278. First Bad Version

Tags	
Property	@September 24, 2022

Question

原文：

You are a product manager and currently leading a team to develop a new product. Unfortunately, the latest version of your product fails the quality check. Since each version is developed based on the previous version, all the versions after a bad version are also bad.

Suppose you have `n` versions `[1, 2, ..., n]` and you want to find out the first bad one, which causes all the following ones to be bad.

You are given an API `bool isBadVersion(version)` which returns whether `version` is bad. Implement a function to find the first bad version. You should minimize the number of calls to the API.

我的理解：

太長直上翻譯(這天腦袋有點頓

簡單來說找到`isBadVersion`回傳中的第一個true，true表示產品壞掉，那它後面的產品也都會是true

翻譯：

你是一名产品经理，目前正带领一个团队开发一个新产品。不幸的是，你的产品的最新版本没有通过质量检查。由于每个版本都是在前一个版本的基础上开发的，所以在一个坏版本之后的所有版本也是坏的。

假设你有n个版本[1, 2, ..., n]，你想找出第一个坏的版本，它导致后面的所有版本都是坏的。

你得到了一个API bool isBadVersion(version)，它返回版本是否是坏的。实现一个函数来查找第一个坏版本。你应该尽量减少对API的调用次数。

自評翻譯正確性：0

- Word Memory：

Code

```
// The API isBadVersion is defined for you.
// bool isBadVersion(int version);

class Solution {
public:
    int firstBadVersion(int n) {
        double left=0,right=n,mid;//設置搜尋範圍
        if(isBadVersion(n)==true&&isBadVersion(n-1)==false){//排除邊緣test case 一開始就在末端抓不到
            return n;
```

```
        }
        if(isBadVersion(1)==true){//排除邊緣test case 一開始就在1抓不到
            return 1;
        }
        while(1){//確認搜尋範圍的中心點是不是true
            mid=(left+right)/2;
            if(isBadVersion(mid)==false){//不是的話 繼續往後找，所以左邊界來到中間的位置，搜右半邊
                left=mid;
            }
            else if(isBadVersion(mid)==true){//是的話，確認前一位是否是false是的話 他就是true的起始 第一個壞掉的位置
                if(isBadVersion(mid-1)==false){
                    return mid;
                }
                right=mid;//該位置不是第一位true的話，右邊界往中間靠攏，因為答案會在左半邊
            }
        }
    }
};
```

> 思路：設置左右邊界，直接看中心點是不是isBadVersion等於true，類似二元搜尋的概念下去找，實際寫了之後，如果錯誤出在頭尾的部分會找不到所以額外寫了兩個if處理頭尾的問題

優良code參考

```
class Solution {
public:
    int firstBadVersion(int n) {
        int lo = 1, hi = n, mid;
        while (lo < hi) {
            mid = lo + (hi - lo) / 2;
            if (isBadVersion(mid)) hi = mid;
            else lo = mid+1;
        }
        return lo;
    }
};
```

> 思路：思路類似，但他算中心點跟移動中心的點的做法有些許不同，可以覆蓋到頭尾