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 21ВВ1.2

Брянцев Артём  
Сущёв Максим  
Тюрин Владислав

Приняли:

Митрохин М. А.

Юрова О. В.

Пенза 2022

**Название**

Определение 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.

**Листинг**

lb5.1.cpp

#include <iostream>

int sizeGraph(int\*\* g, int n); //размер графа (кол-во) рёбер

void createMatrix(int\*\* g, int n); //создание матрицы

void createMatrix(int\*\* g1, int\*\* g2, int n1, int n2);

void printMatrix(int\*\* g, int n); //вывод матрицы

void printMatrix(int\*\* g, int n, int num);

void freedom(int\*\* g, int n); //освобждение памяти для двумерного массива (матрицы)

void searchIsolatedVertices(int\*\* g, int n); //поиск изолированных вершин

void searchEndVertices(int\*\* g, int n); //поиск концевых вершин

void searchDominantVertices(int\*\* g, int n); //поиск доминирующих вершин

void sizeGraph(int\*\* g, int n, int num); //размер графа (кол-во) рёбер

void searchIsolatedVertices(int\*\* g, int n, int num); //поиск изолированных вершин

void searchEndVertices(int\*\* g, int n, int num); //поиск концевых вершин

void searchDominantVertices(int\*\* g, int n, int num); //поиск доминирующих вершин

using namespace std;

int main()

{

setlocale(LC\_ALL, "");

srand(time(NULL));

int\*\* g1, \*\* g2, n, num\_edge;

cout << "Введите количество вершин для графов: ";

cin >> n;

cout << endl;

g1 = (int\*\*)malloc(n \* sizeof(int\*)); //выделяем память под столбец указателей

createMatrix(g1, n);

printMatrix(g1, n);

num\_edge = sizeGraph(g1, n);

searchIsolatedVertices(g1, n);

searchEndVertices(g1, n);

searchDominantVertices(g1, n);

g2 = (int\*\*)malloc(n \* sizeof(int\*));

cout << endl;

createMatrix(g1, g2, n, num\_edge);

printMatrix(g2, n, num\_edge);

cout << "Размер графа = " << num\_edge << endl;

searchIsolatedVertices(g2, n, num\_edge);

searchEndVertices(g2, n, num\_edge);

searchDominantVertices(g2, n, num\_edge);

freedom(g1, n);

freedom(g2, n);

}

void createMatrix(int\*\* g, int n) {

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

\*(g + i) = (int\*)malloc(n \* sizeof(int)); //выделяем память под значения столбцов

}

for (int i = 0; i < n; i++) // цикл по строкам

{

for (int j = i; j < n; j++) // цикл по столбцам строки

{

if (i == j) {

\*(\*(g + i) + j) = 0;

}

else {

\*(\*(g + i) + j) = rand() % 2;

\*(\*(g + j) + i) = \*(\*(g + i) + j);

}

}

}

}

void createMatrix(int\*\* g1, int\*\* g2, int n1, int n2) {

for (int i = 0; i < n1; i++) {

\*(g2 + i) = (int\*)malloc(n2 \* sizeof(int)); //выделяем память под значения столбцов

for (int j = 0; j < n2; j++) // цикл по столбцам строки

{

g2[i][j] = 0;

}

}

n2 = 0;

for (int i = 0; i < n1; i++) // цикл по строкам

{

for (int j = i; j < n1; j++) // цикл по столбцам строки

{

if (g1[i][j] == 1) {

g2[i][n2] = 1;

g2[j][n2] = 1;

n2++;

}

}

}

}

void printMatrix(int\*\* g, int n) {

cout << " ";

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

cout.width(3);

cout << i;

}

cout << endl << endl;

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

cout << i << " ";

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

{

cout.width(3);

cout << \*(\*(g + i) + j);

}

cout << endl;

}

cout << endl << endl;

}

void printMatrix(int\*\* g, int n, int num) {

cout << " ";

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

cout.width(3);

cout << i;

}

cout << endl << endl;

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

cout << i << " ";

for (int j = 0; j < num; j++)

{

cout.width(3);

cout << \*(\*(g + i) + j);

}

cout << endl;

}

cout << endl << endl;

}

void freedom(int\*\* g, int n) {

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

free(\*(g + i)); //освобождаем память

}

free(g); //освобождаем память

}

int sizeGraph(int\*\* g, int n) {

int size = 0;

for (int i = 0; i < n; i++) // цикл по строкам

{

for (int j = i; j < n; j++) // цикл по столбцам строки

{

if (g[i][j] == 1) size++;

}

}

cout << "Размер графа = " << size << endl;

return size;

}

void searchIsolatedVertices(int\*\* g, int n) {

int k = 0;

for (int i = 0; i < n; i++) // цикл по строкам

{

for (int j = 0; j < n; j++) // цикл по столбцам строки

{

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

k++;

break;

}

}

if (k == 0) cout << i << " - изолированная вершина" << endl;

k = 0;

}

}

void searchEndVertices(int\*\* g, int n) {

int k = 0;

for (int i = 0; i < n; i++) // цикл по строкам

{

for (int j = 0; j < n; j++) // цикл по столбцам строки

{

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

k++;

}

}

if (k == 1) cout << i << " - концевая вершина" << endl;

k = 0;

}

}

void searchDominantVertices(int\*\* g, int n) {

int k = 0;

for (int i = 0; i < n; i++) // цикл по строкам

{

for (int j = 0; j < n; j++) // цикл по столбцам строки

{

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

k++;

}

}

if (k == n - 1) cout << i << " - доминирующая вершина" << endl;

k = 0;

}

}

void searchIsolatedVertices(int\*\* g, int n, int num) {

int k = 0;

for (int i = 0; i < n; i++) // цикл по строкам

{

for (int j = 0; j < num; j++) // цикл по столбцам строки

{

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

k++;

break;

}

}

if (k == 0) cout << i << " - изолированная вершина" << endl;

k = 0;

}

}

void searchEndVertices(int\*\* g, int n, int num) {

int k = 0;

for (int i = 0; i < n; i++) // цикл по строкам

{

for (int j = 0; j < num; j++) // цикл по столбцам строки

{

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

k++;

}

}

if (k == 1) cout << i << " - концевая вершина" << endl;

k = 0;

}

}

void searchDominantVertices(int\*\* g, int n, int num) {

int k = 0;

for (int i = 0; i < n; i++) // цикл по строкам

{

for (int j = 0; j < num; j++) // цикл по столбцам строки

{

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

k++;

}

}

if (k == n - 1) cout << i << " - доминирующая вершина" << endl;

k = 0;

}

}

**Результат работы программы**

Результаты работы программы (задание 1 и 2) показаны на рисунке 1.

Изображение выглядит как текст

Автоматически созданное описание**Рисунок 1 — Результаты работы программы**

### Выводы

Граф – фигура, состоящая из заданных точек (вершин), соединенных отрезками, которые, называются ребрами (дугами) графа.

Вершина называется изолированной, если она не является концом ни для одного ребра.

Вершина называется концевой, если она является концом ровно одного ребра.

Вершина графа смежная с каждой другой, называется доминирующей.