LeetCode 277

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`.

Method

We need to find the one who doesn’t know others.
The brute force is that we can scan the array(`from 0 to n-1`), check if every two persons know each other. In this solution, we need to call `knows` `2*n^2` times in worst case.

Only when `knows(A,B)=0 && knows(B,A)=1`, `A` is an alternative of the celerity. We can save `A` in a stack. If we found `(A,C)` didn't satisfy this condition, `A` is not the celerity, so we should delete `A` from the stack.

In above solution, we have some duplicate operations, like we call `knows(A,B)` and `knows(B,A)` when we meet `A`, we also call `knows(B,A)` and `knows(A,B)` when we meet `B`. How to decrease the call times?

```
int findCelebrity(int n) {
    stack<int> mys;
    for(int i=0;i<n;++i){
        for(int j=0;j<n;++j){
            if(j!=i){
                if(!knows(i,j) && knows(j,i)){
                    if(mys.empty() || mys.top() != i)
                        mys.push(i);
                }
                else {
                    if(!mys.empty() && mys.top() ==i)
                        mys.pop();
                    break;
                }
            }
        }
    }
    if(mys.empty())
        return -1;
    else
        return mys.top();
}
```

Note

A celerity is the one who others know him, but he doesn’t know others. If A does not know D, and D also does not know A, A will not be a celerity.