Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №8

# по дисциплине: «Логика и основы алгоритмизации в инженерных задачах»

на тему: «Определение характеристик графов»

Выполнили:

студенты группы 20ВВ3

Горожанин Я. А.

Тихонов А. А.

Скирдова В. М.

Приняли:

Юрова О. В.

Митрохин М. А.

Пенза 2021

**Название**

Определение характеристик графов

**Цель работы**

Научиться определять характеристики графов, среди которых эксцентриситет вершины; радиус, диаметр графа, а также изолированные, концевые и доминирующие вершины, подмножество периферийных и центральных вершин графа.

**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.

**Листинг**

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <locale.h>

#include <iostream>

#include <stack>

#include <queue>

#include <iomanip>

using namespace std;

int rad = 1000;

int dum = 0;

void Google\_search(int\*\* a, int\*\* dist, int size) {

queue <int> q;

int num;

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

int j = i;

dist[i][j] = 0;

q.push(i);

while (!q.empty()) {

num = q.front();

q.pop();

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

if (a[num][j] > 0 && dist[i][j] > dist[i][num] + a[num][j]) {

q.push(j);

dist[i][j] = dist[i][num] + a[num][j];

}

}

}

}

}

void rad\_dum\_wasteland( int\* ext, int size, int\*\* distG ) {

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

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

if (distG[i][j] > ext[i])

ext[i] = distG[i][j];

}

cout << endl << " Эксцентриситеты всех вершин: " << endl;

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

if (ext[i] == 1000)

cout << " Вершина изолирована ";

else

cout << " " << ext[i];

}

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

if (ext[i] > dum)

dum = ext[i];

if (ext[i] < rad)

rad = ext[i];

}

cout << endl << endl << " Радиус графа равен: " << rad;

cout << endl << " Диаметр графа равен: " << dum;

}

void rad\_dum\_wasteland\_v2(int\* ext, int size) {

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

if (ext[i] > dum)

dum = ext[i];

if (ext[i] < rad)

rad = ext[i];

}

cout << endl << " Радиус графа равен: " << rad;

cout << endl << " Диаметр графа равен: " << dum;

}

void best\_wateland\_survivalists( int\* ext, int size) {

cout << endl << " Подмножество центральных вершин: ";

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

if (ext[i] == rad)

cout << setw(2) << i;

cout << endl << " Подмножество переферийных вершин: ";

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

if (ext[i] == dum)

cout << setw(2) << i;

cout << endl;

}

void step\_vrshi\_search\_v1(int size, int\*\* arr) {

auto step = new int[size];

int iznum = 0;

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

step[i] = 0;

for (int row = 0; row < size; row++)

for (int col = 0; col < size; col++) {

if (arr[row][col] > 0)

step[row]++

;

}

cout << endl << " Изолированные вершины: ";

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

if (step[i] == 0) {

cout << setw(2) << i;

iznum++;

}

cout << endl << " Концевые вершины: ";

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

if (step[i] == 1)

cout << setw(2) << i;

cout << endl << " Доминирующие вершины: ";

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

if (step[i] == size - 1 - iznum)

cout << " " << i;

}

void bfs(int\*\* arr\_inc, int num, int\* dist, int size, int edge\_count) {

queue <int> que;

dist[num] = 0;

que.push(num);

while (!que.empty()) {

num = que.front();

que.pop();

for (int i = 0; i < edge\_count; i++) {

if (arr\_inc[num][i] > 0) {

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

if (arr\_inc[count][i] > 0 && dist[count] > dist[num] + arr\_inc[count][i]) {

que.push(count);

dist[count] = dist[num] + arr\_inc[count][i];

}

}

}

}

}

}

void main() {

srand(time(0));

setlocale(LC\_ALL, "Russian");

int size;

cout << " Введите размерность матрицы: ";

cin >> size;

cout << endl;

int\*\* arr = new int\* [size];

for (int count = 0; count < size; ++count)

arr[count] = new int[size];

for (int row = 0; row < size; row++) {

arr[row][row] = 0;

for (int col = row + 1; col < size; col++) {

arr[row][col] = rand() % 100;

if (arr[row][col] < 40) {

arr[row][col] = 0;

arr[col][row] = arr[row][col];

