|  |
| --- |
| 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» |
|  |
|  |
| Выполнила студентка группы 19ВВ3:  Ханбекова Е. В.  Принял:  Митрохин М. А. |
| Пенза  2020 |

**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.

**Листинг**

#include <stdio.h>

#include <conio.h>

#include <Windows.h>

#include <stdlib.h>

#include <queue>

#include <iostream>

using namespace std;

typedef int tData;

typedef struct sNode {

tData value;

tData nom;

struct sNode \*next;

} tNode;

tNode\* create\_list(int N, int k, int \*\*A, int i)

{

tNode \*p\_begin = NULL;

tNode \*p = NULL;

//заполним односвязный список

p\_begin = (tNode \*)malloc(sizeof(tNode));

p = p\_begin;

p->next = NULL;

p->value = i + 1;

p->nom = i + 1;

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

if (A[i][x] != 0) {

p->next = (tNode \*)malloc(sizeof(tNode));

p = p->next;

//заполнить \_новую\_ структуру данных

p->next = NULL;

p->nom = x + 1;

p->value = A[i][x];

}

}

return p\_begin;

}

void print\_list(tNode \*p\_begin)

{

tNode \*p = p\_begin;

while (p != NULL) {

printf("-> %d ", p->nom);

p = p->next;

}

}

tNode \*\*p\_begin;

int BFS(int\*\*A, int\*M, int i, int N) {

queue <int> Q;

int j;

Q.push(i);

M[i] = 0;

while (Q.empty() != true) {

i = Q.front();

Q.pop();

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

if ((A[i][j] != 0) && (M[j] > M[i] + A[i][j])) {

M[j] = 0;

Q.push(j);

M[j] = M[i] + A[i][j];

}

}

}

return 0;

}

int main(void)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int N, i, j, C=0;

printf("Ведите размер матрицы \n");

scanf\_s("%d", &N);

srand(13);

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

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

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

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

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

}

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

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

A[i][j] = rand() % 5;

if (A[i][j] > 2) {

A[i][j] = 0;

}

A[j][i] = A[i][j];

if (i == j) {

A[i][j] = 0;

}

}

}

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

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

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

}

printf("\n");

}

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

M[i] = 1000;

}

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

EX[i] = -1;

}

printf("\n\nПоиск эксцентриситетов :\n");

int D = -1, r = 10000;

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

BFS(A, M, i, N);

printf("\n");

/\*for (int l = 0; l < N; l++) {

printf("%d ", M[l]);

}\*/

int k;

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

if ((EX[i] < M[k]) && (M[k]!=1000)) {

EX[i] = M[k];

}

}

if ((EX[i] < r)&&(EX[i] != 0 )) {

r = EX[i];

}

if (EX[i] > D) {

D = EX[i];

}

printf("эксцентириситет вершины %d : %d\n", i + 1, EX[i]);

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

M[l] = 1000;

}

}

printf("радиус графа : %d\n", r);

printf("диаметр графа : %d\n", D);

printf("центральные вершины: ");

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

if (EX[n] == r) {

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

}

}

printf("\nпериферийные вершины: ");

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

if (EX[n] == D) {

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

}

}

printf("\nизолированные вершины: ");

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

if (EX[n] == 0) {

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

}

}

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

int step = 0;

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

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

step++;

}

}

if (step == 1) {

printf("\nВершина %d концевая", i);

}

if (step == N-1) {

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

}

}

free(A);

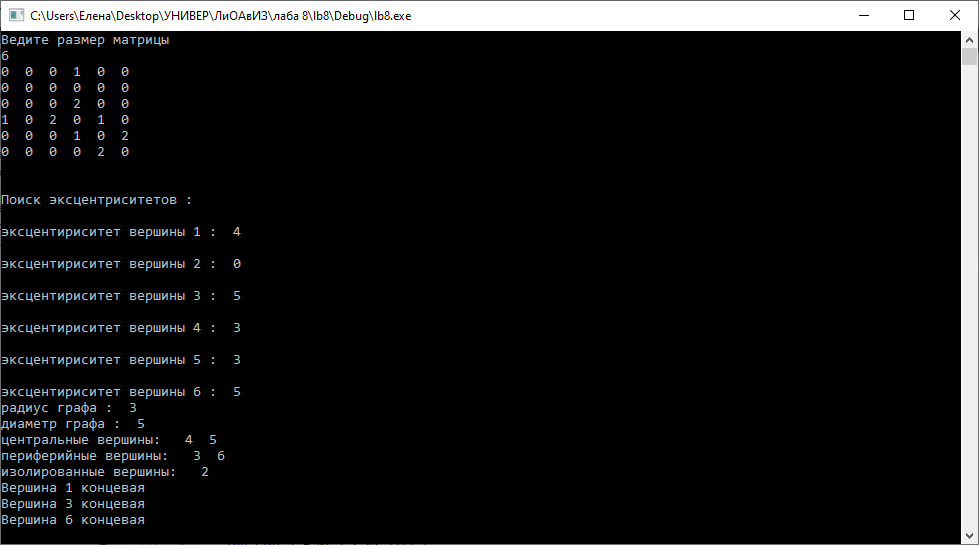
free(M);

free(EX);

\_getch();

}

**Результат работы программы:**



**Вывод:**

В результате работы программы на матрице смежности графа были найдены эксцентриситеты вершин, радиус и диаметр графа, множества центральных, периферийных, изолированных, концевых и доминирующих вершин. Все это является характеристиками графа.