//topology sort

//and find minimum edge to be way each node to other node

#include <iostream>

#include <vector>

#include <cstring>

using namespace std;

#define MAX1 20005

vector <int > arr[MAX1];

vector <int > arrin[MAX1];

int res[MAX1];

int visited[MAX1];

int counter=0;

int set1[MAX1];

int in\_out[MAX1][2];

int comp=0;

int N;

void dfs1(int s){

visited[s]=1;

for(int i=0;i<arr[s].size();i++){

if(!visited[arr[s][i]]){

dfs1(arr[s][i]);

}

}

res[counter--]=s;

}

void dfs2(int s,int comp){

set1[s]=comp;

visited[s]=1;

for(int i=0;i<arrin[s].size();i++){

if(!visited[arrin[s][i]]){

dfs2(arrin[s][i],comp);

}

}

}

void dfs3(int s){

visited[s]=1;

for(int i=0;i<arr[s].size();i++){

if(set1[s] != set1[arr[s][i]]) {

//cout<<"set1:"<<set1[s]<<" setdes:"<<set1[arr[s][i]]<<endl;

in\_out[set1[s]][1]++;

in\_out[set1[arr[s][i]]][0]++;

}

if(!visited[arr[s][i]]){

dfs3(arr[s][i]);

}

}

}

int main() {

int T;cin>>T;

for(int t=0;t<T;t++){

cin>>N;

int n;cin>>n;

counter=N-1;

for(int i=0;i<MAX1;i++){

arr[i].clear();

arrin[i].clear();

}

memset(res,-1, sizeof(res));

memset(visited,0, sizeof(visited));

memset(set1,-1, sizeof(set1));

for(int i=0;i<MAX1;i++){

in\_out[i][0]=in\_out[i][1]=0;

}

comp=0;

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

int a;cin>>a;

int b;cin>>b;

arr[a].push\_back(b);

arrin[b].push\_back(a);

}

for(int i=1;i<=N;i++){

if(!visited[i]){

dfs1(i);

}

}

memset(visited,0, sizeof(visited));

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

if(!visited[res[i]]){

dfs2(res[i],comp++);

}

//cout<<"node :"<<res[i]<<" nemofcomp :"<<set1[res[i]]<<endl;

}

memset(visited,0,sizeof(visited));

for(int i=1;i<=N;i++){

if(!visited[i]){

dfs3(i);

}

}

int in1=0;

int out1=0;

for(int i=0;i<comp;i++)if(comp-1){

if(in\_out[i][0]==0){

//cout<<"in\_comp :"<<i<<endl;

in1++;

}

if(in\_out[i][1]==0){

//cout<<"out\_comp: "<<i<<endl;

out1++;

}

}

cout<<"Case "<<t+1<<": "<<max(in1,out1)<<endl;

}

return 0;

}