ninjassolutions.s3.amazonaws.com/0000000000000774.cpp

```
#include <iostream>
#include <vector>
#include <unordered_set>
using namespace std;

void dfs(int start, vector<int>* edges, int n, unordered_set<int>& visited,
unordered_set<int> * component) {
 visited.insert(start);
 component->insert(start);
 vector<int>::iterator it = edges[start].begin();
 for (;it != edges[start].end(); it++) {
  int i = *it;
  if (visited.count(i) > 0) {
   continue;
  }
  dfs(i, edges,n, visited, component);
 }
}

unordered_set<unordered_set<int>*>* getComponents(vector<int>* edges, int n) {
 unordered_set<int> visited;
 unordered_set<unordered_set<int>*>* output = new unordered_set<unordered_set<int>*>();
 for (int i = 0; i < n; i++) {
  if (visited.count(i) == 0) {
   unordered_set<int> * component = new unordered_set<int>();
   dfs(i, edges,n, visited, component);
   output->insert(component);
  }
 }
 return output;
}

int main() {
 int n;
 cin >> n;
 vector<int>* edges = new vector<int>[n];
 int m;
 cin >> m;
 for (int i = 0; i < m; i++) {
  int j, k;
  cin >> j >> k;
  edges[j - 1].push_back(k - 1);
  edges[k - 1].push_back(j - 1);
 }
 unordered_set<unordered_set<int>*>* components = getComponents(edges, n);
 unordered_set<unordered_set<int>*>::iterator it = components->begin();
 while (it != components->end()) {
  unordered_set<int>* component = *it;
```

```
  unordered_set<int>::iterator it2 = component->begin();
  while (it2 != component->end()) {
   cout << *it2 + 1 << " ";
   it2++;
  }
  cout << endl;
  delete component;
  it++;
 }
 delete components;
 delete edges;
}
```