Министерство науки и высшего образования Российской федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЁТ**

по лабораторной работе №5

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Определение характеристик графов»

Выполнил ст. группы 22ВВВ1:

Николаев А.А.

Приняли:

К.э.н., доцент Акифьев И. В.

К.т.н., доцент Юрова О. В.

Пенза 2023

**Цель работы:**

Цель данной лабораторной работы заключается в изучении определений характеристик графов.

**Лабораторное задание:**

**Задание 1:**

1. Сгенерируйте матрицу смежности для неориентированного графа G. Выведите матрицу на экран.

2.Определите размер графа G, используя матрицу смежности графа.

3.Найдите изолированные, концевые и доминирующие вершины.

**Задание 2:**

1.Постройте для графа G матрицу инцидентности.

2. Определите размер графа G, используя матрицу инцидентности графа.

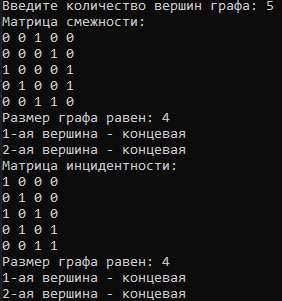
3. Найдите изолированные, концевые и доминирующие вершины.

**Ход работы:**

Создаем первый двумерный динамический массив. Обнуляем его диагональ. Заполняем верхнюю половину рандомно единицами и нулями. Отражаем на нижнюю часть. Считаем количество единиц в верхней части, для того чтобы определить размер графа. Проверяем количество единиц в строчке, чтобы определить какие вершины. Создает второй динамический массив. Строим его на основе первой матрицы. Так же выводим его размер и типы вершин.

**Результаты работы программы:**

Задание 1 и 2:



**Вывод:**

В ходе лабораторной работы научились создавать матрицы смежности и инцидентности, реализовали функцию определения типа вершины.

**Листинг:**

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

void main() {

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

int\*\* smej = 0; // матрица смежности

int\*\* inc = 0; // Матрица инцидентности

int i, j, ver, k, Count = 0, g;

cout << "Введите количество вершин графа: ";

cin >> ver;

smej = new int\* [ver];

if (ver == NULL)

{

cout << "Не удалось выделить память!" << endl;

return;

}

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

{

smej[i] = new int[ver];

}

// генерировать случайные значения для матрицы смежности

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

{

for (j = i; j < ver; j++)

{

if (i == j)

{

smej[i][j] = 0; // на главной диагонали нули

}

else

{

smej[i][j] = rand() % 2; // случайные значения 0 или 1

smej[j][i] = smej[i][j]; // симметрично заполнять значения для неориентированного графа

}

}

}

// выводить матрицу смежности на экран

cout << "Матрица смежности:" << endl;

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

{

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

{

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

}

cout << endl;

}

//Счетчик ребер

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

{

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

{

if (smej[i][j] == 1)

{

Count++;

}

}

}

//Данные по вершинам

cout << "Размер графа равен: " << Count << endl;

int\* A = 0;

A = new int[ver];

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

{

A[i] = 0;

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

{

if (smej[i][j] == 1)

{

A[i]++;

}

}

if (A[i] == 1)

{

cout << i + 1 << "-ая вершина - концевая" << endl;

}

else if (A[i] == ver - 1)

{

cout << i + 1 << "-ая вершина - доминирующая" << endl;

}

else if (A[i] == 0)

{

cout << i + 1 << "-ая вершина - изолированная" << endl;

}

}

delete[] A;

//Выделение памяти

inc = new int\* [ver];

if (inc == NULL)

{

cout << "Не удалось выделить память!" << endl;

return;

}

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

{

inc[i] = new int[Count];

}

g = Count;

// построение матрицы инцидентности

Count = 0;

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

{

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

{

if (smej[i][j] == 1)

{

for (k = 0; k < ver; k++)

{

if (k == i || k == j)

{

inc[k][Count] = 1;

}

else

{

inc[k][Count] = 0;

}

}

Count++;

}

}

}

// вывод матрицы инцидентности на экран

cout << "Матрица инцидентности:" << endl;

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

{

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

{

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

}

cout << endl;

}

//Данные по вершинам

cout << "Размер графа равен: " << Count << endl;

int\* Ar;

Ar = new int[ver];

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

{

Ar[i] = 0;

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

{

if (inc[i][j] == 1)

Ar[i]++;

}

if (Ar[i] == 1)

{

cout << i + 1 << "-ая вершина - концевая" << endl;

}

else if (Ar[i] == ver - 1)

{

cout << i + 1 << "-ая вершина - доминирующая" << endl;

}

else if (Ar[i] == 0)

{

cout << i + 1 << "-ая вершина - изолированная" << endl;

}

}

delete[] Ar;

// освобождение памяти

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

{

delete[] smej[i];

delete[] inc[i];

}

delete[] smej;

delete[] inc;

}