First Unique Character in a String

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

Examples:

```
s = "leetcode"
return 0.

s = "loveleetcode",
return 2.
```

Note: You may assume the string contain only lowercase letters.

Solution 1

Hey guys. My solution is pretty straightforward. It takes O(n) and goes through the string twice:

1. Get the frequency of each character.
2. Get the first character that has a frequency of one.

Actually the code below passes all the cases. However, according to@xietao0221, we could change the size of the frequency array to 256 to store other kinds of characters. Thanks for all the other comments and suggestions. Fight on!

```
public class Solution {
    public int firstUniqChar(String s) {
        int freq [] = new int[26];
        for(int i = 0; i < s.length(); i ++)
            freq [s.charAt(i) - 'a'] ++;
        for(int i = 0; i < s.length(); i ++)
            if(freq [s.charAt(i) - 'a'] == 1)
                return i;
        return -1;
    }
}
```

written by ZachC original link here

Solution 2

```
public static int firstUniqChar(String s) {

  char[] a = s.toCharArray();

  for(int i=0; i<a.length;i++){
   if(s.indexOf(a[i])==s.lastIndexOf(a[i])){return i;}
  }
  return -1;
    }
```

written by Lindsayling original link here

Solution 3

```
class Solution {
public:
    int firstUniqChar(string s) {
        int alphabet[26] = {0};
        for (int i = 0; i < s.size(); ++i){++alphabet[s[i] -'a'];}
        int i = 0;
        while (i < s.size() && alphabet[s[i]-'a'] > 1) ++i;
        return i == s.size() ? -1 : i;
    }
};
```

written by AlgoGuruZ original link here