```cpp
#include <bits/stdc++.h>
#define NAME "data."
using namespace std;
typedef vector<vector<int>> vvi;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");

const int64_t MOD = (int64_t)1e9 + 7;

void dfs(const  vvi &graph,
          vector<bool> &used,
          set< pair<int, int>> &bridges,
          vector<int> &tin,
          vector<int> &fup,
         int &timer, int v, int p)
{
  used[v] = true;
  tin[v] = fup[v] = timer++;
  for (int u : graph[v])
  {
    if (u == p)continue;
    if (!used[u])
    {
      dfs(graph, used, bridges, tin, fup, timer, u, v);
      fup[v] =  min(fup[v], fup[u]);
      if (fup[u] > tin[v]) bridges.insert({ min(v, u),  max(v, u)});

    } else fup[v] =  min(fup[v], tin[u]);
  }
}

void findComponent(const  vvi &graph,
                    vector<bool> &used,
                   const  set< pair<int, int>> &bridges,
                    vector<int> &getComponent,
                   int component, int &vs, int v)
{
  used[v] = true;
  getComponent[v] = component;
  vs++;
  for (int u : graph[v])
  {
   pair<int, int> p = { min(v, u),  max(v, u)};
   if (bridges.find(p) != bridges.end()) continue;
   if(!used[u])findComponent(graph,used,bridges,getComponent,component,vs,u);
  }
}

int main()
{
  int n, m;
  fi>>n>>m;
   vvi graph(n);
  for (int i = 0; i < m; i++)
  {
    int v, u;
    fi>>v>>u;
    --v, --u;
    graph[v].push_back(u);
    graph[u].push_back(v);
  }
   set< pair<int, int>> bridges;
   vector<bool> used(n, 0);
   vector<int> tin(n, 0);
   vector<int> fup(n, 0);
  int timer = 0;
  for (int i = 0; i < n; i++)
    if (!used[i]) dfs(graph, used, bridges, tin, fup, timer, i, -1);
  fill(used.begin(), used.end(), false);
  vector<int> componentSize(n, 0);
  vector<int> getComponent(n, 0);
  int component = 0;
  for (int i = 0; i < n; i++)
  {
    if (!used[i])
    {
      int vs = 0;
      findComponent(graph, used, bridges, getComponent, component, vs, i);
      componentSize[component++] = vs;
    }
  }
   vector<int> deg(component, 0);
  for (const auto &bridge : bridges)
  {
    deg[getComponent[bridge.first]]++;
    deg[getComponent[bridge.second]]++;
  }
  int ans = 0;
  int cnt = 1;
  for (int i = 0; i < component; i++)
    if (deg[i] <= 1)
    {
      ans++;
      cnt = (cnt * 1LL * componentSize[i]) % MOD;
    }

  fo<<ans<<' '<<cnt;
  fo<<"\nTime: "<<clock()/(double)1000<<" sec";
  return 0;
}
```