387. First Unique Character in a String

Dec. 9, 2018 | 278.9K views

Average Rating: 4.29 (55 votes)

Given a string, find the first non-repeating character in it and return its index. If it doesn't exist, return -1.

Examples:

```
s = "leetcode"
return 0.

s = "loveleetcode"
return 2.
```

Note: You may assume the string contains only lowercase English letters.

Solution

Approach 1: Linear time solution

The best possible solution here could be of a linear time because to ensure that the character is unique you have to check the whole string anyway.

The idea is to go through the string and save in a hash map the number of times each character appears in the string. That would take $\mathcal{O}(N)$ time, where `N` is a number of characters in the string.

And then we go through the string the second time, this time we use the hash map as a reference to check if a character is unique or not.
If the character is unique, one could just return its index. The complexity of the second iteration is $\mathcal{O}(N)$ as well.

1 / 15

Java Python Copy

```java
class Solution:
    def firstUniqChar(self, s: str) -> int:
        """
        :type s: str
        :rtype: int
        """
        # build hash map : character and how often it appears
        count = collections.Counter(s)

        # find the index
        for idx, ch in enumerate(s):
            if count[ch] == 1:
                return idx
        return -1
```

Complexity Analysis

- Time complexity : $\mathcal{O}(N)$ since we go through the string of length `N` two times.
- Space complexity : $\mathcal{O}(1)$ because English alphabet contains 26 letters.

Rate this article:

Previous Next

Comments: 75 Sort By

Type comment here... (Markdown is supported)

Preview Post

rahulkun ★ 454 December 10, 2018 1:31 AM
space complexity is constant O(26) ~ O(1) for any length of string
224 Share Reply
SHOW 10 REPLIES

pritam_roy ★ 65 December 9, 2018 11:24 PM
Should not the space complexity be O(1)? The algorithm is iterating over a constant(26) number of bins as keys for hashmap. Alternatively one can use an integer array count[26] for keeping the count of char counts in the word.
53 Share Reply
SHOW 10 REPLIES

terriblewhiteboard ★ 3841 May 5, 2020 1:51 PM
I made a video if anyone is having trouble understanding the solution
https://www.youtube.com/watch?v=21LDcomZ1as&feature=youtu.be


Read More
23 Share Reply

keeproaring ★ 59 May 12, 2019 2:48 PM
Why is space Complexity O(N)? at most there will be 26 characters. Mostly less than that only.so Hashmap Big Oh should be O(1)
16 Share Reply
SHOW 2 REPLIES

acmqi ★ 24 April 4, 2019 7:33 AM
I think the 1st approach should be O(n^2) time with O(1) space, which is pick one character and check if there is repeating character in the rest of the string.
2nd approach will go to the hashMap version, which is optimized from 1st solution.
20 Share Reply

xi31 ★ 32 April 25, 2019 10:21 PM
My approach is slightly different. Strict time complexity is O(N + 26) instead of O(N * 2).

Basically, what stored in letters[] is the index of the first occurrence (If duplicated, set to -1). Then we scan the array of size 26 to find the minimum index, and that is the answer. We do not need to scan the whole string again!
Read More
12 Share Reply
SHOW 5 REPLIES

BlackLord ★ 10 June 22, 2019 3:36 PM
Does order of insertion maintain in HashMap?
9 Share Reply
SHOW 5 REPLIES

kevlar_ksb ★ 37 April 8, 2019 12:36 PM
Since we can assume only lowercase letters.
Using a fixed array instead of a HashMap is a lot faster.

```
class Solution {
```

Read More
13 Share Reply
SHOW 2 REPLIES

KendraDonaldson ★ 43 February 15, 2019 6:30 AM
use enumerate in python! :)
8 Share Reply
SHOW 2 REPLIES

yameen ★ 18 January 15, 2019 11:31 AM
Similar concept using an array.

public int firstUniqChar(String s) {
int [] charCount = new int[26];

Read More
8 Share Reply
SHOW 4 REPLIES

‹ 1 2 3 4 5 6 7 8 ›