First Bad Version

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 will return whether `version` is bad. Implement a function to find the first bad version. You should minimize the number of calls to the API.

Credits:
Special thanks to @jianchao.li.fighter for adding this problem and creating all test cases.

Solution 1

The binary search code:

```
public int firstBadVersion(int n) {
    int start = 1, end = n;
    while (start < end) {
        int mid = start + (end-start) / 2;
        if (!isBadVersion(mid)) start = mid + 1;
        else end = mid;
    }
    return start;
}
```

written by Pixel_ original link here

Solution 2

```
class Solution {
public:
    int firstBadVersion(int n) {
        int lower = 1, upper = n, mid;
        while(lower < upper) {
            mid = lower + (upper - lower) / 2;
            if(!isBadVersion(mid)) lower = mid + 1;   /* Only one call to API */
            else upper = mid;
        }
        return lower;   /* Because there will alway be a bad version, return lowe
r here */
    }
};
```

written by whaleking1990 original link here

Solution 3

Is there any difference between " (low + high) / 2 " and " low + (high - low) / 2 "?

When I use the first one, it told me "time limit exceed" but if I use the second one, it worked!

written by aaronwei original link here