

Message Route

Syrjälä's network has n computers and m connections.

Your task is to find out if Uolevi can send a message to Maija, and if it is possible, what is the minimum number of computers on such a route.

Input

The first input line has two integers n and m : the number of computers and connections.

The computers are numbered 1, 2, ..., n . Uolevi's computer is 1 and Maija's computer is n .

Then, there are m lines describing the connections. Each line has two integers a and b : there is a connection between those computers.

Every connection is between two different computers, and there is at most one connection between any two computers.

Output

If it is possible to send a message, first print k : the minimum number of computers on a valid route.

After this, print an example of such a route. You can print any valid solution.

Then, print k lines that describe the new roads. You can print any valid solution.

If there are no routes, print "IMPOSSIBLE".

Constraints

$2 \leq n \leq 10^5$

$1 \leq m \leq 2 \cdot 10^5$

$1 \leq a, b \leq n$

Example

Input:

5 5

1 2

1 3

1 4

2 3

5 4

Output:

3

1 4 5

Problem Breakdown

Lets understand the problem first

We have n computers and m cables between them.

Is there a path from computer 1 to computer n ?

If so, output the shortest path.



Recognize Graph Concepts

Each computer = node

Each cable = edge

Have to find the shortest path in an unweighted graph

BFS is the best fit.

Shortest path in graph (unweighted)

This is a classic case for Breadth-First Search (BFS).

BFS explores layer by layer, guaranteeing the first time we reach B , we've used the fewest steps.



Algorithm Plan

1. Build graph using adjacency list.
2. Use BFS from node 1.
3. Track $\text{parent}[node]$ to reconstruct the path.
4. If node n is never visited \Rightarrow IMPOSSIBLE.
5. Else backtrack from n to 1 and print path.

Path Reconstruction Tips

We don't store the full path during BFS — only the parent. When BFS completes, we backtrack: Reverse the path at the end.

Algorithm

```
#include <bits/stdc++.h>
using namespace std;

#define endl "\n"

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    vector<vector<int>> graph;
    vector<bool> visited;
    vector<int> parent;

    int n, m;
    cin >> n >> m;

    graph.resize(n+1);
    visited.assign(n+1, false);
    parent.assign(n+1, -1);

    for(int i=0; i<m; i++) {
        int u, v;
        cin >> u >> v;

        graph[u].push_back(v);
        graph[v].push_back(u);
    }

    queue<int> q;
    q.push(1); Starting BFS from node 1

    visited[1] = true;
    bool found = false;

    while(!q.empty() && !found) {
        int u = q.front();
        q.pop();

        for(int v : graph[u]) {
            if(!visited[v]) {
                visited[v] = true;
                parent[v] = u;
                q.push(v);

                if(v == n) {
                    found = true;
                    break;
                }
            }
        }
    }

    if(!found) {
        cout << "IMPOSSIBLE" << endl;
        return 0;
    }

    vector<int> path;
    int curr = n;
    while(curr != -1) {
        path.push_back(curr);
        curr = parent[curr];
    }

    reverse(path.begin(), path.end());
}
```

$O(N + M)$

we visit every cell at most once.

Space Complexity

$O(N + M)$

for queues and auxiliary arrays