**Strongly Connected Components (Tarjan’s Algorithm)**

#include <iostream>

#include <vector>

#include <stack>

using namespace std;

void dfs(int node, int &currentTime, vector<int> &disc, vector<int> &low, vector<bool> &inStack, stack<int> &st, vector<vector<int>> &adjList, vector<vector<int>> &scc)

{

disc[node] = low[node] = currentTime;

currentTime++;

st.push(node);

inStack[node] = true;

for (int neighbor : adjList[node])

{

if (disc[neighbor] == -1)

{

dfs(neighbor, currentTime, disc, low, inStack, st, adjList, scc);

low[node] = min(low[node], low[neighbor]);

}

else if (inStack[neighbor])

{

low[node] = min(low[node], disc[neighbor]);

}

}

if (low[node] == disc[node])

{

vector<int> currentSCC;

int top = -1;

while (top != node)

{

top = st.top();

st.pop();

inStack[top] = false;

currentSCC.push\_back(top);

}

scc.push\_back(currentSCC);

}

}

vector<vector<int>> stronglyConnectedComponents(int n, vector<vector<int>> &edges)

{

vector<vector<int>> adjList(n);

for (vector<int> &edge : edges)

{

int u = edge[0];

int v = edge[1];

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

}

vector<int> disc(n, -1);

vector<int> low(n, -1);

vector<bool> inStack(n, false);

stack<int> st;

vector<vector<int>> scc;

int currentTime = 0;

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

{

if (disc[i] == -1)

{

dfs(i, currentTime, disc, low, inStack, st, adjList, scc);

}

}

return scc;

}