# 785. Is Graph Bipartite?

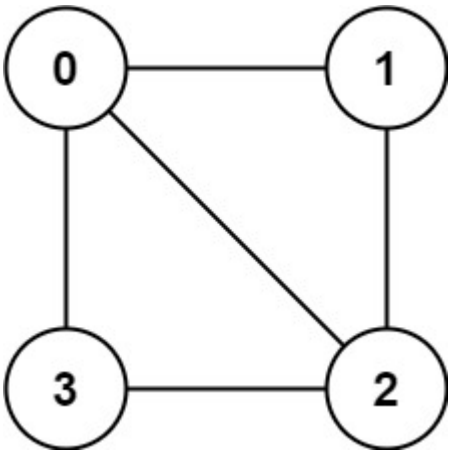
There is an undirected graph with `n` nodes, where each node is numbered between `0` and `n - 1`. You are given a 2D array `graph`, where `graph[u]` is an array of nodes that node `u` is adjacent to. More formally, for each `v` in `graph[u]`, there is an undirected edge between node `u` and node `v`.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

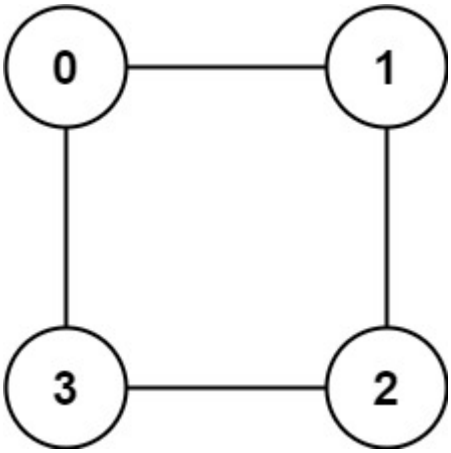
Return `true`* if and only if it is bipartite*.

Example 1:




```
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:




```
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`.

```python
class Solution:
    def isBipartite(self, graph: List[List[int]]) -> bool:
        visited = [-1]*len(graph)
        for i in range(len(graph)):
            if visited[i]==-1:
                if self.checkBipartite(graph,visited,i)==False:
                    return False
        return True


    def checkBipartite(self,graph,visited,src):
        queue = []
        queue.append([src,0])

        while queue:
            node,lev = queue.pop(0)
            if visited[node]!=-1:
```

```
                if lev!=visited[node]:
                    return False
            visited[node] = lev
            for neigh in graph[node]:
                if visited[neigh]==-1:
                    queue.append([neigh,lev+1])

        return True
```