ninjassolutions.s3.amazonaws.com/0000000000000778.cpp

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

bool bipartite(vector<int>* edges, int n) {
 if (n == 0) {
  return true;
 }
 unordered_set<int> sets[2];
 vector<int> pending;
 sets[0].insert(0);
 pending.push_back(0);
 while (pending.size() > 0) {
  int current = pending.back();
  pending.pop_back();
  int currentSet = sets[0].count(current) > 0 ? 0 : 1;
  for (int i = 0; i < edges[current].size(); i++) {
   int neighbor = edges[current][i];
   if (sets[0].count(neighbor) == 0 && sets[1].count(neighbor) == 0) {
    sets[1 - currentSet].insert(neighbor);
    pending.push_back(neighbor);
   } else if (sets[currentSet].count(neighbor) > 0) {
    return false;
   }
  }
 }
 return true;
}

int main() {
 while (true) {
  int n;
  cin >> n;
  if (n == 0)
   break;
  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].push_back(k);
   edges[k].push_back(j);
  }
  bool ans = bipartite(edges, n);
  delete [] edges;
  if (ans) {
```

```
   cout << "BICOLORABLE." << endl;
  } else {
   cout << "NOT BICOLORABLE." << endl;
  }
 }
}
```