Министерство образования Российской Федерации

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»

Выполнили студенты группы 20ВВ3:

Мальков И.А.

Педай Н.Д.

Приняли:

Митрохин М.А.

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.

**Ход работы:**

1. Код программы

#include <stdio.h>

#include <iostream>

#include <locale.h>

#include <stdlib.h>

#include <queue>

using namespace std;

int e = 0;

void BFSD(int\*\* G, int n, int\* dist, int s) {

queue<int> q;

q.push(s);

dist[s] = 0;

while (!q.empty()) {

s = q.front();

q.pop();

//cout << s << " ";

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

if (G[s][j] > 0 && dist[j] > dist[s] + G[s][j]) {

dist[j] = dist[s] + G[s][j];

q.push(j);

}

}

}

}

void BFSD\_IND(int\*\* M, int\* dist, int s, int n) {

queue<int> q;

q.push(s);

dist[s] = 0;

while (!q.empty()) {

s = q.front();

q.pop();

//cout << s << " ";

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

if (M[s][i] > 0) {

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

if (M[j][i] > 0 && dist[j] > dist[s] + M[j][i]) {

dist[j] = dist[s] + M[j][i];

q.push(j);

}

}

}

}

}

}

int\*\* create(int n) {

int\*\* G, k = 1;

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

srand(time(NULL));

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

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

G[i][i] = 0;

for (int j = k; j < n; j++) {

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

}

k++;

}

k = 1;

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

for (int j = k; j < n; j++) {

G[j][i] = G[i][j];

}

k++;

}

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

cout << "V" << i << " ";

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

cout << endl;

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

cout << G[i][j] << " ";

}

}

return G;

}

void vert(int\*\* G, int n) {

cout << endl;

int x = 0;

cout << "isolated: ";

for (int p = 0; p < n; p++) {

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

if (G[p][k] == 0) {

x++;

}

}

if (x == n) {

cout << "V" << p << " ";

}

x = 0;

}

cout << endl;

cout << "end: ";

for (int p = 0; p < n; p++) {

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

if (G[p][k] != 0) {

x++;

}

}

if (x == 1) {

cout << "V" << p << " ";

}

x = 0;

}

cout << endl;

cout << "dominate: ";

for (int p = 0; p < n; p++) {

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

if (G[p][k] != 0) {

x++;

}

}

if (x == n - 1) {

cout << "V" << p << " ";

}

x = 0;

}

cout << endl;

}

int\*\* ind(int\*\* G, int n) {

int v, num = 0;

//int e = 0;

int k = 1;

int\*\* M;

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

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

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

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

N[i][j] = -1;

}

}

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

for (int j = k; j < n; j++) {

if (G[i][j] > 0) {

N[i][j] = num;

num++;

}

}

k++;

}

k = 1;

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

for (int j = k; j < n; j++) {

N[j][i] = N[i][j];

}

k++;

}

cout << endl;

/\*for (int i = 0; i < n; i++)

cout << "A" << i << " ";

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

cout << endl;

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

cout << N[i][j] << " ";

}

}\*/

cout << endl;

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

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

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

e++;

}

}

}

v = n;

e = e / 2;

//cout << endl << v << " " << e << " " << endl;

k = 1;

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

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

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

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

M[i][j] = 0;

}

k++;

}//////////////////////////

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

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

if (G[i][j] > -1) {

M[i][N[i][j]] = G[i][j];

}

}

}//////////////////////////

cout << endl;

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

cout << "E" << i << " ";

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

cout << endl;

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

cout << M[i][j] << " ";

}

}

cout << endl;

return M;

}

int main() {

int n, s, \*\* t, \*\* t1, \*\* t2, \*\* t3;

int rad = INT16\_MAX;

int diam = 0;

cout << "Enter array size: ";

cin >> n;

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

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

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

dist[i] = INT16\_MAX;

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

extr[i] = -1;

cout << endl;

t = create(n);

/////////////////////////////////////////////////////

cout << endl;

cout << endl;

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

for (int m = 0; m < n; m++) {

dist[m] = INT16\_MAX;

}

BFSD(t, n, dist, i);

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

if (extr[i] < dist[j] && dist[j] != 0)

extr[i] = dist[j];

}

}

cout << "excentricities: " << endl;

for (int p = 0; p < n; p++) {

cout << extr[p] << " ";

}

cout << endl;

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

if (extr[i] < rad && dist[i] != 0)

rad = extr[i];

}

cout << endl << "rad = " << rad;

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

if (extr[i] > diam)

diam = extr[i];

}

cout << endl << "diam = " << diam << endl;

/////////////////////////////////////////////////

cout << endl;

cout << "center: ";

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

if (extr[i] == rad)

cout << "V" << i << " ";

}

cout << endl;

cout << "peripheral: ";

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

if (extr[i] == diam)

cout << "V" << i << " ";

}

cout << endl;

////////////////////////////////////////////////

vert(t, n);

////////////////////////////////////////////////

t1 = ind(t, n);

cout << endl;

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

dist[i] = INT16\_MAX;

//BFSD\_IND(t1, dist, s, n);

cout << endl;

for (int p = 0; p < n; p++) {

extr[p] = -1;

}

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

for (int m = 0; m < n; m++) {

dist[m] = INT16\_MAX;

}

BFSD\_IND(t1, dist, i, n);

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

if (extr[i] < dist[j] && dist[j] != 0)

extr[i] = dist[j];

}

}

cout << "excentricities: " << endl;

for (int p = 0; p < n; p++) {

cout << extr[p] << " ";

}

cout << endl;

//////////////////////////////////////////

cout << endl;

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

if (extr[i] < rad && dist[i] != 0)

rad = extr[i];

}

cout << endl << "rad = " << rad;

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

if (extr[i] > diam)

diam = extr[i];

}

cout << endl << "diam = " << diam << endl;

//////////////////////////////////////////

cout << endl;

cout << "center: ";

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

if (extr[i] == rad)

cout << "V" << i << " ";

}

cout << endl;

cout << "peripheral: ";

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

if (extr[i] == diam)

cout << "V" << i << " ";

}

cout << endl;

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

free(t[i]);

free(t);

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

free(t1[i]);

free(t1);

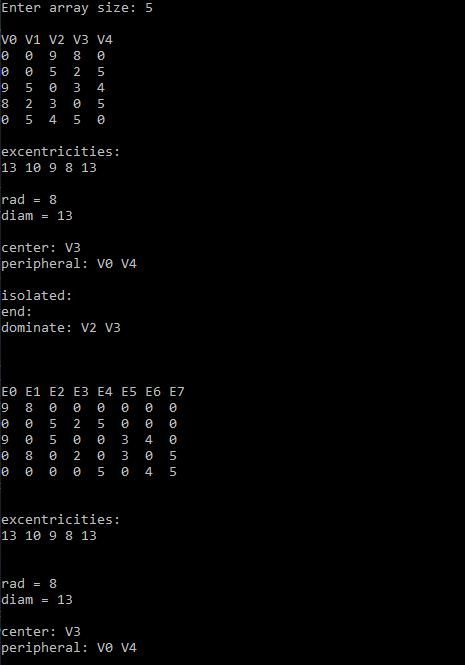
free(dist);

//system("pause");

return 0;

}

1. Результат работы программы.



Вывод: в ходе выполнения данной лабораторной работы разработали программу, согласно заданию.