First Bad Version Leetcode Solution

First Bad Version LeetCode Solution is a popular problem that many developers encounter while practicing coding for interviews or improving their algorithmic skills. The problem revolves around a scenario where you are tasked with finding the first bad version of a product from a series of versions. This article will delve into the problem statement, explain the binary search approach that leads to an efficient solution, and also provide a detailed analysis of the time and space complexity involved.

Problem Statement

In the First Bad Version problem, you are given a function `bool isBadVersion(version)` which returns `true` if the version is bad and `false` otherwise. Your goal is to find the first bad version in a sequence of `n` versions, where versions are numbered from `1` to `n`.

The problem can be summarized as follows:

1. You are given a total of `n` versions.
2. There exists a first bad version, and all subsequent versions are also bad.
3. You have to determine the first bad version in an optimized manner.

The challenge lies in efficiently narrowing down the range of versions to find this first bad version, rather than checking each version one by one.

Understanding the Problem

To understand how to approach this problem, let’s clarify a few key points:

Characteristics of the Versions

- Sequential Nature: If version `k` is bad, then all versions greater than `k` are also bad.
- Existence of a Bad Version: There is guaranteed to be at least one bad version in the list of versions.

Given this structure, a direct approach (i.e., checking each version sequentially) would be inefficient, especially for larger values of `n`. The optimal way to solve this problem is to utilize the properties of binary search.

Binary Search Approach

Binary search is an efficient algorithm for finding an item from a sorted list of items. It works by repeatedly dividing the search interval in half. In this problem, we can leverage the sorted nature of the versions based on whether they are bad or good.

Steps to Implement Binary Search

1. Initialization: Start with two pointers, `left` initialized to `1` and `right` initialized to `n`.
2. Loop: While `left` is less than `right`:
- Calculate the midpoint: `mid = left + (right - left) // 2`.
- Use the `isBadVersion(mid)` function to check if the version at `mid` is bad.
- If it is bad, then the first bad version must be at `mid` or to the left of `mid`. Thus, set `right = mid`.
- If it is not bad, then the first bad version must be to the right of `mid`. Thus, set `left = mid + 1`.
3. Termination: Once the loop terminates, `left` will point to the first bad version.

Code Implementation

Here’s how the binary search algorithm can be implemented in Python:

```python
def isBadVersion(version):
This function is predefined and returns True if the version is bad, False
```

```
otherwise.
pass

def firstBadVersion(n):
left, right = 1, n
while left < right:
mid = left + (right - left) // 2
if isBadVersion(mid):
right = mid the first bad version is mid or to the left
else:
left = mid + 1 the first bad version is to the right
return left left is now pointing to the first bad version
```

In this code, we define the `firstBadVersion` function which implements the binary search logic as described. The `isBadVersion` function is assumed to be predefined and is used to check the status of each version.

Time and Space Complexity Analysis

Understanding the efficiency of the solution is crucial, especially in competitive programming and real-world applications.

Time Complexity

- The time complexity of this binary search approach is O(log n). Each iteration of the while loop effectively halves the search space, leading to logarithmic time complexity.

Space Complexity

- The space complexity of the algorithm is O(1) because we are only using a constant amount of space for the variables `left`, `right`, and `mid`.

Conclusion

The First Bad Version problem is a classic example of how binary search can be applied to efficiently solve problems that might seem linear at first glance. By understanding the properties of the problem and leveraging the binary search algorithm, we can significantly reduce the time complexity from linear to logarithmic. This not only makes the solution faster but also more scalable for larger inputs.

Practicing problems like this one is essential for honing your algorithmic skills and preparing for technical interviews. The key takeaway is to always look for patterns and properties in the problem that allow you to reduce the complexity of your solution. With the right approach, even seemingly difficult problems can be tackled effectively.

Frequently Asked Questions

What is the 'First Bad Version' problem on LeetCode?

The 'First Bad Version' problem is a coding challenge that asks you to find the first bad version of a software release given a function that can determine if a version is bad. You need to implement a solution that efficiently identifies this version using binary search.

What is the optimal time complexity for solving the 'First Bad Version' problem?

The optimal time complexity for solving the 'First Bad Version' problem is O(log n), where n is the total number of versions. This is achieved by using a binary search approach.

What is the primary function signature for the 'First Bad Version' solution?

The primary function signature for the solution is typically 'public int firstBadVersion(int n)', where n represents the total number of versions.

How does the binary search algorithm work in the context of 'First Bad Version'?

In the binary search algorithm for the 'First Bad Version', you maintain two pointers, low and high. You check the middle version. If it's bad, it means the first bad version is either the middle version or to the left; if it's good, the first bad version must be to the right. You continue narrowing down until you find the first bad version.

