1. **Topological Sort:**

#include<bits/stdc++.h>

using namespace std;

stack<int> Stack;

vector< int > adj[1000];

int t, node, visited[1000] = {0}, discover\_time[1000] , finish\_time[1000];

void topologicalSortUtil(int i){

t += 1;

discover\_time[i] = t;

visited[i] = 1;

for(int j=0; j<adj[i].size(); j++){

int v = adj[i][j];

if(visited[v] == 0){

topologicalSortUtil(v);

}

}

t += 1;

finish\_time[i] = t;

Stack.push(i);

}

void topologicalSort(){

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

{

visited[i] = 0;

}

t = 0;

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

if(visited[i] == 0)

topologicalSortUtil(i);

}

cout<<"Topological Sort: "<<endl;

while(Stack.empty() == false){

cout<<Stack.top()<<" ";

Stack.pop();

}

}

int main(){

int edge;

int x,y;

cin>> node >> edge;

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

cin>>x>>y;

adj[x].push\_back(y);

//adj[y].push\_back(x);

}

topologicalSort();

}

1. **Strongly Connected Component:**

#include<bits/stdc++.h>

using namespace std;

stack<int> Stack;

vector< int > adj[1000];

vector< int > transposeAdj[1000];

int t, node, visited[1000] = {0};

void computeFinishTime(int i){

visited[i] = 1;

for(int j=0; j<adj[i].size(); j++){

int v = adj[i][j];

if(visited[v] == 0){

computeFinishTime(v);

}

}

Stack.push(i);

}

void dfs\_visit(int i, vector< int > adj[]){

visited[i] = 1;

cout<<i<<" ";

for(int j=0; j<adj[i].size(); j++){

int v = adj[i][j];

if(visited[v] == 0){

dfs\_visit(v,adj);

}

}

}

void getTranspose(){

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

vector< int > :: iterator it;

for(it = adj[i].begin(); it != adj[i].end(); it++)

{

transposeAdj[\*it].push\_back(i);

}

}

}

void printComponents(){

int count = 1;

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

{

visited[i] = 0;

}

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

if(visited[i] == 0)

computeFinishTime(i);

}

getTranspose();

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

{

visited[i] = 0;

}

while(!Stack.empty()){

int v = Stack.top();

Stack.pop();

if(visited[v] == 0){

printf("%d : ",count++);

dfs\_visit(v, transposeAdj);

cout<<endl;

}

}

}

int main(){

int edge;

int x,y;

cin>> node >> edge;

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

cin>>x>>y;

adj[x].push\_back(y);

//adj[y].push\_back(x);

}

printComponents();

}