Problem 2076: Process Restricted Friend Requests

Problem Information

Difficulty: Hard
Acceptance Rate: 57.95%
Paid Only: No
Tags: Union Find, Graph

Problem Description

You are given an integer `n` indicating the number of people in a network. Each person is labeled from `0` to `n - 1`.

You are also given a **0-indexed** 2D integer array `restrictions`, where `restrictions[i] = [xi, yi]` means that person `xi` and person `yi` **cannot** become **friends** ,**** either **directly** or **indirectly** through other people.

Initially, no one is friends with each other. You are given a list of friend requests as a **0-indexed** 2D integer array `requests`, where `requests[j] = [uj, vj]` is a friend request between person `uj` and person `vj`.

A friend request is **successful** if `uj` and `vj` can be **friends**. Each friend request is processed in the given order (i.e., `requests[j]` occurs before `requests[j + 1]`), and upon a successful request, `uj` and `vj` **become direct friends** for all future friend requests.

Return _a**boolean array** _`result`,_where each_`result[j]`_is_`true` _if the_`jth` _friend request is**successful** or _`false` _if it is not_.

Note: If `uj` and `vj` are already direct friends, the request is still **successful**.

Example 1:

Input: n = 3, restrictions = [[0,1]], requests = [[0,2],[2,1]] **Output:** [true,false] **Explanation:** Request 0: Person 0 and person 2 can be friends, so they become direct friends. Request 1: Person 2 and person 1 cannot be friends since person 0 and person 1

would be indirect friends (1--2--0).

Example 2:

Input: n = 3, restrictions = [[0,1]], requests = [[1,2],[0,2]] **Output:** [true,false] **Explanation:** Request 0: Person 1 and person 2 can be friends, so they become direct friends. Request 1: Person 0 and person 2 cannot be friends since person 0 and person 1 would be indirect friends (0--2--1).

Example 3:

Input: n = 5, restrictions = [[0,1],[1,2],[2,3]], requests = [[0,4],[1,2],[3,1],[3,4]] **Output:** [true,false,true,false] **Explanation:** Request 0: Person 0 and person 4 can be friends, so they become direct friends. Request 1: Person 1 and person 2 cannot be friends since they are directly restricted. Request 2: Person 3 and person 1 can be friends, so they become direct friends. Request 3: Person 3 and person 4 cannot be friends since person 0 and person 1 would be indirect friends (0--4--3--1).

Constraints:

* `2 <= n <= 1000` * `0 <= restrictions.length <= 1000` * `restrictions[i].length == 2` * `0 <= xi, yi <= n - 1` * `xi != yi` * `1 <= requests.length <= 1000` * `requests[j].length == 2` * `0 <= uj, vj <= n - 1` * `uj != vj`

Code Snippets

C++:

```
class Solution {
public:
vector<bool> friendRequests(int n, vector<vector<int>>& restrictions,
vector<vector<int>>& requests) {

}
};
```

Java:

```
class Solution {
public boolean[] friendRequests(int n, int[][] restrictions, int[][]
```

```
requests) {

}
}
```

Python3:

```
class Solution:
def friendRequests(self, n: int, restrictions: List[List[int]], requests:
List[List[int]]) -> List[bool]:
```