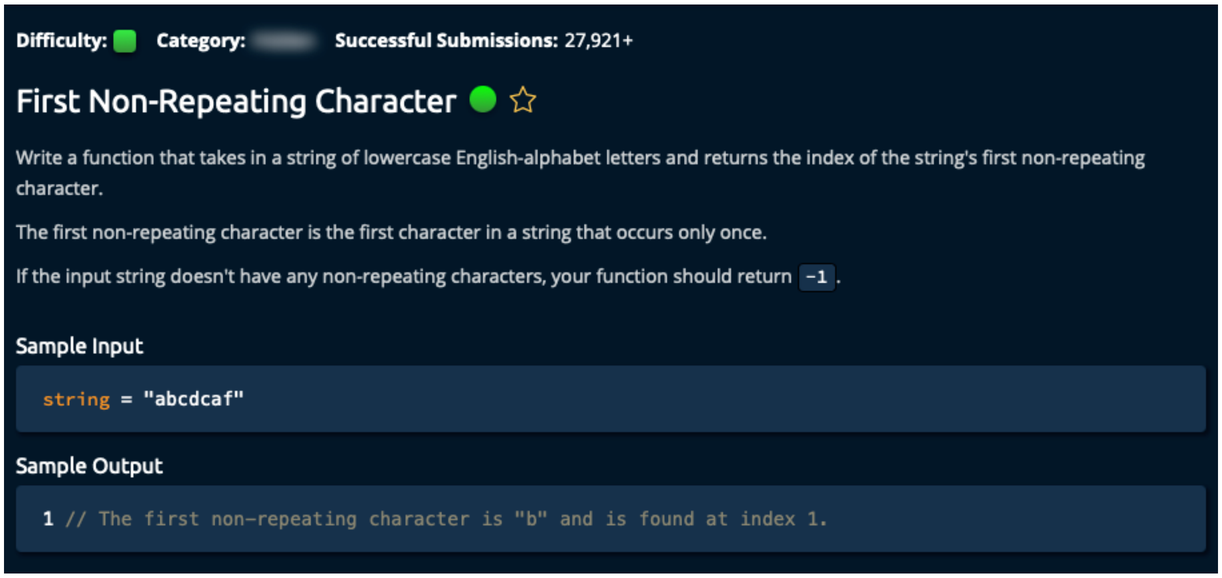
Code Challenge #15 First Non-Repeating Character (Easy)



Difficulty: 🟩 **Category:** [redacted] **Successful Submissions:** 27,921+

First Non-Repeating Character ● ☆

Write a function that takes in a string of lowercase English-alphabet letters and returns the index of the string's first non-repeating character.

The first non-repeating character is the first character in a string that occurs only once.

If the input string doesn't have any non-repeating characters, your function should return `-1`.

Sample Input

```
string = "abcdcaf"
```

Sample Output

```
// The first non-repeating character is "b" and is found at index 1.
```

Solution #1

```javascript
function firstNonRepeatingCharacter(string) {
  for (let idx = 0; idx < string.length; idx++) {
        let foundDuplicate = false;
        for (let idx2 = 0; idx2 < string.length; idx2++){
              if (string[idx] === string[idx2] && idx !== idx2) foundDuplicate = true;
        }

        if (!foundDuplicate) return idx;
  }

   return -1;
}

```

Explanation

The solution to this problem requires us to use two for loops. The first for loop is implemented with an (let idx = 0; idx < string.length; idx++). Inside this for loop we have a let variable called foundDuplicate which has a value of false. Within this for loop we have another for loop for (let idx2 = 0; idx2 < string.length;

idx2++). Within this for loop we check to see if the values at each string index match and the indexes are not the same. If both conditions are true we change the value of the foundDuplicate value to true. Outside the of the inner for loop we have an if condition to check if the foundDuplicate is the opposite of false which is true we return the idx. This means that we found the index which does not have a nonrepeating character. If the string does not have a nonrepeating character, we return -1. This solution has a time complexity of O(n^2).

Solution #2

```
function firstNonRepeatingCharacter(string) {
  const characterFrequencies = {};

  for (const character of string) {
        if (!(character in characterFrequencies)) {characterFrequencies[character] = 0}

        characterFrequencies[character]++;
  }

        for (let idx = 0; idx < string.length; idx++) {
              const character = string[idx];
              if (characterFrequencies[character] === 1) return idx;
        }
   return -1;
}
```

Explanation

This solution uses a hash solution which is labelled const characterFrequencies. It then uses a syntactic sugar loop called for const character of string to iterate through each character of the string. In the for loop it checks to see if the character is not in the characterFrequencies hash. If it isn't we assign it in the characterFrequencies hash using characterFrequencies[character] = 0. We then increment the characterFrequencies[character] by 1. We then have a second for loop outside of the first for loop which is implemented using a let idx. It is implemented using this logic: for (let idx = 0; idx < string.length; idx++) . We then have a variable called const character which finds the value of the string using string[idx]. We then check if characterFrequencies[character] === 1. The first character that matches this is the first nonrepeating character and we return the idx. If no match is found we return -1 which is the last line of code outside of both for loops inside the main function of firstNonRepeatingCharacter. This code runs in O(n) time.