Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №5

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Определение характеристик графов»

Выполнили:

Студенты группы 23ВВВ2

Герасимов В.Р.

Мадамкин В. М.

Приняли:

Юрова О. В.

Митрохин М. А.

Пенза 2024

**Лабораторное задание**

**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**

Задание 1:

Постройте для графа G матрицу инцидентности.

1. Сначала подсчитывается количество ребер m, проходя по верхней треугольной части матрицы смежности.

2.Далее выделяется память для матрицы инцидентности размером n x m.

3.Матрица инцидентности заполняется: если есть ребро между вершинами i и j, то соответствующие элементы матрицы устанавливаются в 1.

Задание 2:

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

1.Размер графа size определяется как количество ребер m, которое было подсчитано при построении матрицы инцидентности.

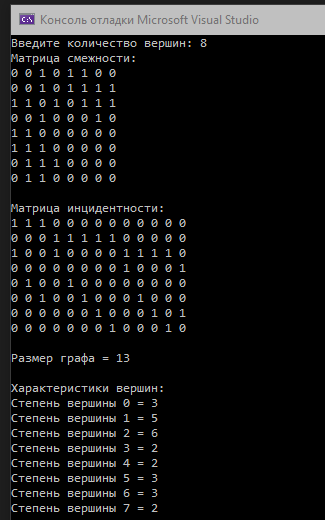
Задание 3:

Найдите изолированные, концевые и доминирующие вершины.

Сначала подсчитываются степени вершин. Для каждой вершины i суммируем все значения в строке i матрицы смежности.

Затем выводим характеристики каждой вершины:

* Если степень вершины равна 0, она изолированная.
* Если степень вершины равна 1, она концевая.
* Если степень вершины равна n-1, она доминирующая.



**Вывод:** мы на практике смогли построить матрицу инцидентности, определили размер графа используя матрицу инцидентности, и нашли изолированные, концевые и доминирующие вершины.

**Приложение A**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale>

int main() {

setlocale(LC\_ALL, "Russian");

int n = 10, \*\* G = NULL, size = 0, \* deg = NULL, m = 0;

int\*\* incidenceMatrix = NULL;

printf("Введите количество вершин: ");

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

// Выделение памяти для матрицы смежности

G = (int\*\*)malloc(n \* sizeof(int\*));

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

G[i] = (int\*)malloc(n \* sizeof(int));

}

// Выделение памяти для массива степеней вершин

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

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

deg[i] = 0;

}

// Инициализация генератора случайных чисел

srand(time(NULL));

// Заполнение матрицы смежности

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

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

G[i][j] = rand() % 2; // Случайное значение 0 или 1

G[j][i] = G[i][j]; // Симметричное заполнение

// Обнуление диагональных элементов

if (i == j) {

G[i][j] = 0;

}

}

}

// Подсчет количества ребер для создания матрицы инцидентности

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

for (int j = i + 1; j < n; j++) {

if (G[i][j] != 0) {

m++;

}

}

}

// Выделение памяти для матрицы инцидентности

incidenceMatrix = (int\*\*)malloc(n \* sizeof(int\*));

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

incidenceMatrix[i] = (int\*)malloc(m \* sizeof(int));

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

incidenceMatrix[i][j] = 0;

}

}

// Заполнение матрицы инцидентности

int edgeIndex = 0;

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

for (int j = i + 1; j < n; j++) {

if (G[i][j] != 0) {

incidenceMatrix[i][edgeIndex] = 1;

incidenceMatrix[j][edgeIndex] = 1;

edgeIndex++;

}

}

}

// Вывод матрицы смежности

printf("Матрица смежности:\n");

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

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

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

}

printf("\n");

}

// Вывод матрицы инцидентности

printf("\nМатрица инцидентности:\n");

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

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

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

}

printf("\n");

}

// Подсчет размера графа, используя матрицу инцидентности

size = m;

printf("\nРазмер графа = %d\n", size);

// Подсчет степеней вершин

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

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

if (G[i][j] != 0) {

deg[i]++;

}

}

}

// Вывод степеней вершин и их характеристик

printf("\nХарактеристики вершин:\n");

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

printf("Степень вершины %d = %d\n", i, deg[i]);

if (deg[i] == 0) {

printf("Вершина %d - изолированная\n", i);

}

if (deg[i] == 1) {

printf("Вершина %d - концевая\n", i);

}

if (deg[i] == n - 1) {

printf("Вершина %d - доминирующая\n", i);

}

}

// Освобождение памяти

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

free(G[i]);

free(incidenceMatrix[i]);

}

free(G);

free(incidenceMatrix);

free(deg);

return 0;

}

.