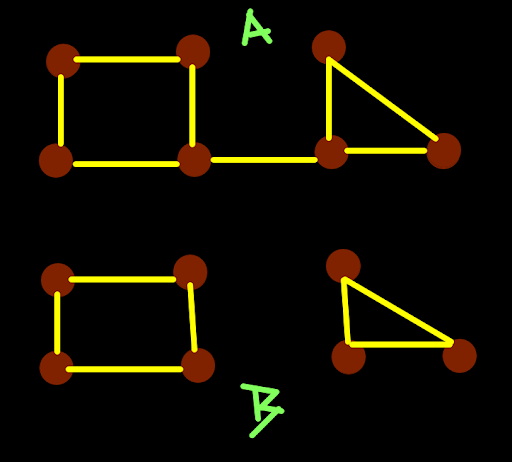
**1. PROBLEM DISCUSSION**

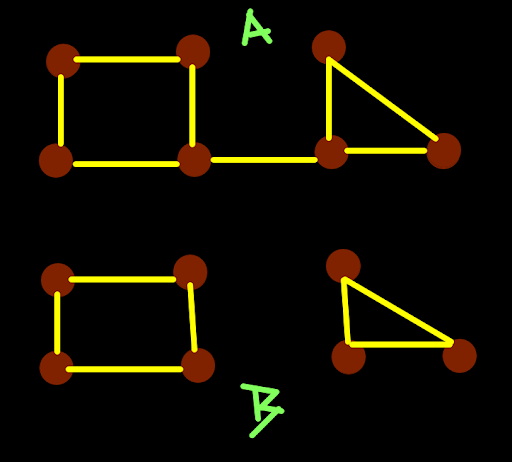
In this problem we will be given a graph and we have to find whether the graph is connected or not. ie.we have to find if there is a path from every vertex to every other vertex

****

Like in the above example Graph A is connected and Graph B is not connected.

**2. APPROACH**

So we can say a graph is connected if we start traversing the graph from any random node and if we are able to fully traverse the graph .

****

Like in Graph A if we start to traverse the graph from any random node we can fully traverse the graph. Similarly if we start to traverse the graph B we will not be able to traverse the graph fully. In one traversal we will be able to traverse any one of the two components of the graph. So we will traverse the graph to find the number of components of the graph. If the graph has one component then it is connected else not connected.

ConsoleCpp

#include<bits/stdc++.h>

using namespace std;

class Edge {

public:

int src;

int nbr;

int wt;

Edge(int src, int nbr, int wt) {

this->src = src;

this->nbr = nbr;

this->wt = wt;

}

};

void dfs(vector<Edge>graph[], int src, vector<bool>& visited) {

visited[src] = true;

for (Edge ed : graph[src]) {

if (visited[ed.nbr] == true)continue;

dfs(graph, ed.nbr, visited);

}

}

int main() {

int vtces;

cin >> vtces;

vector<Edge>graph[vtces];

int edges;

cin >> edges;

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

int v1 ;

int v2 ;

int wt ;

cin >> v1 >> v2 >> wt;

graph[v1].push\_back( Edge(v1, v2, wt));

graph[v2].push\_back( Edge(v2, v1, wt));

}

int src;

cin >> src;

int dest;

cin >> dest;

// write your code here

vector<bool> vis(vtces, false);

dfs(graph, 0, vis);

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

if (!vis[i]) {

cout << "false";

return 0;

}

}

cout << "true";

}

**3. ANALYSIS**

Time complexity O( (v+e) ). Space complexity s=O(v)