрица смежности:\n");

GenWeighed(ArrayM1, N);

ClearDistanceWeighed(DIST, N);

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

eks[i] = -1;

}

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

BFSWeighed(ArrayM1, i, DIST, N);

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

if (eks[i] < DIST[j] && DIST[j] != 1000) {

eks[i] = DIST[j];

}

}

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

R = eks[i];

}

if (eks[i] > D) {

D = eks[i];

}

ClearDistanceWeighed(DIST, N);

}

printf("Вершины:");

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

printf("%d ", eks[i]);

}

printf("\nРадиус: %d", R);

printf("\nДиаметр: %d", D);

printf("\nЦентральные вершины:");

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

if (eks[i] == R) {

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

}

}

printf("\nПереферийные вершины:");

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

if (eks[i] == D) {

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

}

}

printf("\nИзолированные:");

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

if (eks[i] == 0) {

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

}

}

printf("\nКонцевые:");

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

count = 0;

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

if (ArrayM1[i][j] != 0) {

count++;

}

}

if (count == 1) {

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

}

}

printf("\nДоминирующие:");

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

count = 0;

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

if (ArrayM1[i][j] != 0) {

count++;

}

}

if (count == N - 1) {

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

}

}

printf("\n");

}

void ioperations(int\*\* ArrayM2, int\* eks, int N, int\* DIST, int R, int D, int count) {

ClearDistanceWeighed(DIST, N);

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

eks[i] = -1;

}

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

DIST[i] = 0;

BFSIN(ArrayM2, DIST, i, count, N);

for (int k = 0; k < N; k++) {

if ((eks[i] < DIST[k]) && (DIST[k] != 1000)) {

eks[i] = DIST[k];

}

}

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

R = eks[i];

}

if (eks[i] > D) {

D = eks[i];

}

ClearDistanceWeighed(DIST, N);

}

printf("Вершины:");

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

printf("%d ", eks[i]);

}

printf("\nРадиус: %d", R);

printf("\nДиаметр: %d", D);

printf("\nЦентральные вершины:");

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

if (eks[i] == R) {

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

}

}

printf("\nПереферийные вершины:");

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

if (eks[i] == D) {

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

}

}

printf("\nИзолированные:");

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

if (eks[i] == 0) {

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

}

}

printf("\nКонцевые:");

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

int index = 0;

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

if (ArrayM2[i][j] != 0) {

index++;

}

}

if (index == 1) {

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

}

}

printf("\nДоминирующие:");

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

int index = 0;

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

if (ArrayM2[i][j] != 0) {

index++;

}

}

if (index == count) {

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

}

}

}

int main()

{

srand(time(NULL));

setlocale(LC\_ALL, "RUS");

int count=0;

int N = 0;

int D = -1, R = 10000;

int\*\* ArrayM1 = NULL;

int\*\* ArrayM2 = NULL;

int\* DIST = NULL;

int\* eks = NULL;

scanf("%d", &N);

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

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

ArrayM1 = (int\*\*)malloc(N \* sizeof(int\*));

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

{

ArrayM1[i] = (int\*)malloc(N \* sizeof(int));

}

//Матрица смежности//

operations(ArrayM1, eks, N, DIST, R, D, count);

//Матрица инцидентности//

D = -1, R = 10000; count = 0;

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

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

if (ArrayM1[i][j] != 0) {

count++;

}

}

}

printf("Количество рёбер=%d\n",count);

ArrayM2 = (int\*\*)malloc(N \* sizeof(int\*));

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

{

ArrayM2[i] = (int\*)malloc(count \* sizeof(int));

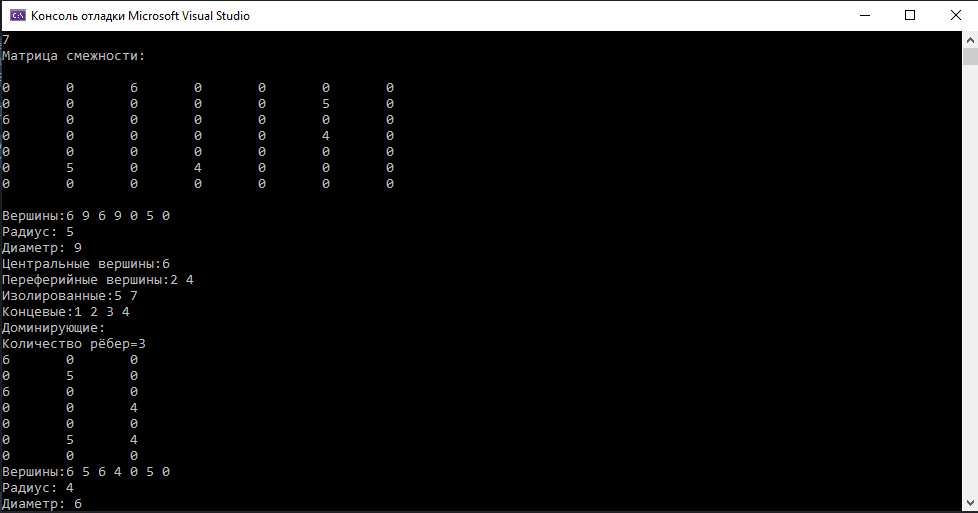
}

transfer(ArrayM1, ArrayM2, N, count);

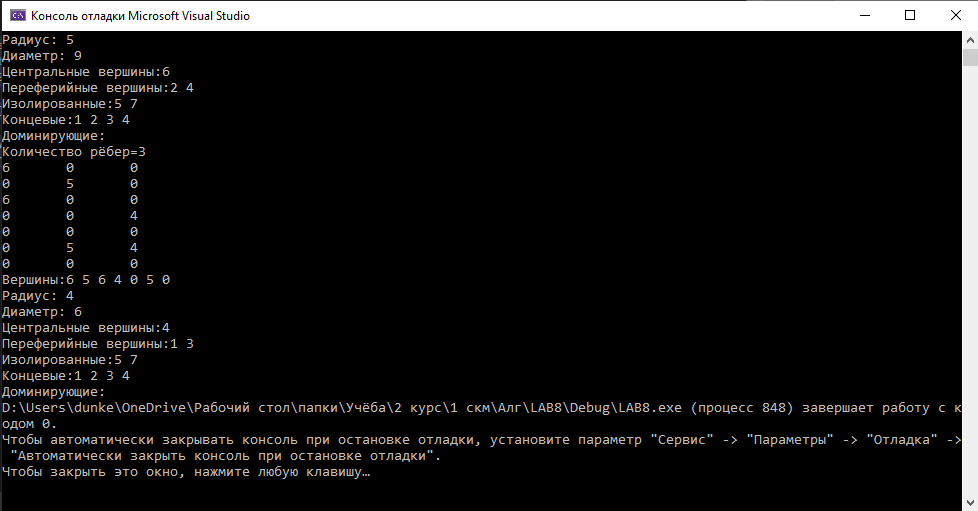
ioperations(ArrayM2, eks, N, DIST, R, D, count);

}

### Результат работы программы:



**Рисунок1. Результат работы программы**



**Рисунок 2. Результат работы программы**

### Выводы

В ходе выполнения лабораторной работы изучил способы определения характеристик графа.