What edge cases should be considered when implementing the 'First Bad Version' solution?

Edge cases include scenarios where all versions are good, all are bad, or where there is only one version. You should also handle the case where the first version is the bad one.

Can you provide a sample code implementation for the

'First Bad Version' problem?

Sure! Here's a sample implementation in Python:
```python
class Solution:
def isBadVersion(self, version: int) -> bool:
Assume this function is provided
pass

def firstBadVersion(self, n: int) -> int:
low, high = 1, n
while low < high:
mid = (low + high) // 2
if isBadVersion(mid):
high = mid
else:
low = mid + 1
return low
```

Why is it important to use binary search for the 'First Bad Version' problem?

Using binary search is important because it significantly reduces the number of checks needed to find the first bad version. Instead of checking each version sequentially, binary search allows you to halve the search space with each iteration, making the solution efficient for large inputs.

First Bad Version Leetcode Solution

2025年 **7**月显卡天梯图（更新**RTX 5060**）
Jun 30, 2025 · 显卡游戏性能天梯 1080P/2K/4K分辨率，以最新发布的RTX 5060为基准（25款主流游戏测试成绩取平均值）

中国人填资料时，**first name**是名还是姓？_百度知道
first name 是名； last name 是姓。比如说中国人“张三”，其中“张”是last name，“三”是first name。外国人的人名是“姓在后，名在前”以Jim Green为例，其中格林是姓，吉姆是名，那 …

英文**1**号到**31**号日期缩写 **-** 百度知道
Jun 10, 2022 · 英文1号到31号日期缩写1日: first (1st)、2日: second (2nd)、3日: third (3rd)、4日: fourth (4th)、5日: fifth (5th)、6日: sixth (6th)、7 …

1st、2nd、3rd、...10th ...10th ...
first 1st second 2nd third 3rd fourth 4th fifth 5th sixth 6th seventh 7th eighth ninth 9th tenth 10th eleventh 11th twelfth 12th ...

*first name、last name?*_...
...first name、last name?last name、family name、first name、given name...Michael Jordan. Michael、(first name)、Jordan、(last name)、1、...

surname、first name、family name...
... surname、first name、family name... 1、surname, family name...first name ... 2...surname 、family name... ...

first name、last name? - ...
... shiyatoz ...2017-11-24 · TA...2291... ... Leszek = first name Godzik = last name ...first name ...last name...family ...

stata、ivreghdfe... - ...
...stata...(╥...

... **-** ...
... (first name)...,... (last name). ...first name、last name... ...

Address line1、Address line2..._...
... .../Add line 1: ...+...+...+... .../Address line2: ...+...+... Address line1... ...

2025、7、...RTX 5060、
Jun 30, 2025 · ... 1080P/2K/4K...RTX 5060...25...

first name..._...
first name ... last name ... ..."..."..."、"...last name、"、"...first name、 ..."..." 、Jim Green... ... ...

1、31 - ...
Jun 10, 2022 · 1、31...1、first、1st、2、second、2nd、3、third、3rd、4、fourth、4th、5、fifth、5th、6、sixth、6th、7 ...

1st、2nd、3rd、...10th ...10th ...
first 1st second 2nd third 3rd fourth 4th fifth 5th sixth 6th seventh 7th eighth ninth 9th tenth 10th eleventh 11th twelfth 12th ...

*first name、last name?*_...
...first name、last name?last name、family name、first name、given name...Michael Jordan. Michael、(first name)、Jordan、(last name)、1、...

surname、first name、family name...
... surname、first name、family name... 1、surname, family name...first name ... 2...surname 、family name... ...

... first name、last name? - ...

举报 shiyatoz 推荐于2017-11-24 · TA获得超过2291个赞 关注 Leszek = first name Godzik = last name 外国人的first name 一般在前面，就是我们的名，而last name在最后，也称作为family ...

stata使用ivreghdfe报错？ - 知乎

如图为多维固定效应模型应用工具变量时stata回归的代码，可是报错了，有哪位好心人知道是怎么回事吗？(╥...

对一个陌生的英文名字，如何快速确定哪个是姓哪个是名？ - 知乎

这里我以美国人的名字为例，在美国呢，人们习惯于把自己的名字 (first name)放在前,姓放在后面 (last name). 这也就是为什么叫first name或者last name的原因（根据位置摆放来命名的）。比 ...

Address line1和Address line2分别要怎么填写？_百度知道

两者就一个区别：地址 精确度 不同。第一行/Add line 1: 所在区+所在城市+所在省（+所在国家）。第二行/Address line2: 门牌号+楼号+街道号。Address line1填写内容要细致到门牌 ...

Discover the efficient solution to the 'First Bad Version' problem on LeetCode. Learn more about the approach and optimize your coding skills today!

Back to Home