// two method to solve it

#include<bits/stdc++.h>

using namespace std;

vector<vector<pair<int,int> > > graph;

void addEdge(int x, int y, int w){

graph[x].push\_back(make\_pair(y,w));

graph[y].push\_back(make\_pair(x,w));

}

void display(){

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

cout<<i<<"->";

for(auto it: graph[i]){

cout<<it.first<<" "<<it.second<<", ";

}

cout<<endl;

}

}

bool cycle(int src, vector<bool> &visited){

queue<int> q;

q.push(src);

while(!q.empty()){

int first = q.front();

q.pop();

if(visited[first]==true){

return true;

}

visited[first] = true;

for(auto it: graph[first]){

if(visited[it.first]==false){

q.push(it.first);

}

}

}

return false;

}

bool cycle\_check(int src, vector<bool> visited){

queue<pi> q;

q.push(make\_pair(src,-1));

while(!q.empty()){

int node = q.front().first;

int par = q.front().second;

q.pop();

if(visited[node]==false){

visited[node] = true;

for(auto it: graph[node]){

if(visited[it.first]==false){

q.push(make\_pair(it.first,node));

}

else{

if(par!=it.first){

return true;

}

}

}

}

}

return false;

}

bool cycle\_DFS(int src, vector<bool> &visited, int par){

visited[src] = true;

for(auto it: graph[src]){

if(visited[it.first]==false){

return cycle\_DFS(it.first,visited,src);

}

else{

if(par!=it.first){

return true;

}

}

}

return false;

}

int main(){

int v,e;

cin>>v>>e;

graph.resize(v);

while(e--){

int x,y,w;

cin>>x>>y>>w;

addEdge(x,y,w);

}

bool a = false;

vector<bool> visited(v);

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

if(visited[i]==false){

bool flag = cycle(i,visited);

if(flag){

a = true;

}

}

}

if(a){

cout<<"true";

}

else{

cout<<"false";

}

}