**Is Graph Bipartite?**

**Medium**

There is an **undirected** graph with n nodes, where each node is numbered between 0 and n - 1. You are given a 2D array graph, where graph[u] is an array of nodes that node u is adjacent to. More formally, for each v in graph[u], there is an undirected edge between node u and node v. The graph has the following properties:

* There are no self-edges (graph[u] does not contain u).
* There are no parallel edges (graph[u] does not contain duplicate values).
* If v is in graph[u], then u is in graph[v] (the graph is undirected).
* The graph may not be connected, meaning there may be two nodes u and v such that there is no path between them.

A graph is **bipartite** if the nodes can be partitioned into two independent sets A and B such that **every** edge in the graph connects a node in set A and a node in set B.

Return true*if and only if it is****bipartite***.

**Example 1:**

A picture containing text, clock

Description automatically generated

**Input:** graph = [[1,2,3],[0,2],[0,1,3],[0,2]]

**Output:** false

**Explanation:** There is no way to partition the nodes into two independent sets such that every edge connects a node in one and a node in the other.

**Example 2:**

A picture containing text, clock

Description automatically generated

**Input:** graph = [[1,3],[0,2],[1,3],[0,2]]

**Output:** true

**Explanation:** We can partition the nodes into two sets: {0, 2} and {1, 3}.

**Constraints:**

* graph.length == n
* 1 <= n <= 100
* 0 <= graph[u].length < n
* 0 <= graph[u][i] <= n - 1
* graph[u] does not contain u.
* All the values of graph[u] are **unique**.
* If graph[u] contains v, then graph[v] contains u.

class Solution {

public:

    void dfs(int i, int col, vector<vector<int>> &graph, vector<bool> &visited, vector<int> &color, bool &f) {

        visited[i]=true;

        color[i]=col;

        for (auto x : graph[i]) {

            if (!visited[x]) {

                dfs(x, col xor 1, graph, visited, color, f);

                if (!f) return;

            }

            else if (color[x]==color[i]) {

                f=false;

                return;

            }

        }

    }

    bool isBipartite(vector<vector<int>>& graph) {

        int V=graph.size();

        vector<int> color(V, -1);

        vector<bool> visited(V, false);

        for (int i=0; i<V; i++) {

            if (!visited[i]) {

                bool f=true;

                dfs(i, 0, graph, visited, color, f);

                if (!f) return false;

            }

        }

        return true;

    }

};