Министерство науки и высшего 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 23ВВВ4

Брагин А.М.

Зарубин Я.

Герасимов К.

Приняли:

Деев М.В.

Юрова О.В.

Пенза 2024

**Общие сведения.**

Если G – граф (рисунок 1), содержащий непустое множество n вершин

V и множество ребер E, где e(v i , v j ) – ребро между двумя произвольными вершинами v i  и v j , тогда размер графа G есть мощность множества ребер |E(G)| или, количество ребер графа.

Степенью вершины графа G называется число инцидентных ей ребер.

Степень вершины v i обозначается через deg(v i ).

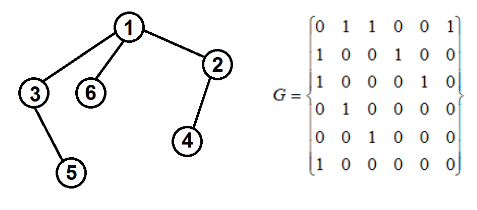


Рисунок 1 – Граф

Вершина v i со степенью 0 называется изолированной, со степенью 1 –

концевой.

Вершина графа, смежная с каждой другой его вершиной, называется доминирующей.

**Практическая часть:**

**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.

Листинг программы

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

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

#define MAX\_EDGES 45 // Максимальное количество рёбер для графа с 10 вершинами (n\*(n-1)/2 для неориентированного графа)

void generateAdjacencyMatrix(int n, int graph[MAX\_VERTICES][MAX\_VERTICES]) {

// Генерация матрицы смежности

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

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

if (i == j) {

graph[i][j] = 0; // Нет петли

} else {

graph[i][j] = rand() % 2; // Случайное ребро

graph[j][i] = graph[i][j]; // Граф неориентированный

}

}

}

}

void printAdjacencyMatrix(int n, int graph[MAX\_VERTICES][MAX\_VERTICES]) {

// Вывод матрицы смежности

printf("Матрица смежности:\n");

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

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

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

}

printf("\n");

}

}

int graphSize(int n, int graph[MAX\_VERTICES][MAX\_VERTICES]) {

// Определение размера графа (количество рёбер)

int edges = 0;

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

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

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

edges++;

}

}

}

return edges;

}

void findSpecialVertices(int n, int graph[MAX\_VERTICES][MAX\_VERTICES]) {

int degree;

int isolated = 0, pendant = 0, dominating = 0;

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

degree = 0;

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

degree += graph[i][j];

}

if (degree == 0) {

printf("Вершина %d: изолированная\n", i + 1);

isolated++;

} else if (degree == 1) {

printf("Вершина %d: концевая\n", i + 1);

pendant++;

} else if (degree == n - 1) {

printf("Вершина %d: доминирующая\n", i + 1);

dominating++;

}

}

printf("\nВсего: изолированных вершин = %d, конечных = %d, доминирующих = %d\n", isolated, pendant, dominating);

}

void generateIncidenceMatrix(int n, int graph[MAX\_VERTICES][MAX\_VERTICES], int incidence[MAX\_VERTICES][MAX\_EDGES], int \*edgeCount) {

// Построение матрицы инцидентности

int edgeIndex = 0;

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

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

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

incidence[i][edgeIndex] = 1;

incidence[j][edgeIndex] = 1;

edgeIndex++;

}

}

}

\*edgeCount = edgeIndex;

}

void printIncidenceMatrix(int n, int edgeCount, int incidence[MAX\_VERTICES][MAX\_EDGES]) {

// Вывод матрицы инцидентности

printf("Матрица инцидентности:\n");

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

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

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

}

printf("\n");

}

}

int main() {

srand(time(0));

int n;

printf("Введите количество вершин графа: ");

scanf("%d", &n);

if (n > MAX\_VERTICES) {

printf("Количество вершин не может превышать %d\n", MAX\_VERTICES);

return 1;

}

int graph[MAX\_VERTICES][MAX\_VERTICES];

int incidence[MAX\_VERTICES][MAX\_EDGES] = {0}; // Инициализация матрицы инцидентности нулями

int edgeCount;

// Генерация матрицы смежности

generateAdjacencyMatrix(n, graph);

// Вывод матрицы смежности

printAdjacencyMatrix(n, graph);

// Определение размера графа

int edges = graphSize(n, graph);

printf("Размер графа (количество рёбер): %d\n", edges);

// Нахождение изолированных, конечных и доминирующих вершин

findSpecialVertices(n, graph);

// Генерация и вывод матрицы инцидентности

generateIncidenceMatrix(n, graph, incidence, &edgeCount);

printIncidenceMatrix(n, edgeCount, incidence);

return 0;

}

**Результат работы программы:**

