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»

Выполнил:

студент группы 20ВВ2

Кочергин В.П.

Принял:

к.т.н. доцент Юрова О.В.

д.т.н. профессор Митрохин М. А.

Пенза 2021

**Цель работы:** Написать алгоритм для определения характеристик графов.

**Задание 1** Сгенерировать матрицу смежности и матрицу инцидентности. Найти их характеристики.

Листинг:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <windows.h>

#include <limits.h>

#include <queue>

using namespace std;

void BFS(int\* dist, int\*\* a, int size, int num)

{

queue <int> q;

int i;

q.push(num);

dist[num] = 0;

while (!q.empty())

{

num = q.front();

q.pop();

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

{

if (dist[i] > dist[num] + a[num][i] && a[num][i] != 0)

{

q.push(i);

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

}

}

}

}

int main()

{

int i, j, size, \*\* a, \*\*g, \* dist, \*deg, ran, num = 0, rad = INT\_MAX, D = 0, \*ex, n;

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setvbuf(stdin, NULL, \_IONBF, 0);

setvbuf(stdout, NULL, \_IONBF, 0);

printf("Введите размеры массива ");

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

printf("Матрица смежности:\n");

a = (int\*\*)malloc(size \* sizeof(int));

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

deg = (int\*)malloc(size \* sizeof(int));

ex = (int\*)malloc(size \* sizeof(int));

srand(time(NULL));

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

{

a[i] = (int\*)malloc(size \* sizeof(int));

}

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

{

deg[i] = 0;

}

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

{

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

{

a[i][j] = 0;

}

}

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

{

for (j = 1 + i; j < size; j++)

{

ran = rand() % 101;

if (ran > 40) a[i][j] = 1;

else a[i][j] = 0;

}

}

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

{

for (j = 1 + i; j < size; j++)

{

if (a[i][j] == 1) a[i][j] = rand() % 8;

}

}

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

{

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

{

a[j][i] = a[i][j];

}

}

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

{

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

{

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

}

printf("\n");

}

printf("\nРасстояния до вершин:\n");

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

{

for (n = 0; n < size; n++) dist[n] = INT\_MAX;

BFS(dist, a, size, i);

for (n = 0; n < size; n++) printf("%d ", dist[n]);

printf("\n");

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

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

ex[i] = dist[j];

}

printf("\nЭксцентриситет каждой вершины равен: ");

for (i = 0; i < size; i++) printf("%d ", ex[i]);

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

{

if (ex[i] < rad) rad = ex[i];

if (ex[i] > D) D = ex[i];

}

printf("\nРадиус равен: %d", rad);

printf("\nДиаметр равен: %d\n", D);

printf("\nПерифирийные вершины: ");

for (n = 0; n < size; n++)

if (ex[n] == D) printf("%d ", n);

printf("\nЦентральные вершины: ");

for (n = 0; n < size; n++)

if (ex[n] == rad) printf("%d ", n);

printf("\nСтепени вершин: ");

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

{

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

if (a[i][j] > 0) deg[i] += 1;

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

}

printf("\n\nИзолированные вершины: ");

for (n = 0; n < size; n++)

if (deg[n] == 0) printf("%d ", n);

printf("\nКонцевые вершины: ");

for (n = 0; n < size; n++)

if (deg[n] == 1) printf("%d ", n);

printf("\nДоминирующие вершины: ");

for (n = 0; n < size; n++)

if (deg[n] == size - 1) printf("%d ", n);

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

{

ex[i] = 0;

}

int sum = 0, \*distx;

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

{

sum += deg[i];

}

sum /= 2;

g = (int\*\*)malloc(sum \* sizeof(int));

distx = (int\*)malloc(size \* sizeof(int));

for (i = 0; i < sum; i++)

{

g[i] = (int\*)malloc(size \* sizeof(int));

}

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

{

for (j = 0; j < sum; j++)

{

g[i][j] = 0;

}

}

n = 0;

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

{

for (j = 1 + i; j < size; j++)

{

if (a[i][j] != 0)

{

g[i][n] = a[i][j];

g[j][n] = a[j][i];

n += 1;

}

}

}

printf("\nМатрица инцидентности:\n");

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

{

for (j = 0; j < sum; j++)

{

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

}

printf("\n");

}

printf("\nРасстояния до вершин:\n");

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

{

for (n = 0; n < sum; n++) distx[n] = INT\_MAX;

BFS(distx, a, size, i);

for (n = 0; n < size; n++) printf("%d ", distx[n]);

printf("\n");

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

if (ex[i] < distx[j] && distx[j] != 0)

{

ex[i] = distx[j];

}

}

printf("\nЭксцентриситет каждой вершины равен: ");

for (i = 0; i < size; i++) printf("%d ", ex[i]);

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

{

if (ex[i] < rad) rad = ex[i];

if (ex[i] > D) D = ex[i];

}

printf("\nРадиус равен: %d", rad);

printf("\nДиаметр равен: %d\n", D);

printf("\nПерифирийные вершины: ");

for (n = 0; n < size; n++)

if (ex[n] == D) printf("%d ", n);

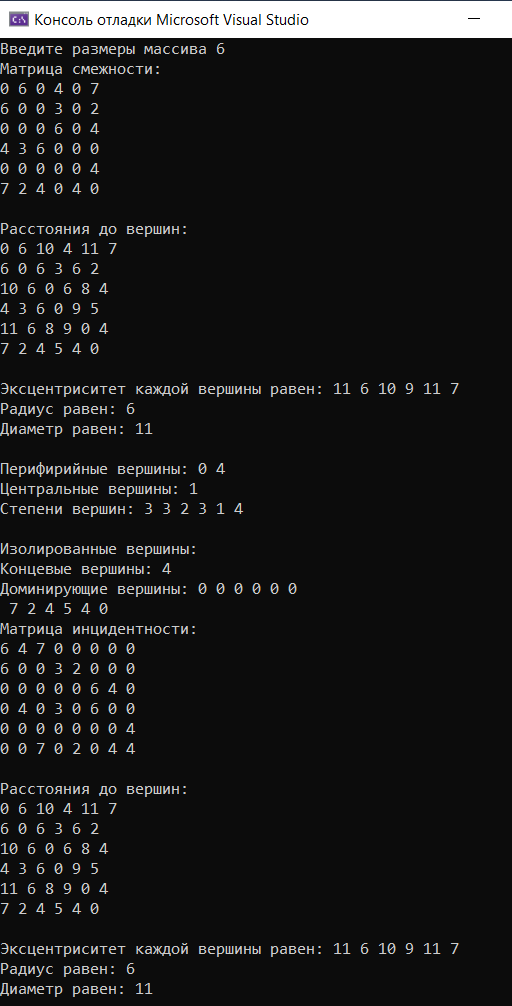
printf("\nЦентральные вершины: ");

for (n = 0; n < size; n++)

if (ex[n] == rad) printf("%d ", n);

}

Результат работы программы



**Вывод:** Сегодня удалось написать алгоритм для определения характеристик графов в матрице смежности и инцидентной матрице.