Interviewer B Packet - TIPS #6

# Interviewer:

## Behavioral:

Tell me about a time when you worked under close supervision or extremely loose supervision. How did you handle that?

## Question:

<https://leetcode.com/problems/find-the-town-judge/>

In a town, there are n people labeled from 1 to n. There is a rumor that one of these people is secretly the town judge.

If the town judge exists, then:

1. The town judge trusts nobody.
2. Everybody (except for the town judge) trusts the town judge.
3. There is exactly one person that satisfies properties 1 and 2.

You are given an array trust where trust[i] = [ai, bi] representing that the person labeled ai trusts the person labeled bi.

Return *the label of the town judge if the town judge exists and can be identified, or return* -1 *otherwise*.

Example 1:

* Input: n = 2, trust = [[1,2]]
* Output: 2

Example 2:

* Input: n = 3, trust = [[1,3],[2,3]]
* Output: 3

Example 3:

* Input: n = 3, trust = [[1,3],[2,3],[3,1]]
* Output: -1

Example 4:

* Input: n = 3, trust = [[1,2],[2,3]]
* Output: -1

Example 5:

* Input: n = 4, trust = [[1,3],[1,4],[2,3],[2,4],[4,3]]
* Output: 3

## Follow up Q&A:

Constraints:

* 1 <= n <= 1000
* 0 <= trust.length <= 104
* trust[i].length == 2
* All the pairs of trust are unique.
* ai != bi
* 1 <= ai, bi <= n

## Hint(s):

*Ask if they would like a hint before giving a hint*

## Solution(s): (General concept and time/space complexity)

EXPLANATION:-

Given constraints are

1. The judge believes no one.

2. Everybody believes judge.

so, from these two points, we can say that if any person is trusted by N - 1 persons and the same person believes no one, then we can say that he is a judge.

Now, let's do this in program.

1. Create an array of Size N + 1 to represent each person.

arr[i] represents trust score of i th person

and arr[i] = number of persons trusts him - number of

persons he trusts.

2. Now, traverse through given array.

a, b = a trusts b.

if a person trusts others,

then decrease his score by 1. i.e, arr[a]--

if a person is trusted by others,

then increase his score by 1. i.e, arr[b]++

3. At last traverse through each person,

if anyone found with N - 1 trusts,

then return his index.

4. if not found, return -1

time:- O(N)

space:- O(N)

#### Approach #1: Java

class Solution {

public int findJudge(int N, int[][] trust) {

int[] isTrusted = new int[N+1];

for(int[] person : trust){

isTrusted[person[0]]--;

isTrusted[person[1]]++;

}

for(int i = 1;i < isTrusted.length;i++){

if(isTrusted[i] == N-1) return i;

}

return -1;

}

}

#### Approach #2: C++

class Solution {

public:

int findJudge(int N, vector<vector<int>>& trust) {

vector<int> trusts(N + 1, 0);

for(auto person : trust){

trusts[person[0]]--;

trusts[person[1]]++;

}

for(int i = 1;i <= N;i++){

if(trusts[i] == N - 1)

return i;

}

return -1;

}

};

### Other questions follow up

*Ask if there is more than 5 minutes remaining when they finish their code and testing.*

# Interviewee:

## Question:

<https://leetcode.com/problems/keys-and-rooms/>

There are n rooms labeled from 0 to n - 1 and all the rooms are locked except for room 0. Your goal is to visit all the rooms. However, you cannot enter a locked room without having its key.

When you visit a room, you may find a set of distinct keys in it. Each key has a number on it, denoting which room it unlocks, and you can take all of them with you to unlock the other rooms.

Given an array rooms where rooms[i] is the set of keys that you can obtain if you visited room i, return true *if you can visit all the rooms, or* false *otherwise*.

Examples:

Example 1:

Input: rooms = [[1],[2],[3],[]]

Output: true

Explanation:

We visit room 0 and pick up key 1.

We then visit room 1 and pick up key 2.

We then visit room 2 and pick up key 3.

We then visit room 3.

Since we were able to visit every room, we return true.

Example 2:

Input: rooms = [[1,3],[3,0,1],[2],[0]]

Output: false

Explanation: We can not enter room number 2 since the only key that unlocks it is in that room.

## Follow up Q&A:

Constraints:

* n == rooms.length
* 2 <= n <= 1000
* 0 <= rooms[i].length <= 1000
* 1 <= sum(rooms[i].length) <= 3000
* 0 <= rooms[i][j] < n
* All the values of rooms[i] are unique.

## Code below or on leetcode

class Solution {

public:

int findJudge(int n, vector<vector<int>>& trust) {

}

};