Find the Celebrity

Suppose you are at a party with `n` people (labeled from `0` to `n - 1`) and among them, there may exist one celebrity. The definition of a celebrity is that all the other `n - 1` people know him/her but he/she does not know any of them.

Now you want to find out who the celebrity is or verify that there is not one. The only thing you are allowed to do is to ask questions like: "Hi, A. Do you know B?" to get information of whether A knows B. You need to find out the celebrity (or verify there is not one) by asking as few questions as possible (in the asymptotic sense).

You are given a helper function `bool knows(a, b)` which tells you whether A knows B. Implement a function `int findCelebrity(n)`, your function should minimize the number of calls to `knows`.

Note: There will be exactly one celebrity if he/she is in the party. Return the celebrity's label if there is a celebrity in the party. If there is no celebrity, return `-1`.

Solution 1

The idea is as follows:

first, if person A knows person B, then B could be the candidate of being a celebrity, A must not be a celebrity. We iterate all the n persons and we will have a candidate that everyone knows this candidate.

second, we check two things after we get this candidate. 1. If this candidate knows other person in the group, if the candidate knows anyone in the group, then the candidate is not celebrity, return -1; 2. if everyone knows this candidate, if anyone does not know the candidate, return -1;

// Forward declaration of the knows API.

bool knows(int a, int b);

class Solution {

public:

```
int findCelebrity(int n) {
    if(n<=1) return n;

    int candidate = 0;

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

        if ( !knows(i,candidate) ){
            candidate = i;
        }
    }


    for(int j=0; j<n; j++){

        if(j== candidate) continue;

        if( !knows(j,candidate) || knows(candidate,j) ){
              //if j does not know candidate, or candidate knows j, return -1;
            return -1;
        }

    }


    return candidate;

}
```

};

written by hbsophia original link here

Solution 2

The first pass is to pick out the candidate. If candidate knows i, then switch candidate. The second pass is to check whether the candidate is real.

```
public class Solution extends Relation {
    public int findCelebrity(int n) {
        int candidate = 0;
        for(int i = 1; i < n; i++){
            if(knows(candidate, i))
                candidate = i;
        }
        for(int i = 0; i < n; i++){
            if(i != candidate && (knows(candidate, i) || !knows(i, candidate))) r
eturn -1;
        }
        return candidate;
    }
}
```

written by czonzhu original link here

Solution 3

```
public int findCelebrity(int n) {
    // base case
    if (n <= 0) return -1;
    if (n == 1) return 0;

    Stack<Integer> stack = new Stack<>();

    // put all people to the stack
    for (int i = 0; i < n; i++) stack.push(i);

    int a = 0, b = 0;

    while (stack.size() > 1) {
        a = stack.pop(); b = stack.pop();

        if (knows(a, b))
            // a knows b, so a is not the celebrity, but b may be
            stack.push(b);
        else
            // a doesn't know b, so b is not the celebrity, but a may be
            stack.push(a);
    }

    // double check the potential celebrity
    int c = stack.pop();

    for (int i = 0; i < n; i++)
        // c should not know anyone else
        if (i != c && (knows(c, i) || !knows(i, c)))
            return -1;

    return c;
}
```

written by jeantimex original link here