

BAT算法面试题(三)-无重复字符的最长子串(2)

文章出自 Hello Code 开发者学习平台 CC老师

获取更新文章/视频 关注公众号:

HelloCode开发者学习平台



昨天分享的是暴力解决方法.暴力法非常简单,但是它的速度不够快!那我们该如何去做优化了?

一.算法题

• 题目

Given a string, find the length of the longest substring without repeating characters.

• Example

- Given "abcabcbb", the answer is "abc", which the length is 3.
- Given "bbbbbb", the answer is "b", with the length of 1.
- Given "pwwkew", the answer is "wke", with the length of
- Note that the answer must be a substring, "pwke" is a subsequence and not a substring.

二.算法题解读

- **题目大意:**给定一个字符串,找出不含有重复字符的最长子串的长度
- **解读Example**

- 给定"abcbabb",没有重复字符的最长子串是"abc",那么长度就是3
- 给定"bbbbbb",最长子串就是"b",长度就是1
- 给定pwwkew,最长子串就是"wke",长度为3,
- **注意**,必须是一个子串."pwke",是子序列,而不是子串

三."滑动窗口法"优化解决

使用暴力法解决是非常简单,但是在暴力法中我们会反复检查一个子字符串是否含有重复的字符.但其实没有这个必要.

3.1前导关键词介绍

- **HashSet**

HashSet 是 Java 中实现 Set 接口.由哈希表支持.它不保证Set的迭代顺序,但是它利用Hash的原理来确保元素的唯一性.在 HashSet 中,元素都存到 HashMap 键值对的key上面.而 Value 时有一个统一的 Hash 值.

- **HashSet的插入**

当有新的值加入时,底层的 HashMap 会判断Key值是否存在,如果不存在则插入新值.同时这个插入的细节会按照 HashMap 插入细节.如果存在则不插入.

- **滑动窗口:**是指的是数组/字符串问题的常用抽象概念.窗口通常在数组/字符串中由开始和结束的索引定义的一系列元素的集合.即可 $[i, j)$ (左闭,右开).而滑动窗口是可以将2个边界向某一个方向"滑动"的窗口.例如,我们将 $[i, j)$ 向右滑动1个元素,则它将变成 $[i+1, j+1)$ (左闭,右开);

3.2 思路

如果从索引 `i` 到 `j-1` 之间的子字符串 `S[i:j]` 已经被检查为没有重复字符. 那则只需要检查 `s[j]` 对应的字符是否存在于子字符串 `s[i:j]`;

由于在C语言中是没有集合这一个概念的. 所以我们使用java来实现. 我们可以通过HashSet作为活动窗口. 那我们只需要用O(1)的时间来完成对字符是否在当前子字符串的检查.

我们使用 `HashSet` 将字符存储在当前窗口 `[i, j)`, 最初 `i=j`. 然后我们向右侧滑动索引 `j`, 如果它不在 `HashSet` 中, 则我们会继续滑动 `j`. 直到 `s[j]` 已经存在于 `HashSet` 中, 此时, 我们就已经找到的没有重复字符的最长子串将会以索引 `i` 开头. 如果我们将所有的 `i`, 都做如此操作即可得到结果.

Java Code

```
public class Solution {
    public int lengthOfLongestSubstring(String s) {
        int n = s.length();
        Set<Character> set = new HashSet<>();
        int ans = 0, i = 0, j = 0;
        while (i < n && j < n) {
            //试图调整[i, j]的范围
            if (!set.contains(s.charAt(j))) {
                set.add(s.charAt(j++));
                ans = Math.max(ans, j - i);
            }
            else {
                set.remove(s.charAt(i++));
            }
        }
        return ans;
    }
}
```

3.3 复杂度分析

- 时间复杂度: $O(2n) = O(n)$; 在最糟糕的情况下, 每个字符顶多被 `i, j` 访问

2次.

- **空间复杂度:** $O(\min(m, n))$. 窗口滑动法需要 $O(K)$ 的空间, K 指的是集合大小. 而集合的大小取决于字符串 n 的大小以及字符串集的大小.

小编OS:

如有疑问, 留言即可. 胖C会利用空余时间给大家做一个简单解答的.
持续更新关注公众号!



