Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»  
  
  
  
  
  
  
**ОТЧЁТ**

по лабораторной работе № 10

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Поиск расстояний в взвешенном графе»

Выполнил:

Студент групп 21ВВ1.3

Самохвалов Я.Д.

Приняли:

к.т.н., доцент Юрова О.А.

д.т.н., профессор, зав. каф. ВТ Митрохин М. А.

Пенза 2022

**Название:** Поиск расстояний во взвешенном графе

**Цель работы:** Выполнить поиск расстояний во взвешенном графе и найти диаметр и радиус графа и периферийные и центральные вершины соответственно.

**Лабораторные задания:**

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного взвешенного графа *G*. Выведите матрицу на экран.
2. Для сгенерированного графа осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием. При реализации алгоритма в качестве очереди используйте класс **queue** из стандартной библиотеки С++.
3. \* Сгенерируйте (используя генератор случайных чисел) матрицу смежности для ориентированного взвешенного графа *G*. Выведите матрицу на экран и осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием.

**Задание 2**

1. Для каждого из вариантов сгенерированных графов (ориентированного и не ориентированного) определите радиус и диаметр.
2. Определите подмножества периферийных и центральных вершин.

**Листинг:**

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <conio.h>

#include <malloc.h>

#include <locale.h>

#include <queue>

using namespace std;

#pragma warning (disable:4996)

int \_stateMenu;

void Menu()

{

printf("Выберите действие:\n1 - Ввод неориентированного графа\n2 - Ввод ориентированного графа\n0 - Выйти\n");

\_stateMenu = \_getch() - 48;

}

void BFSD(int\* VIS, int\*\* M, int n, queue <int> q, int v)

{

VIS[v] = 0;

q.push(v);

while (!q.empty())

{

v = q.front();

q.pop();

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

{

if (M[v][i] > 0 && VIS[i] == -1)

{

q.push(i);

VIS[i] = VIS[v] + M[v][i];

}

}

}

}

void BFSDK(int\* VIS, int\*\* M, int n, queue <int> q, int v, int\* DIS)

{

int k = v;

VIS[v] = 0;

q.push(v);

while (!q.empty())

{

v = q.front();

q.pop();

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

{

if (M[v][i] > 0 && VIS[i] == -1)

{

q.push(i);

VIS[i] = VIS[v] + M[v][i];

}

}

}

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

{

if (VIS[i] > DIS[k]) DIS[k] = VIS[i];

}

}

int main()

{

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

int n, V = 0;

int maxv = -1, minv = 2147483647;

int d = -1, r = 2147483647;

int\*\* M, \* VIS, \* DIS;

printf("Введите размер графа: ");

scanf("%d", &n);

printf("\n");

queue <int> q;

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

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

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

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

{

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

DIS[i] = 0;

}

Menu();

while (\_stateMenu != 0)

{

switch (\_stateMenu)

{

case 1:

printf("\n");

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

{

for (int j = 0; j < i + 1; j++)

{

M[i][j] = rand() % 2;

if (M[i][j] == 1)

{

M[i][j] = rand() % 101 + 1;

}

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

if (i == j)

{

M[i][j] = 0;

}

}

}

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

{

VIS[i] = -1;

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

{

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

}

printf("\n");

}

printf("Нахожднние диаметра и радиуса графа, множества центральных и периферийных вершин.\n");

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

{

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

{

VIS[j] = -1;

}

BFSDK(VIS, M, n, q, i, DIS);

printf("Вершина №%d\n", i + 1);

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

{

if (VIS[i] == -1) printf("%d - -1\n", i + 1);

else printf("%d - %d\n", i + 1, VIS[i]);

}

if (DIS[i] < r && DIS[i] > 0) r = DIS[i];

if (DIS[i] > d) d = DIS[i];

}

printf("Эксцентрисеты вершин: ");

for (int i = 0; i < n; i++) printf("%d - %d; ", i + 1, DIS[i]);

printf("\n");

printf("D(G) = %d, r(G) = %d\n", d, r);

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

{

if (DIS[i] == r) printf("%d - центральная вершина\n", i + 1);

if (DIS[i] == d) printf("%d - периферийная вершина\n", i + 1);

}

getch();

exit(EXIT\_SUCCESS);

case 2:

printf("\n");

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

{

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

{

M[i][j] = rand() % 2;

if (M[i][j] == 1)

{

M[i][j] = rand() % 99 + 1;

}

if (i == j)

{

M[i][j] = 0;

}

}

}

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

{

VIS[i] = -1;

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

{

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

}

printf("\n");

}

printf("Нахожднние диаметра и радиуса графа, множества центральных и периферийных вершин.\n");

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

{

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

{

VIS[j] = -1;

}

BFSDK(VIS, M, n, q, i, DIS);

printf("Вершина №%d\n", i + 1);

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

{

if (VIS[i] == -1) printf("%d - -1\n", i + 1);

else printf("%d - %d\n", i + 1, VIS[i]);

}

if (DIS[i] < r && DIS[i] > 0) r = DIS[i];

if (DIS[i] > d) d = DIS[i];

}

printf("Эксцентрисеты вершин: ");

for (int i = 0; i < n; i++) printf("%d - %d; ", i + 1, DIS[i]);

printf("\n");

printf("D(G) = %d, r(G) = %d\n", d, r);

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

{

if (DIS[i] == r) printf("%d - центральная вершина\n", i + 1);

if (DIS[i] == d) printf("%d - периферийная вершина\n", i + 1);

