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»

Выполнили ст. группы 22ВВВ1:

Уткин М.М.

Саветкин Д.Д.

Соколовский Е.В

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 (используя генератор случайных чисел) 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\*

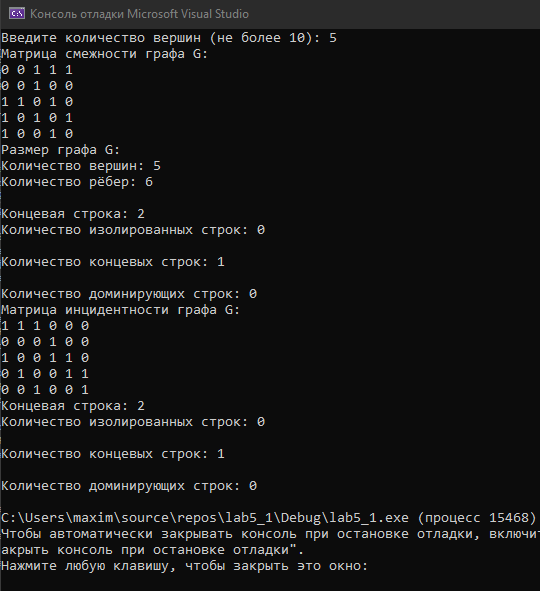
1. Постройте для графа G матрицу инцидентности.
2. Определите размер графа *G*, используя матрицу инцидентности графа.
3. Найдите изолированные, концевые и доминирующие вершины.

**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:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

#define MAX\_VERTICES 10

int main() {

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

int num\_vertices;

int adjacency\_matrix[MAX\_VERTICES][MAX\_VERTICES];

printf("Введите количество вершин (не более %d): ", MAX\_VERTICES);

scanf("%d", &num\_vertices);

if (num\_vertices <= 0 || num\_vertices > MAX\_VERTICES) {

printf("Неверное количество вершин. Завершение программы.\n");

return 1;

}

for (int i = 0; i < num\_vertices; i++) {

for (int j = 0; j < num\_vertices; j++) {

if (i == j) {

adjacency\_matrix[i][j] = 0;

}

else {

adjacency\_matrix[i][j] = rand() % 2;

adjacency\_matrix[j][i] = adjacency\_matrix[i][j];

}

}

}

printf("Матрица смежности графа G:\n");

for (int i = 0; i < num\_vertices; i++) {

for (int j = 0; j < num\_vertices; j++) {

printf("%d ", adjacency\_matrix[i][j]);

}

printf("\n");

}

int num\_edges = 0;

for (int i = 0; i < num\_vertices; i++) {

for (int j = i + 1; j < num\_vertices; j++) {

if (adjacency\_matrix[i][j] == 1) {

num\_edges++;

}

}

}

printf("Размер графа G:\n");

printf("Количество вершин: %d\n", num\_vertices);

printf("Количество рёбер: %d\n", num\_edges);

printf("\n");

// Определение изолированных, концевых и доминирующих строк

int isolated\_rows = 0;

int leaf\_rows = 0;

int dominating\_rows = 0;

for (int i = 0; i < num\_vertices; i++) {

int degree = 0;

for (int j = 0; j < num\_vertices; j++) {

degree += adjacency\_matrix[i][j];

}

if (degree == 0) {

printf("Изолированная строка: %d\n", i + 1);

isolated\_rows++;

}

else if (degree == 1) {

printf("Концевая строка: %d\n", i + 1);

leaf\_rows++;

}

else if (degree >= num\_vertices - 1) {

printf("Доминирующая строка: %d\n", i + 1);

dominating\_rows++;

}

}

printf("Количество изолированных строк: %d\n", isolated\_rows);

printf("\n");

printf("Количество концевых строк: %d\n", leaf\_rows);

printf("\n");

printf("Количество доминирующих строк: %d\n", dominating\_rows);

int incidence\_matrix[MAX\_VERTICES][MAX\_VERTICES];

for (int i = 0; i < num\_vertices; i++) {

for (int j = 0; j < num\_edges; j++) {

incidence\_matrix[i][j] = 0;

}

}

int edge\_count = 0;

for (int i = 0; i < num\_vertices; i++) {

for (int j = i + 1; j < num\_vertices; j++) {

if (adjacency\_matrix[i][j] == 1) {

incidence\_matrix[i][edge\_count] = 1;

incidence\_matrix[j][edge\_count] = 1;

edge\_count++;

}

}

}

printf("Матрица инцидентности графа G:\n");

for (int i = 0; i < num\_vertices; i++) {

for (int j = 0; j < num\_edges; j++) {

printf("%d ", incidence\_matrix[i][j]);

}

printf("\n");

}

// Определение изолированных, концевых и доминирующих строк

int isolated\_rows2 = 0;

int leaf\_rows2 = 0;

int dominating\_rows2 = 0;

for (int i = 0; i < num\_vertices; i++) {

int degree = 0;

for (int j = 0; j < num\_vertices; j++) {

degree += adjacency\_matrix[i][j];

}

if (degree == 0) {

printf("Изолированная строка: %d\n", i + 1);

isolated\_rows2++;

}

else if (degree == 1) {

printf("Концевая строка: %d\n", i + 1);

leaf\_rows2++;

}

else if (degree >= num\_vertices - 1) {

printf("Доминирующая строка: %d\n", i + 1);

dominating\_rows2++;

}

}

printf("Количество изолированных строк: %d\n", isolated\_rows2);

printf("\n");

printf("Количество концевых строк: %d\n", leaf\_rows2);

printf("\n");

printf("Количество доминирующих строк: %d\n", dominating\_rows2);

return 0;

}