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

Кукуєв Руслан Олександрович

Перевірила

Солонська С.В.

Харків 2022

ЛАБОРАТОРНА РОБОТА №6

Тема: ФУНДАМЕНТАЛЬНІ АЛГОРИТМИ НА ГРАФАХ І ДЕРЕВАХ.

Мета роботи: познайомитися з фундаментальними алгоритмами на графаг та деревах.

**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. 2 Список суміжності. Варіанти алгоритмів. 1 Пошук у ширину. На екран потрібно вивести вершини у порядку обходу. Для кожної вказати час прибуття та предка у дереві обходу. 2 Пошук у глибину. На екран потрібно вивести вершини у порядку обходу. Для кожної вказати час початку розгляду, кінця розгляду та предка у дереві обходу. 3 Топологічне сортування. На екран потрібно вивести ті ж дані, що і для пошуку в глибину, а також результат сортування. 4 Визначити, чи є заданий граф деревом або лісом. 5 Побудувати остовне дерево алгоритмом Прима. 6 Побудувати остовне дерево алгоритмом Крускала.

**Варіант №12**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

#define edge pair<int, int>

vector<pair<int, edge>> G;

vector<pair<int, edge>> T;

int \*parent;

void Init(int N)

{

parent = new int[N];

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

{

parent[i] = i;

}

G.clear();

T.clear();

}

int find\_set(int i)

{

if (i == parent[i])

return i;

else

return find\_set(parent[i]);

}

void union\_set(int u, int v)

{

parent[u] = parent[v];

}

void Kruskal()

{

int i, uRep, vRep;

sort(G.begin(), G.end());

for (i = 0; i < G.size(); i++)

{

uRep = find\_set(G[i].second.first);

vRep = find\_set(G[i].second.second);

if (uRep != vRep)

{

T.push\_back(G[i]);

union\_set(uRep, vRep);

}

}

}

void PrintGraphAdjacencyList()

{

cout << "\nGraph adjacency list:" << endl;

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

{

cout << T[i].second.first << " - " << T[i].second.second << " : " << T[i].first;

cout << endl;

}

}

int main()

{

int N, M;

cout << "Nodes: ";

cin >> N;

cout << "Edges: ";

cin >> M;

if (1 >= N || M > 256)

{

std::cout << "Incorrect value\n";

system("pause");

return 0;

}

Init(N);

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

{

int u, v, w;

cout << "Node 1, Node 2, Weight: ";

cin >> u >> v >> w;

G.push\_back(make\_pair(w, edge(u, v)));

}

Kruskal();

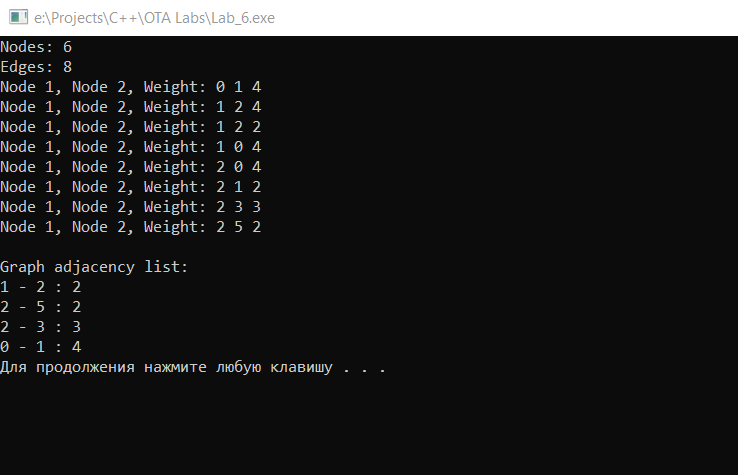
PrintGraphAdjacencyList();

system("pause");

return 0;

}

**Демонстрація роботи:**



**Висновок:** виконавши лабораторну роботу №6, я познайомився з фундаментальними алгоритмами на графаг та деревах.