}

getch();

exit(EXIT\_SUCCESS);

case 0:

exit(EXIT\_SUCCESS);

default:

system("cls");

printf("Неправильная цифра\n");

system("pause");

system("cls");

Menu();

break;

}

}

return 0;

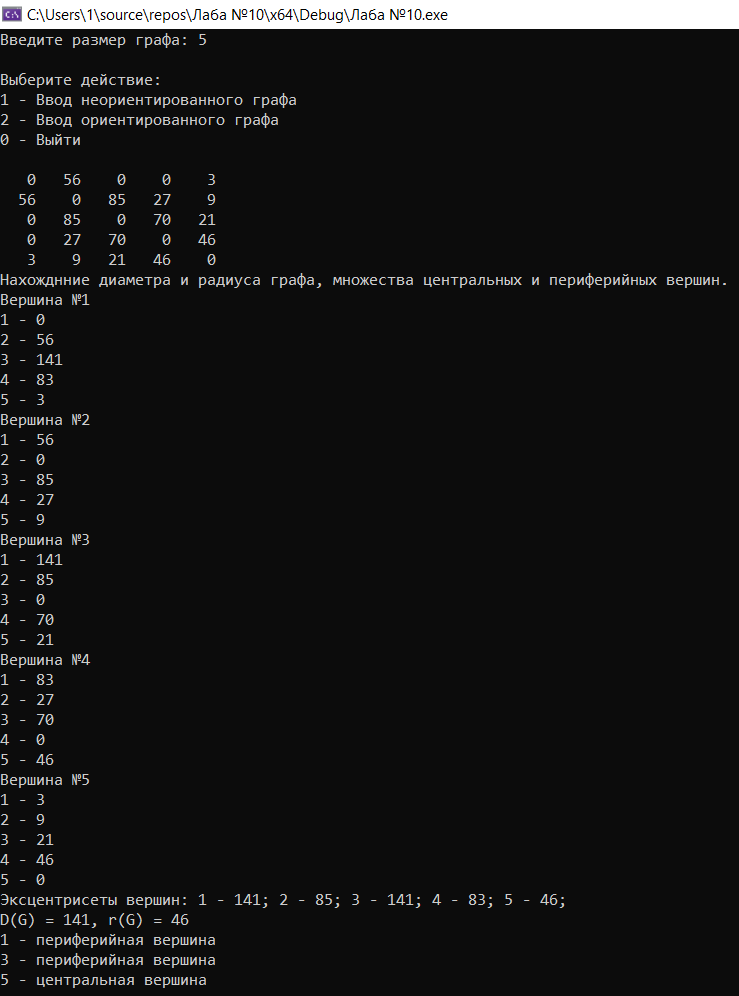
}

**Вывод результатов:**

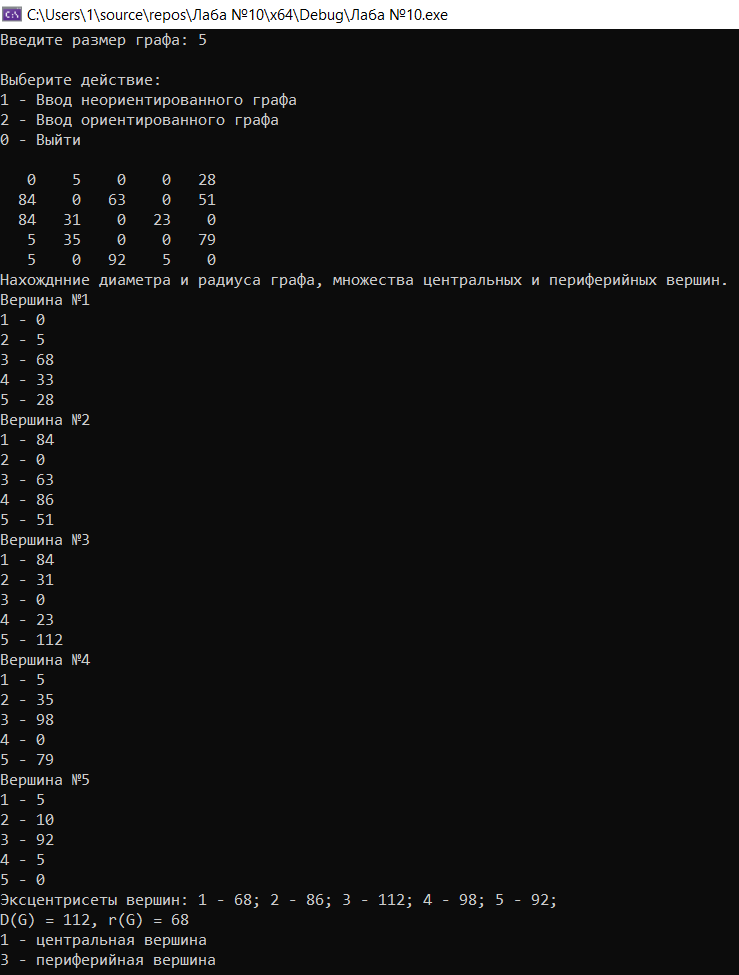
Первым делом мы записываем количество вершин в нашем графе и выбираем сам граф: ориентированный или неориентированный.

Граф выводится, а за ней нахождение среди всех расстояний диаметр и радиус графа и периферийные и центральные вершины соответственно.

Для неориентированного графа:

****

Для ориентированного графа:

****

**Вывод:** мы создали программу, которая выполняет поиск расстояний во взвешенном графе и находит диаметр и радиус графа и периферийные и центральные вершины соответственно, и она корректно работает.