}

else {

arr[row][col] = rand() % 15;

arr[col][row] = arr[row][col];

}

}

}

cout << " ";

for (int row = 0; row < size; row++)

{

for (int col = 0; col < size; col++) {

cout << arr[row][col] << setw(3);

}

cout << endl << " ";

}

cout << endl;

cout << endl;

///////////////////////////////////////////////////// num 1.1

auto distG = new int\* [size];

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

distG[i] = new int[size];

for (int row = 0; row < size; row++)

{

for (int col = 0; col < size; col++) {

distG[row][col] = 1000;

}

}

Google\_search(arr, distG, size);

cout << " ";

for (int row = 0; row < size; row++)

{

for (int col = 0; col < size; col++) {

if(distG[row][col] == 1000)

cout << 0 << setw(3);

else

cout << distG[row][col] << setw(3);

}

cout << endl << " ";

}

cout << endl << endl;

auto ext = new int[size];

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

ext[i] = 0;

rad\_dum\_wasteland( ext, size, distG);

//////////////////////////////////////////////////////////////// num 1.2

best\_wateland\_survivalists( ext, size);

//////////////////////////////////////////////////////////////// num 1.3

step\_vrshi\_search\_v1(size, arr);

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

int edge\_count = 0;

for (int row = 0; row < size; row++) {

for (int col = row + 1; col < size; col++) {

if (arr[row][col])

edge\_count++;

}

}

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

int\*\* arr\_inc = new int\* [size];

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

arr\_inc[count] = new int[edge\_count];

for (int col = 0; col < edge\_count; col++)

arr\_inc[count][col] = 0;

}

edge\_count = 0;

for (int row = 0; row < size; row++) {

for (int col = row + 1; col < size; col++) {

if (arr[row][col]) {

arr\_inc[row][edge\_count] = arr[row][col];

arr\_inc[col][edge\_count] = arr[row][col];

edge\_count++;

}

}

}

for (int row = 0; row < size; row++)

{

cout << endl << " ";

for (int col = 0; col < edge\_count; col++) {

cout << arr\_inc[row][col] << " ";

}

}

int\* dist = new int[size];

int\* ext\_s = new int[size];

int max;

cout << endl << endl << " Эксцентриситеты вершин: " << endl << " ";

for (int cunt = 0; cunt < size; ++cunt) {

for (int count = 0; count < size; ++count)

dist[count] = INT\_MAX;

bfs(arr\_inc, cunt, dist, size, edge\_count);

if (dist[0] == INT\_MAX)

dist[0] = 0;

max = dist[0];

for (int fuck = 0; fuck < size; ++fuck) {

if (dist[fuck] == INT\_MAX)

dist[fuck] = 0;

if (dist[fuck] > max)

max = dist[fuck];

}

ext\_s[cunt] = max;

cout << max << " ";

}

cout << endl;

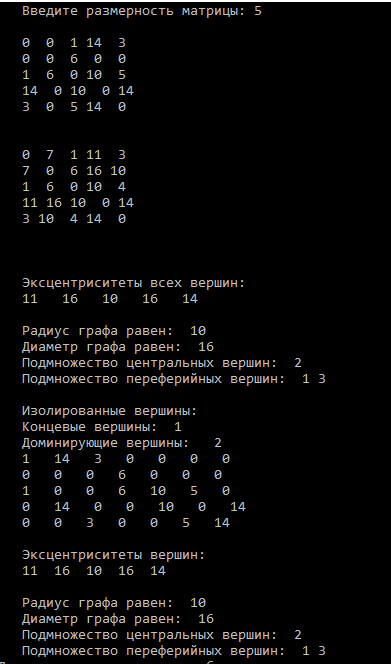
rad\_dum\_wasteland\_v2(ext\_s, size);

best\_wateland\_survivalists(ext\_s, size);

system("pause");

}

**Результат работы программы**



**Вывод**

В ходе выполнения данной лабораторной работы были получены навыки определения характеристик графов, среди которых эксцентриситет вершины; радиус, диаметр графа, а также изолированные, концевые и доминирующие вершины, подмножество периферийных и центральных вершин графа.