CSE246

Section 4

Fall 2023

Group:06

**Lab Task - 06**

**Topic: Cycle Finding and Printing**

### **Maximal Connected Component in an Undirected Graph**

**Topological Sorting in Ascending order**

Submitted By

| **Name** | **ID** |
| --- | --- |
| B M Shahria Alam | 2021-3-60-016 |
| Golam Kibria | 2021-3-60-215 |
| Sidratul Moontaha | 2021-3-60-048 |
| MD Imran Khan | 2021-3-60-206 |

### **#Problem 1: Cycle Finding and Printing**

#include <bits/stdc++.h>

using namespace std;

const int MAX\_VERTICES = 100000;

vector<int> adj[MAX\_VERTICES];

bool visited[MAX\_VERTICES];

int parent[MAX\_VERTICES];

void initGraph(int V) {

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

adj[i].clear();

visited[i] = false;

parent[i] = -1;

}

}

void addEdge(int u, int v) {

adj[u].push\_back(v);

adj[v].push\_back(u);

}

bool dfs(int v, int par) {

visited[v] = true;

parent[v] = par;

for (int u : adj[v]) {

if (!visited[u]) {

if (dfs(u, v)) {

return true;

}

} else if (u != par) {

cout << "YES" << endl;

vector<int> cycle;

int cur = v;

while (cur != u) {

cycle.push\_back(cur);

cur = parent[cur];

}

cycle.push\_back(u);

for (int vertex : cycle) {

cout << vertex << " ";

}

cout << endl;

return true;

}

}

return false;

}

int main() {

int V, E;

cin >> V >> E;

initGraph(V);

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

int u, v;

cin >> u >> v;

addEdge(u, v);

}

for (int i = 1; i <= V; ++i) {

if (!visited[i]) {

if (dfs(i, -1)) {

return 0;

}

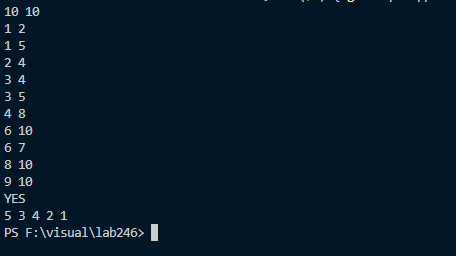
}

}

cout << "NO" << endl;

return 0;

}



### **#Problem 2: Maximal Connected Component in an Undirected Graph**

#include <bits/stdc++.h>

using namespace std;

const int MAX\_VERTICES = 1000;

vector<int> adj[MAX\_VERTICES];

bool visited[MAX\_VERTICES];

void initGraph(int V) {

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

adj[i].clear();

visited[i] = false;

}

}

void addEdge(int u, int v) {

adj[u].push\_back(v);

adj[v].push\_back(u);

}

int dfs(int v) {

visited[v] = true;

int size = 1;

for (int u : adj[v]) {

if (!visited[u]) {

size += dfs(u);

}

}

return size;

}

int main() {

int V, E;

cin >> V >> E;

initGraph(V);

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

int u, v;

cin >> u >> v;

addEdge(u, v);

}

int maxCompSize = 0;

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

if (!visited[i]) {

int compSize = dfs(i);

maxCompSize = max(maxCompSize, compSize);

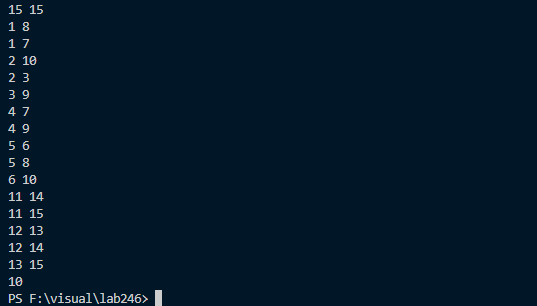
}

}

cout << maxCompSize << endl;

return 0;

}



### **#Problem 3: Topological Sorting in Ascending order**

#include <bits/stdc++.h>

using namespace std;

const int MAX\_VERTICES = 100000;

vector<int> adj[MAX\_VERTICES];

int inDegree[MAX\_VERTICES];

void addEdge(int u, int v) {

adj[u].push\_back(v);

inDegree[v]++;

}

void topologicalSort(int V) {

priority\_queue<int, vector<int>, greater<int>> pq;

for (int i = 1; i <= V; ++i) {

if (inDegree[i] == 0) {

pq.push(i);

}

}

while (!pq.empty()) {

int u = pq.top();

pq.pop();

cout << u << endl;

for (int v : adj[u]) {

inDegree[v]--;

if (inDegree[v] == 0) {

pq.push(v);

}

}

}

}

int main() {

int V, E;

cin >> V >> E;

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

int u, v;

cin >> u >> v;

addEdge(u, v);

}

topologicalSort(V);

return 0;

}

