Articles > 278. First Bad Version

Previous (/articles/two-sum/) Next (/articles/range-sum-query-immutable/)

278. First Bad Version (/problems/first-bad-version/)

March 5, 2016 | 129.6K views

Average Rating: 4.72 (167 votes)

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.

Example:

```
Given n = 5, and version = 4 is the first bad version.

call isBadVersion(3) -> false
call isBadVersion(5) -> true
call isBadVersion(4) -> true

Then 4 is the first bad version.
```

Summary

This is a very simple problem. There is a subtle trap that you may fall into if you are not careful. Other than that, it is a direct application of a very famous algorithm.

Solution

Approach #1 (Linear Scan) [Time Limit Exceeded]

The straight forward way is to brute force it by doing a linear scan.

Java

Copy

```java
public int firstBadVersion(int n) {
    for (int i = 1; i < n; i++) {
        if (isBadVersion(i)) {
            return i;
        }
    }
    return n;
}
```

Complexity analysis

- Time complexity : $O(n)$. Assume that $isBadVersion(version)$ takes constant time to check if a *version* is bad. It takes at most $n - 1$ checks, therefore the overall time complexity is $O(n)$.
- Space complexity : $O(1)$.

Approach #2 (Binary Search) [Accepted]

It is not difficult to see that this could be solved using a classic algorithm - Binary search. Let us see how the search space could be halved each time below.

```
Scenario #1: isBadVersion(mid) => false

 1 2 3 4 5 6 7 8 9
 G G G G G G B B B       G = Good, B = Bad
 |       |       |
left    mid    right
```

Let us look at the first scenario above where $isBadVersion(mid) \Rightarrow false$. We know that all versions preceding and including mid are all good. So we set $left = mid + 1$ to indicate that the new search space is the interval $[mid + 1, right]$ (inclusive).

```
Scenario #2: isBadVersion(mid) => true

 1 2 3 4 5 6 7 8 9
 G G G B B B B B B       G = Good, B = Bad
 |       |       |
left    mid    right
```

The only scenario left is where $isBadVersion(mid) \Rightarrow true$. This tells us that mid may or may not be the first bad version, but we can tell for sure that all versions after mid can be discarded. Therefore we set $right = mid$ as the new search space of interval $[left, mid]$ (inclusive).

In our case, we indicate $left$ and $right$ as the boundary of our search space (both inclusive). This is why we initialize $left = 1$ and $right = n$. How about the terminating condition? We could guess that $left$ and $right$ eventually both meet and it must be the first bad version, but how could you tell for sure?

The formal way is to prove by induction (http://www.cs.cornell.edu/courses/cs211/2006sp/Lectures/L06-Induction/binary_search.html), which you can read up yourself if you are interested. Here is a helpful tip to quickly prove the correctness of your binary search algorithm during an interview. We just need to test an input of size 2. Check if it reduces the search space to a single element (which must be the answer) for both of the scenarios above. If not, your algorithm will never terminate.

If you are setting $mid = \frac{left+right}{2}$, you have to be very careful. Unless you are using a language that does not overflow such as Python (https://www.reddit.com/r/Python/comments/36xu5z/can_integer_operations_overflow_in_python/), $left + right$ could overflow. One way to fix this is to use $left + \frac{right-left}{2}$ instead.

If you fall into this subtle overflow bug, you are not alone. Even Jon Bentley's own implementation of binary search had this overflow bug (https://en.wikipedia.org/wiki/Binary_search_algorithm#Implementation_issues) and remained undetected for over twenty years.

Java Copy

```java
public int firstBadVersion(int n) {
    int left = 1;
    int right = n;
    while (left < right) {
        int mid = left + (right - left) / 2;
        if (isBadVersion(mid)) {
            right = mid;
        } else {
            left = mid + 1;
        }
    }
    return left;
}
```

Complexity analysis

- Time complexity : $O(\log n)$. The search space is halved each time, so the time complexity is $O(\log n)$.
- Space complexity : $O(1)$.

Rate this article:

Previous (/articles/two-sum/)

Next (/articles/range-sum-query-immutable/)

Comments: 26 Sort By

Type comment here... (Markdown is supported)

Preview Post

kevincongcc (kevincongcc) ★ 254 January 30, 2018 7:09 AM

well,got TLE five times becaues I use (left + right) / 2,now I know why should use left + (right - left) / 2.

46 Share Reply

SHOW 6 REPLIES

ankitshah009 (ankitshah009) ★ 12 March 12, 2019 4:59 AM

Efficient python library

```
import bisect
```

Read More

10 ⌃ ⌄ Share Reply

SHOW 2 REPLIES

sotondolphin (sotondolphin) ★ 2 November 30, 2017 3:41 PM

why not just start from the lasted version - 1? since the bad version is likely to be near the latest version

3 ⌃ ⌄ Share Reply

SHOW 3 REPLIES

suhaboncukcu (suhaboncukcu) ★ 1 January 29, 2018 2:26 AM

@Hidestor, I got the same result till I try low + ((high-low) >>> 1) (language: js)

1 ⌃ ⌄ Share Reply

codeXcode (codexcode) ★ 20 January 10, 2018 11:41 PM

To my uter surprise low+(high-low)>>1 is giving TLE and low+(high-low)/2 is getting AC. How is this possible? We know bit manipulation is faster than arithmetic calculations or I am wrong in my assumption?

1 ⌃ ⌄ Share Reply

SHOW 2 REPLIES

idomiralin (idomiralin) ★ 1 October 6, 2017 12:25 AM

Why when I use mid = (left + right) / 2 time limit exceeded is returned and when we use mid = left + (right - left) / 2 it works?

1 ⌃ ⌄ Share Reply

SHOW 2 REPLIES

just_morris (just_morris) ★ 0 May 22, 2019 3:27 AM

Python3 error:
TypeError: firstBadVersion() takes 2 positional arguments but 3 were given in main:
... ret = Solution().firstBadVersion(n, bad) ...
after i fix it (def firstBadVersion(self, n, bad): OR ret = Solution().firstBadVersion(n))
NameError: name 'isBadVersion' is not defined

Read More

0 ⌃ ⌄ Share Reply

asrajavel267 (asrajavel267) ★ 21 May 9, 2019 9:48 AM

What are we supposed to return if there are no bad versions?

0 ⌃ ⌄ Share Reply

SHOW 1 REPLY

heimanchan (heimanchan) ★ 0 ◷ May 3, 2019 5:12 AM ⋮

Why is calculating the mid as left + (right-left) /2 way faster than just (left + right) / 2?

0 ⌃ ⌄ | Share | ↩ Reply

SHOW 3 REPLIES

NideeshT (nideesht) ★ 115 ◷ April 23, 2019 6:34 AM ⋮

(Java) Youtube Video Explanation - accepted

Hi everyone, I'm making Youtube videos to help me study/review solved problems. Wanted to share if it helps!

Note I claim no rights to this question. All rights belong to Leetcode. The company tags in the video title are

Read More

0 ⌃ ⌄ | Share | ↩ Reply

‹ 1 2 3 ›

Help Center (/support/) | Terms (/terms/) | Privacy Policy (/privacy/)

🇺🇸 United States (/region/)