//Graph coloring

#include<iostream>

#include<vector>

using namespace std;

int n, m; *// number of nodes and edges in the graph*

vector<int> adj[100]; *// adjacency list to represent the graph*

int colors[100]; *// colors assigned to nodes*

int num\_colors; *// number of colors available*

bool isSafe(int node, int color) {

*// check if any adjacent node has the same color*

    for(int i=0; i<adj[node].size(); i++) {

        int neighbor = adj[node][i];

        if(colors[neighbor] == color)

            return false;

    }

    return true;

}

bool graphColoring(int node) {

    if(node == n) *// all nodes have been colored*

        return true;

*// try all possible colors for this node*

    for(int color=1; color<=num\_colors; color++) {

        if(isSafe(node, color)) {

            colors[node] = color;

            if(graphColoring(node+1)) *// move to the next node*

                return true;

            colors[node] = 0; *// backtrack*

        }

    }

    return false;

}

int main() {

    cout<<"Enter the number of nodes: ";

    cin>>n;

    cout<<"Enter the number of edges: ";

    cin>>m;

    cout<<"Enter the edges: "<<endl;

    for(int i=0; i<m; i++) {

        int u, v;

        cin>>u>>v;

        adj[u].push\_back(v);

        adj[v].push\_back(u);

    }

    cout<<"Enter the number of colors available: ";

    cin>>num\_colors;

    if(graphColoring(0)) {

        cout<<"Nodes can be colored with "<<num\_colors<<" colors."<<endl;

        cout<<"The colors assigned to nodes are: "<<endl;

        for(int i=0; i<n; i++)

            cout<<"Node "<<i<<": Color "<<colors[i]<<endl;

    }

    else {

        cout<<"Nodes cannot be colored with "<<num\_colors<<" colors."<<endl;

    }

    return 0;

}

Text

Description automatically generated