```
/*** Codeforces 920E - Connected Components:
You are given an undirected graph consisting of n vertices and  edges. Instead of giving you
the edges that exist in the graph, we give you m unordered pairs (x, y) such that there is no
edge between x and y, and if some pair of vertices is not listed in the input, then there is
an edge between these vertices. You have to find the number of connected components in the
graph and the size of each component. A connected component is a set of vertices X such that
for every two vertices from this set there exists at least one path in the graph connecting
these vertices, but adding any other vertex to X violates this rule.

The first line contains two integers n and m(1≤n≤200000, 0≤m≤min()). Then m lines follow,
each containing a pair of integers x and y (1 ≤ x, y ≤ n, x ≠ y) denoting that there is no
edge between x and y. Each pair is listed at most once; (x, y) and (y, x) are considered the
same (so they are never listed in the same test). If some pair of vertices is not listed in
the input, then there exists an edge between those vertices.

Firstly print k — the number of connected components in this graph. Then print k integers —
the sizes of components. You should output these integers in non-descending order.
***/
#include<bits/stdc++.h>
using namespace std;
#define mx 500005
unordered_set<int>node,forbid[mx+5];
vector<int>gr[mx+5];
int n,m,a[mx+5],vis[mx+5],cnt;
void BFS(int s,int id){
    vis[s]=1; cnt++;
    gr[id].push_back(s); node.erase(s);
    queue<int>qq; qq.push(s);
    unordered_set<int> :: iterator it,jt;
    while(!qq.empty()){
        int u = qq.front(); qq.pop();
        for(it=node.begin(); it!=node.end(); ){
            int x = min(u,*it); int y = max(u,*it);

            if(forbid[x].find(y)==forbid[x].end()){
                jt = it; it++;
                cnt++; vis[*jt]=1;
                gr[id].push_back(*jt);
                qq.push(*jt);
                node.erase(*jt);
            }
            else it++;
        }
    }
}
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1; i<=m; i++){
        int u,v;scanf("%d%d",&u,&v); if(u>v)swap(u,v);
        forbid[u].insert(v);
    }
    for(int i=1; i<=n; i++) node.insert(i);
    int id = 0;
    for(int i=1; i<=n; i++){
        if(vis[i]==0){
            cnt = 0;
            BFS(i,++id);
            a[id] = cnt;
        }
    }
    printf("%d\n",id);
    for(int i=1; i<=id; i++){
        printf("%d",a[i]);
        for(int j=0; j<a[i]; j++) printf(" %d",gr[i][j]);
        printf("\n");
    }
    return 0;
}
```

```cpp
#include<bits/stdc++.h>
using namespace std;
#define mx 500005
set<int>nd;
vector<int>ed[mx+5], gr[mx+5];
int n,m,a[mx+5],vis[mx+5],cnt;
void BFS(int s,int id){
    vis[s] = 1;
    cnt++;
    gr[id].push_back(s);
    nd.erase(s);
    queue<int>qq; qq.push(s);
    set<int> :: iterator it,jt;

    while(!qq.empty()){
        int u = qq.front(); qq.pop();

        for(it=nd.begin(); it!=nd.end();){
            int x = min(u,*it);
            int y = max(u,*it);

            if(!binary_search(ed[x].begin(), ed[x].end(),y)){
                jt = it;
                it++;
                cnt++;
                vis[*jt]=1;
                gr[id].push_back(*jt);
                qq.push(*jt);
                nd.erase(*jt);
            }
            else it++;
        }
    }
}
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1; i<=m; i++){
        int u,v; scanf("%d%d",&u,&v);
        if(u>v)swap(u,v);
        ed[u].push_back(v);
    }
    for(int i=1; i<=n; i++) {
        sort(ed[i].begin(),ed[i].end());
    }
    for(int i=1; i<=n; i++) nd.insert(i);
    int id = 0;

    for(int i=1; i<=n; i++){
        if(vis[i]==0){
            cnt = 0;
            BFS(i,++id);
            a[id] = cnt;
        }
    }

    printf("%d\n",id);
    for(int i=1; i<=id; i++){
        printf("%d",a[i]);
        for(int j=0; j<a[i]; j++) printf(" %d",gr[i][j]);
        printf("\n");
    }

    return 0;
}
```