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»

**Выполнил:**

студент группы 20ВВ3

Духнов Олег

**Приняли:**

Юрова О.В.

Митрохин М.А.

Пенза 2021

# Название

# определение характеристик графов.

# Лабораторное задание

**Задание 1**

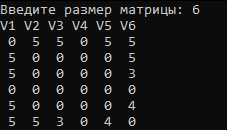
1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного взвешенного графа *G*. Выведите матрицу на экран.
2. Определите радиус и диаметр графа *G*, используя матрицу смежности графа.
3. Определите подмножества периферийных и центральных вершин графа *G*, используя 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.

# Результат работы программы

**Задание 1.1.** Результаты работы программы показаны на рисунке 1.



**Рисунок 1 – Результат работы программы**

**Задание 1.2.** Радиус и диаметр графа.



**Рисунок 2 – Результат работы программы**

**Задание 1.3.** Подмножества периферийных и центральных вершин.

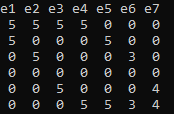


**Рисунок 3 – Результат работы программы**

**Задание 1.4.** Изолированные, концевые и доминирующие вершины.



**Задание 2.1.** Матрица инцидентности для неориентированного взвешенного графа.



**Рисунок 4 – Результат работы программы**

**Задание 2.2.** Радиус и диаметр графа матрицы инцидентности.



**Задание 2.3.** Подмножества периферийных и центральных вершин.



**Рисунок 3 – Результат работы программы**

**Вывод:** яизучил алгоритмы для определения характеристик графа.

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <stdbool.h>

#include <conio.h>

#include <time.h>

#include <stack>

#include <queue>

using namespace std;

int\*\* neor\_vz(int n)

{

int s = 1;

int\*\* Arr;

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

srand(time(NULL));

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

{

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

Arr[i][i] = 0;

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

{

if (rand() % 100 > 72)

Arr[i][j] = 0;

else

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

}

s++;

}

s = 1;

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

{

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

Arr[j][i] = Arr[i][j];

s++;

}

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

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

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

{

printf("\n");

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

{

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

}

}

return Arr;

}

void BFSD(int x, int\* dist, int\*\* Arr, int n)

{

queue<int> que;

que.push(x);

dist[x] = 0;

while (!que.empty())

{

x = que.front();

que.pop();

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

{

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

{

que.push(i);

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

}

}

}

}

void BFSDMIX(int x, int\* dist, int\*\* Arr, int n, int p)

{

queue<int> que;

que.push(x);

dist[x] = 0;

while (!que.empty())

{

x = que.front();

que.pop();

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

{

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

{

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

{

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

{

que.push(j);

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

}

}

}

}

}

}

void paragrah\_1(int\*\* t, int n)

{

int d = -1, r = 10000, s = 0;

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

int\* step = (int\*)malloc(n \* sizeof(int)), \* isol = (int\*)malloc(n \* sizeof(int)), \* konz = (int\*)malloc(n \* sizeof(int)), \* domin = (int\*)malloc(n \* sizeof(int)), \* extr = (int\*)malloc(n \* sizeof(int));

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

{

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

dist[k] = 9999;

BFSD(i, dist, t, n);

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

{

if (extr[i] < dist[j] && dist[j] != 9999)

extr[i] = dist[j];

}

}

printf("\nЭксцентриситет: ");

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

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

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

{

if (d < extr[i])

d = extr[i];

if (r > extr[i] && extr[i] != 0)

r = extr[i];

}

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

step[i] = 0;

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

{

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

{

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

step[i]++;

}

}

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

isol[i] = 0;

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

if (step[i] == 0)

isol[i] = 1;

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

konz[i] = 0;

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

if (step[i] == 1)

konz[i] = 1;

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

domin[i] = 0;

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

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

domin[i] = 1;

printf("\nРадиус: %d", r); printf("\nДиаметр: %d", d);

printf("\n\nПереферийные вершины: ");

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

if (extr[i] == d) printf("%d ", i + 1);

printf("\nЦентральные вершины: ");

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

if (extr[i] == r) printf("%d ", i + 1);

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

if (isol[i] == 1)

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

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

if (konz[i] == 1)

printf("\nКонцевая %d - я вершина", i + 1);

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

if (domin[i] == 1)

printf("\nДоминантная %d - я вершина", i + 1);

}

void paragrah\_2(int\*\* A, int n)

{

int s = 1, x = 0, \*\* A1 = (int\*\*)malloc(n \* sizeof(int\*)), num = 0, \* extr = (int\*)malloc(n \* sizeof(int)), d = -1, r = 1000;

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

{

A1[i] = (int\*)malloc(n \* sizeof(int));

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

A1[i][j] = -1;

}

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

{

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

{

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

{

A1[i][j] = num;

x++;

num++;

}

}

s++;

}

s = 1;

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

{

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

{

A1[j][i] = A1[i][j];

}

s++;

}

int\*\* M1 = (int\*\*)malloc(n \* sizeof(int\*));

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

{

M1[i] = (int\*)malloc(x \* sizeof(int));

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

M1[i][j] = 0;

}

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

{

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

{

if (A1[i][j] > -1)

{

M1[i][A1[i][j]] = A[i][j];

}

}

}

printf("\n\n");

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

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

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

{

printf("\n");

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

printf("%2d ", M1[i][j]);

}

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

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

extr[i] = -1;

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

{

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

dist[k] = 9999;

BFSDMIX(i, dist, M1, n, x);

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

{

if (extr[i] < dist[j] && dist[j] != 9999)

extr[i] = dist[j];

}

}

printf("\nЭксцентриситет: ");

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

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

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

{

if (d < extr[i])

d = extr[i];

if (r > extr[i] && extr[i] != 0)

r = extr[i];

}

printf("\nРадиус: %d", r); printf("\nДиаметр: %d", d);

printf("\n\nПереферийные вершины: ");

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

if (extr[i] == d) printf("%d ", i + 1);

printf("\nЦентральные вершины: ");

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

if (extr[i] == r) printf("%d ", i + 1);

}

int main()

{

setlocale(LC\_ALL, "Ru");

int n, \*\* t;

printf("Введите размер матрицы: ");

scanf("%d", &n);

t = neor\_vz(n);

paragrah\_1(t, n);

paragrah\_2(t, n);

\_getch();

}