МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ УПРАВЛІННЯ

ЗВІТ

З ЛАБОРАТОРНОЇ РОБОТИ №6

ЗА ТЕМОЮ “ФУНДАМЕНТАЛЬНІ АЛГОРИТМИ НА ГРАФАХ І ДЕРЕВАХ”

Виконав студент

групи КН-221д

Габорець

Перевірив

Солонська С.В.

Харків 2022

ЛАБОРАТОРНА РОБОТА №6

Тема: ФУНДАМЕНТАЛЬНІ АЛГОРИТМИ НА ГРАФАХ І ДЕРЕВАХ

Мета роботи: познайомитися зі способами представлення графів та отримати навички програмування алгоритмів, що їх обробляють

Завдання :

Розробити програму, яка читає з клавіатури числа N, M (1 < N, M < 256) — кількість вершин та ребер графу; послідовність M пар цілих чисел - ребра графу. Програма зберігає граф та виконує над ним алгоритм згідно варіанту

Варіанти представлення графів

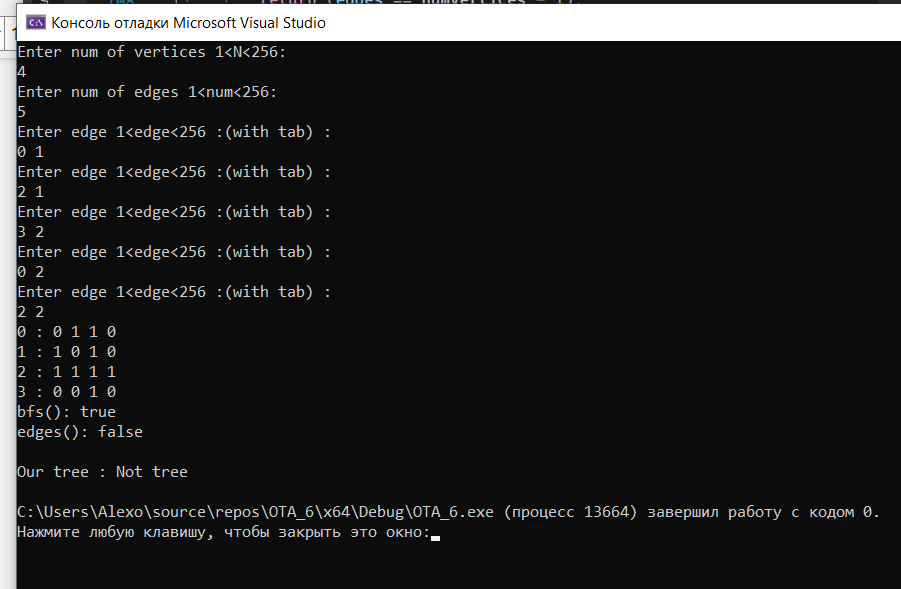
1 Матриця суміжності.

Варіанти алгоритмів:

4 Визначити, чи є заданий граф деревом або лісом.

Варіанти завдань

(1.4)



КОД:

#include<iostream>

#include<vector>

#include<queue>

//Дерево - связный граф с n-1 ребром.

//Варіанти представлення графів.

//1 Матриця смежности.

//Варіанти алгоритмів.

//4 Визначити, чи є заданий граф деревом або лісом.

using namespace std;

class Graph {

private:

bool\*\* adjMatrix;

int numVertices;

public:

Graph(int numVertices) { // Initialize the matrix to zero

this->numVertices = numVertices;

adjMatrix = new bool\* [numVertices];

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

adjMatrix[i] = new bool[numVertices];

for (int j = 0; j < numVertices; j++)// Initialize all array with false

adjMatrix[i][j] = false;

}

}

void addEdge(int i, int j) {// + Rebro(dyga)

adjMatrix[i][j] = true;

adjMatrix[j][i] = true;

}

//void removeEdge(int i, int j) {//- Rebro(dyga)

// adjMatrix[i][j] = false;

// adjMatrix[j][i] = false;

//}

void toString() {

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

cout << i << " : ";

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

cout << adjMatrix[i][j] << " ";

cout << "\n";

}

}

~Graph() {

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

delete[] adjMatrix[i];

delete[] adjMatrix;

}

///////////////////////////

//поиск в ширину

bool bfs()

{

std::queue<int> que;

std::vector<bool> vec(numVertices);

for (auto val : vec)

{

val = false;

}

int cur = 2;

que.push(cur);

vec[cur] = true;

while (!que.empty())

{

cur = que.front();

que.pop();

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

{

if (1 == adjMatrix[cur][j] && vec.at(j) != true)

{

vec.at(j) = true;

que.push(j);

}

}

}

bool ret = true;

for (int i = 0; i < vec.size(); ++i)

{

if (vec[i] != true)

{

std::cout << "bfs() : several connectivity components\n";//несколько компонент связности

ret = false;

break;

}

}

return ret;

}

// число рёбер равно число вершин минус один

bool edges()

{

int edges = 0;

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

{

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

{

if (adjMatrix[i][j] == 1)

{

++edges;

}

}

}

edges /= 2;

return (edges == numVertices - 1);

}

};

int main() {

unsigned int n,m;

std::cout << "Enter num of vertices 1<N<256:\n"; //enter 4

std::cin >> n;

Graph g(n);//4//Tops(vershinu)

std::cout << "Enter num of edges 1<num(M)<256:\n"; //enter 5

std::cin >> m;

int i\_edge, j\_edge;

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

{

std::cout << "Enter edge 1<edge<256 :(with tab) :\n";

std::cin >> i\_edge >> j\_edge;

g.addEdge(i\_edge, j\_edge);

}

g.toString();

bool b = g.bfs();

bool e = g.edges();

cout << "bfs(): " << std::boolalpha << b << endl;

cout << "edges(): " << std::boolalpha << e << endl;

cout << "\nOur tree : " << (b && e ? "Tree" : "Not tree") << endl;

}

Висновок: виконавши лабораторну роботу №6, я познайомився зі способами представлення графів та отримав навички програмування алгоритмів, що їх обробляють.