Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧËТ**  
по лабораторной работе №5  
по дисциплине: «Логика и основы алгоритмизации в инженерных задачах»

На тему: «**Определение характеристик графов**»

Выполнил:

студент группы 22ВВC1

Пилюгин Артём

Принял:

Юрова О.В

Акифьев И.В

Пенза

2023

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

**Задание №1**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <clocale>

#define MAX\_VERTICES 10 // Максимальное количество вершин в графе

int main() {

setlocale(LC\_ALL, "Russian");

int n; // Количество вершин

int i, j;

int adjacencyMatrix[MAX\_VERTICES][MAX\_VERTICES];

// Инициализация генератора случайных чисел

srand(time(NULL));

// Введите количество вершин графа

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

scanf("%d", &n);

// Генерация матрицы смежности

printf("Матрица смежности графа:\n");

for (i = 0; i < n; i++) {

for (j = i; j < n; j++) {

if (i == j) {

adjacencyMatrix[i][j] = 0; // Нет петли у вершины

}

else {

int random = rand() % 2; // Случайное 0 или 1 для определения наличия ребра

adjacencyMatrix[i][j] = random;

adjacencyMatrix[j][i] = random; // Граф неориентированный, поэтому симметричен

}

}

}

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

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

}

printf("\n");

}

// Определение размера графа

int count = 0;

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

{

for (int j = i; j < n; j++)

{

if (adjacencyMatrix[i][j] == 0)

continue;

else

count++;

}

}

printf("Размер графа: %d \n", count);

// Поиск изолированных, концевых и доминирующих вершин

int isolatedVertices = 0;

int leafVertices = 0;

int dominatingVertices = 0;

for (i = 0; i < n; i++) {

int degree = 0;

for (j = 0; j < n; j++) {

if (adjacencyMatrix[i][j] == 1) {

degree++;

}

}

if (degree == 0) {

isolatedVertices++;

}

else if (degree == 1) {

leafVertices++;

}

if (degree == n - 1) {

dominatingVertices++;

}

}

printf("Изолированных вершин: %d\n", isolatedVertices);

printf("Концевых вершин: %d\n", leafVertices);

printf("Доминирующих вершин: %d\n", dominatingVertices);

return 0;

}

**Задание №2**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <clocale>

#define MAX\_VERTICES 10 // Максимальное количество вершин в графе

#define MAX\_EDGES 20 // Максимальное количество рёбер в графе

int main() {

setlocale(LC\_ALL, "Russian");

int n; // Количество вершин

int m; // Количество рёбер

int i, j, k;

int incidenceMatrix[MAX\_VERTICES][MAX\_EDGES];

// Инициализация генератора случайных чисел

srand(time(NULL));

// Введите количество вершин и рёбер графа

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

scanf("%d", &n);

printf("Введите количество рёбер в графе (не более %d): ", MAX\_EDGES);

scanf("%d", &m);

// Генерация матрицы инцидентности

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

for (i = 0; i < n; i++) {

for (j = 0; j < m; j++) {

incidenceMatrix[i][j] = 0; // Инициализируем все значения нулем

}

}

for (k = 0; k < m; k++) {

printf("Введите номера вершин, инцидентных ребру %d (0-%d): ", k, n - 1);

int vertex1, vertex2;

scanf("%d %d", &vertex1, &vertex2);

incidenceMatrix[vertex1][k] = 1;

incidenceMatrix[vertex2][k] = 1;

}

// Вывод матрицы инцидентности

for (i = 0; i < n; i++) {

for (j = 0; j < m; j++) {

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

}

printf("\n");

}

// Определение размера графа

printf("Размер графа: %d вершин, %d рёбер\n", n, m);

// Поиск изолированных, концевых и доминирующих вершин

int isolatedVertices = 0;

int leafVertices = 0;

int dominatingVertices = 0;

for (i = 0; i < n; i++) {

int degree = 0;

for (j = 0; j < m; j++) {

if (incidenceMatrix[i][j] == 1) {

degree++;

}

}

if (degree == 0) {

isolatedVertices++;

}

else if (degree == 1) {

leafVertices++;

}

if (degree == n - 1) {

dominatingVertices++;

}

}

printf("Изолированных вершин: %d\n", isolatedVertices);

printf("Концевых вершин: %d\n", leafVertices);

printf("Доминирующих вершин: %d\n", dominatingVertices);

return 0;

}

**Результат работы программ**

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

**Рисунок 1 - Результат работы программы LR5A**

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

**Рисунок 2 - Результат работы программы LR5B**

**Вывод**

В результате выполнения лабораторной работы были успешно реализованы алгоритмы построения матриц смежности и инциндентности, а также поиск изолированных, концевых и доминирующих вершин. Научились на практике находить глубину графа G, используя матрицу смежности графа, аналогичные действия были совершены с матрицей инцидентности.