#include <bits/stdc++.h>

using namespace std;

#define int long long int

#define MOD 1000000007

int n;

const int N = (int)5e5 + 5;

int factorialNumInverse[N + 1];

int naturalNumInverse[N + 1];

int fact[N + 1];

void InverseofNumber(int p) {

naturalNumInverse[0] = naturalNumInverse[1] = 1;

for (int i = 2; i <= N; i++)

naturalNumInverse[i] = naturalNumInverse[p % i] \* (p - p / i) % p;

}

void InverseofFactorial(int p) {

factorialNumInverse[0] = factorialNumInverse[1] = 1;

for (int i = 2; i <= N; i++)

factorialNumInverse[i] = (naturalNumInverse[i] \* factorialNumInverse[i - 1]) % p;

}

void factorial(int p) {

fact[0] = 1;

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

fact[i] = (fact[i - 1] \* i) % p;

}

}

int Binomial(int N, int R, int p) {

if(R>N) return 0;

if(R==N) return 1;

int ans = ((fact[N] \* factorialNumInverse[R]) % p \* factorialNumInverse[N - R]) % p;

return ans;

}

vector<vector<int>> adj;

vector<int> vis;

vector<int> temp\_ways;

vector<int> sub\_treeee;

int dfs(int node) {

vis[node] = 1;

int ans=1, subtree=0;

for(auto x: adj[node]) {

if(vis[x] == 0) {

int curr\_s = dfs(x);

subtree += curr\_s;

ans\*= Binomial(subtree, curr\_s, MOD);

ans%=MOD;

ans\*=temp\_ways[x];

ans%=MOD;

}

}

ans%=MOD;

temp\_ways[node] = ans;

return subtree+1;

}

vector<pair<int, pair<int, int>>> AA;

int getNode(int node) {

vis[node] = 1;

int subtree=0, curr;

for(auto x: adj[node]) {

if(vis[x] == 0) {

curr = getNode(x);

subtree += curr;

int abcd = min(curr,(n-curr));

AA.push\_back({abcd, {x, node}});

}

}

sub\_treeee[node] = subtree+1;

return subtree+1;

}

int32\_t main() {

factorial(MOD);

InverseofNumber(MOD);

InverseofFactorial(MOD);

int t;

cin>>t;

while(t--) {

int k;

cin>>n>>k;

int m=n-1;

adj.clear(), vis.clear(), temp\_ways.clear();

adj.resize(n), vis.resize(n, 0), temp\_ways.resize(n);

while(m--) {

int x,y;

cin>>x>>y;

adj[x-1].push\_back(y-1);

adj[y-1].push\_back(x-1);

}

if(n==1) {

int k1=0,k2=0;

if(k==1) {

vis.clear(), temp\_ways.clear();

vis.resize(n, 0), temp\_ways.resize(n);

dfs(k1);

cout<<k1+1<<" "<<temp\_ways[k1]%MOD<<endl;

} else {

vis.clear(), temp\_ways.clear();

vis.resize(n, 0), temp\_ways.resize(n);

dfs(k2);

cout<<k2+1<<" "<<temp\_ways[k2]%MOD<<endl;

}

continue;

}

AA.clear(), sub\_treeee.clear(), sub\_treeee.resize(n);

getNode(0);

int idx=1;

sort(AA.begin(), AA.end(), greater<pair<int, pair<int, int>>>());

int k1,k2;

if(AA[0].first != AA[1].first) {

int X = AA[0].second.first, Y = AA[0].second.second;

if(sub\_treeee[X] == n-sub\_treeee[X]) {

k1=max(X, Y);

k2=min(X, Y);

} else {

if(sub\_treeee[X] > n-sub\_treeee[X]) {

k1=X;

k2=Y;

} else {

k2=X;

k1=Y;

}

}

} else {

vector<int> temp;

if((AA[0].second.first == AA[1].second.first) || (AA[0].second.first == AA[1].second.second)) {

k1 = AA[0].second.first;

} else {

k1 = AA[0].second.second;

}

idx = 1;

temp.push\_back((AA[0].second.first != k1)? AA[0].second.first : AA[0].second.second);

while(idx!=n && AA[idx-1].first == AA[idx].first ) {

temp.push\_back((AA[idx].second.first != k1)? AA[idx].second.first : AA[idx].second.second);

++idx;

}

k2 = \*max\_element(temp.begin(), temp.end());

}

if(k==1) {

vis.clear(), temp\_ways.clear();

vis.resize(n, 0), temp\_ways.resize(n);

dfs(k1);

cout<<k1+1<<" "<<((temp\_ways[k1]%MOD) +MOD)%MOD<<endl;

} else {

vis.clear(), temp\_ways.clear();

vis.resize(n, 0), temp\_ways.resize(n);

dfs(k2);

cout<<k2+1<<" "<<((temp\_ways[k2]%MOD)+MOD)%MOD<<endl;

}

}

}

// TC-1

// 1

// 9 1

// 1 2

// 1 3

// 1 4

// 1 5

// 5 6

// 6 7

// 6 8

// 6 9