Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительной техники»

**Отчет**

По лабораторной работе №8

По курсу «Логика и основы алгоритмизации в инженерных задачах»

На тему «Определение характеристик графов»

Выполнили студенты гр.20ВВ4

Филиппов О.Р.

Кузнецов Н.С.

Проверили:

Юрова О.В

Акифьев И.В.

Пенза, 2021

**Задание:**

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.

1. Найдите изолированные, концевые и доминирующие вершины.

**Описание алгоритма:**

Необходимо выполнить стандартный поиск расстояний в графе для каждой вершины и сохранить наибольшее значение (эксцентриситет). Далее из полученных эксцентриситетов найти наибольший (диаметр) и наименьший (радиус). Сравнивая эксцентриситеты вершин с этими значениями выделить центральные и периферийные вершины.

Для определения изолированных, доминирующих и концевых вершин необходимо сложить числа в каждой строке матрицы смежности и сравнить полученные значения с нулем (изолированная), единицей (концевая), и уменьшенным на единицу числом вершин графа (доминирующая).

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<locale.h>

#include<time.h>

using namespace std;

#include<queue>

int BFC\_R(int\*\* matrix, int size, int start);

int\* stepen\_versh(int\*\* matrix, int size);

void main()

{

int\*\* matrix = NULL;

int\* mas = NULL;

int size = 0;

int i = 0, j = 0;

setlocale(LC\_ALL, "rus");

printf("Введите количество вершин: ");

scanf("%d", &size);

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

for (i = 0; i < size; i++) matrix[i] = (int\*)malloc(size \* sizeof(int));

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

srand(time(NULL));

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

{

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

{

if (i == j)

{

matrix[i][j] = 0;

continue;

}

if (rand() % 21 < 8) matrix[i][j] = 0;

else matrix[i][j] = rand()%10;

matrix[j][i] = matrix[i][j];

}

}

printf("Матрица:\n");

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

{

printf("%d|", i);

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

{

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

}

printf("\n");

}

for (i = 0; i < size; i++) mas[i] = BFC\_R(matrix, size, i);

int max = 0, min = 1001;

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

{

if (mas[i] > max) max = mas[i];

}

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

{

if (mas[i]<min) min = mas[i];

}

printf("Радиус = %d; диаметр = %d\n", min, max);

//Переферийные

printf("Переферийные вершины:\n");

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

{

if (mas[i] == max) printf("%d ",i);

}

//Центральные

printf("\nЦентральные вершины:\n");

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

{

if (mas[i] == min) printf("%d ", i);

}

printf("\n");

int\* stepen = stepen\_versh(matrix, size);

printf("Изолированные вершины:\n");

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

{

if (stepen[i] == 0) printf("%d ", i);

}

printf("\nКонцевые вершины:\n");

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

{

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

}

printf("\nДоминирующие вершины:\n");

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

{

if (stepen[i] == (size-1)) printf("%d ", i);

}

for (i = 0; i < size; i++) free(matrix[i]);

free(mas);

free(matrix);

}

int BFC\_R(int\*\* matrix, int size, int start)

{

queue <int> Q;

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

for (int i = 0; i < size; i++) distanse[i] = 1000;

distanse[start] = 0;

int visited;

int max=0;

int i = 0;

Q.push(start);

while (!Q.empty())

{

visited = Q.front();

Q.pop();

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

{

if (matrix[i][visited] > 0 && distanse[i] > distanse[visited] + matrix[i][visited])

{

Q.push(i);

distanse[i] = distanse[visited] + matrix[i][visited];

}

}

}

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

{

if (distanse[i] > max) max = distanse[i];

}

free(distanse);

return max;

}

int\* stepen\_versh(int\*\* matrix, int size)

{

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

int i = 0, j = 0;

for (i = 0; i < size; i++) D[i] = 0;

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

{

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

{

if (matrix[i][j] != 0) D[i] = D[i] + 1;

}

}

return D;

}

**Пояснения:**

Функция BFC\_R выполняет стандартный поиск расстояний в графе, после чего находит наибольшее значение и возвращает его. Функция вызывается для каждой вершины, возвращаемое значение записывается в массив. В итоге получаем массив. В котором каждой вершине (индекс) соответствует эксцентриситет. Находим наибольшее и наименьшее значение массива (радиус и диаметр). Сравнивая значения массива с полученными цифрами находим периферийные и центральные вершины.

Функция stepen\_versh() вычисляет степени всех вершин и заносит данные в массив, возвращает указатель на него. Изолированные, концевые и доминирующие вершины находятся путем сравнения элементов массива и нулем, единицей и уменьшенным на единицу количеством вершин графа (так как граф без петель).

**Вывод:** мы научились определять эксцентриситеты вершин графа, радиус и диаметр, выделять множества доминирующих, изолированных, концевых, периферийных и центральных вершин.