Leetcode May Challenge DAY: 10

If a person got trusted by other people, increase the trust score by 1, since this person receives trust from others.

If a person trusts other people, decrease the trust score by 1, since this person gives its trust to others.

The judge must have trust score of N-1, because judge receives trust from all other N-1 persons and judge never gives its trust.

So, iterate the input list, calculate the trust score for each persion.

Find out if there is a person has trust score of N+1.

There would never be a case that two people got N+1 score.

If we find a person with N-1 scores, all other people must give out their trust to this person, because trust[i] are all different.

If a person gives out trust, the score can never reach N-1. So, there would never be a case that two people got N+1 score.

Time: O(n)

Space: O(n)

1. Python

```
class Solution:
  def findJudge(self, N: int, trust: List[List[int]]) -> int:
    trust_score = [0]*(N+1) # index will be the id of each persion, 0 is a dummy person.
    for a, b in trust:
      trust_score[b] += 1
      trust_score[a] -= 1

    for idx, indegree in enumerate(trust_score):
      if idx != 0 and indegree == N-1:
        return idx
    return -1
```

2. C++

```
class Solution {
public:
    int findJudge(int N, vector<vector<int>>& trust) {
        vector<int> in(N + 1, 0);
        vector<int> out(N + 1, 0);

        for (auto edge : trust) in[edge[1]]++, out[edge[0]]++;

        for (int i = 1; i <= N; i++)
            if (in[i] == N - 1 && out[i] == 0)
                return i;

        return -1;
    }
};
```

3. JAVA

```
class Solution {
    public int findJudge(int N, int[][] trust) {
        int trustsCount[] = new int[N+1];
        int trustedByCount[] = new int[N+1];
        int maxCount = 0;
        int maxCountIndex = 1;
        for(int i = 0; i < trust.length; i++) {
            trustsCount[trust[i][0]]++;
            trustedByCount[trust[i][1]]++;
            if(trustedByCount[trust[i][1]] > maxCount) {
                maxCount = trustedByCount[trust[i][1]];
                maxCountIndex = trust[i][1];
            }
        }

        if(maxCount == N-1 && trustsCount[maxCountIndex] == 0) return maxCountIndex;


        return -1;
    }
}
```