#include<bits/stdc++.h>

using namespace std;

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

int min\_ = INT\_MAX;

int max\_ = INT\_MIN;

int cmax\_ = INT\_MAX;

int fmin\_ = INT\_MIN;

string smin,smax,cstr,fstr,kstr;

typedef pair<int, string> pi;

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

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;

}

}

void allPath(int src, int dest, vector<bool> visited, string ans){

if(src==dest){

cout<<ans<<endl;

return;

}

visited[src] = true;

for(auto it: graph[src]){

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

string str = to\_string(it.first);

allPath(it.first,dest,visited,ans+str);

}

}

visited[src] = false;

}

void connected\_graph(int src, vector<bool> &visited, vector<int> &vi){

visited[src] = true;

vi.push\_back(src);

for(auto it: graph[src]){

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

connected\_graph(it.first,visited,vi);

}

}

}

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);

}

vector<bool> visited(v);

vector<vector<int> > res;

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

if(visited[i]==false){

vector<int> vi;

connected\_graph(i,visited,vi);

res.push\_back(vi);

}

}

//[[0, 1], [2, 3], [4, 5, 6]]

cout<<"[";

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

cout<<"[";

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

if(j==res[i].size()-1){

cout<<res[i][j];

}

else{

cout<<res[i][j]<<", ";

}

}

if(i==res.size()-1){

cout<<"]";

}

else{

cout<<"], ";

}

}

cout<<"]";

}