

```
1: #include<iostream>
2: #include<vector>
3: #include<algorithm>
4: #define ll long long int
5: #define rep(n) for(int i=0;i<n;i++)
6:
7: //(1)
8: class UnionFindTree
9: {
10: public:
11:     std::vector<ll> par;
12:     std::vector<ll> rank;
13:     std::vector<ll> size;
14:     //(2)
15:     UnionFindTree(ll n)
16:     {
17:         par.resize(n);
18:         rep(n)
19:         {
20:             par[i]=i;
21:         }
22:         rank.resize(n);
23:         rep(n)
24:         {
25:             rank[i]=0;
26:         }
27:         size.resize(n);
28:         rep(n)
29:         {
30:             size[i]=1;
31:         }
32:     }
33:     //(3)
34:     ll Find(ll x)
35:     {
36:         if(par[x]==x)
37:         {
38:             return x;
39:         }
40:         par[x]=Find(par[x]);
41:         return par[x];
42:     }
43:     //(4)
44:     void Union(ll x,ll y)
45:     {
46:         x=Find(x);
47:         y=Find(y);
48:         if(x==y)
49:         {
50:             return;
51:         }
52:         //(5)
53:         if(rank[x]<rank[y])
54:         {
55:             par[x]=y;
56:             size[y]+=size[x];
57:         }
58:         else
59:         {
60:             par[y]=x;
61:             size[x]+=size[y];
62:             if(rank[x]==rank[y])
63:             {
64:                 rank[x]++;
65:             }
66:         }
67:     }
68:     //(6)
69:     bool Same(ll x,ll y)
70:     {
```

```
71:         return Find(x)==Find(y);
72:     }
73:
74:     //(7)
75:     ll Size(ll x)
76:     {
77:         return size[Find(x)];
78:     }
79: };
80:
81: int main()
82: {
83:     ll n,m;
84:     std::cin>>n>>m;
85:     UnionFindTree *uft=new UnionFindTree(n);
86:     rep(m)
87:     {
88:         ll x,y;
89:         std::cin>>x>>y;
90:         x--;
91:         y--;
92:         uft->Union(x,y);
93:     }
94:     //(8)
95:     std::vector<ll> group(n);
96:     rep(n)
97:     {
98:         group[i]=uft->Find(i);
99:     }
100:    //(9)
101:    std::sort(group.begin(),group.end());
102:    group.erase(std::unique(group.begin(),group.end()),group.end());
103:    ll group_num=group.size();
104:    //(10)
105:    ll group_max=0;
106:    rep(group_num)
107:    {
108:        ll val=uft->Size(group[i]);
109:        if(val>group_max)
110:        {
111:            group_max=val;
112:        }
113:    }
114:    std::cout<<group_num<<" "<<group_max<<std::endl;
115: }
```