278. First Bad Version

Add to List | Question | Editorial Solution | My Submissions (/problems/first-bad-version/submissions/)

Total Accepted: **77205** Total Submissions: **318561** Difficulty: **Easy** Contributors: **Admin**

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 (https://leetcode.com/discuss/user/jianchao.li.fighter) for adding this problem and creating all test cases.

Hide Company Tags | Facebook (/company/facebook/)

Hide Tags | Binary Search (/tag/binary-search/)

Hide Similar Problems | (M) Search for a Range (/problems/search-for-a-range/) | (M) Search Insert Position (/problems/search-insert-position/) | (E) Guess Number Higher or Lower (/problems/guess-number-higher-or-lower/)

Have you met this question in a real interview? Yes No

Discuss (https://discuss.leetcode.com/category/348) | Top Solutions | Pick One (/problems/random-one-question/)

C++

```cpp
// Forward declaration of isBadVersion API.
bool isBadVersion(int version);

class Solution {
public:
    // related to https://leetcode.com/problems/search-for-a-range/
    int firstBadVersion(int n) {
        int l = 1, r = n;
        while(l < r) {
            int mid = (r - l) / 2 + l;
            if(isBadVersion(mid)) {
                r = mid;
            } else {
                l = mid + 1;
            }
        }
        return l;
    }



    // int binaryFind(int i, int j) {
    //     if(i == j) return i;
    //     int mid = (j-i)/2 + i;
    //     if (isBadVersion(mid))
    //         return binaryFind(i, mid);
    //     else
    //         return binaryFind(mid+1, j);
    // }

    // int firstBadVersion(int n) {
    //   return binaryFind(1, n);
    // }
};
```

Shortcut: Command + enter

Submit Solution

Send Feedback (mailto:admin@leetcode.com?subject=Feedback)

Submission Result: Accepted (/submissions/detail/87440456/) More Details > (/submissions/detail/87440456/)

Next challenges: (H) Split Array Largest Sum (/problems/split-array-largest-sum)

Share your acceptance!

55 29

Notes

Send Feedback (mailto:admin@leetcode.com?subject=Feedback)