3. Longest Substring Without Repeating Characters

Question | My Submissions

Total Accepted: **116682** Total Submissions: **553181** Difficulty: **Medium**

> Given a string, find the length of the longest substring without repeating characters. For example, the longest substring without repeating letters for "abcabcbb" is "abc", which the length is 3. For "bbbbb" the longest substring is "b", with the length of 1.

这题感觉比较难，想了很长时间。

思路： 建立一个动态窗口，又或者说双指针，一头一尾。首先移动尾指针，一直到遇到重复元素为止。这时候临时把头尾指针的距离作为结果。然后把头指针移动到出现重复元素的下一个元素。也就是开始检查新的字符串，但并不需要重新进行扫描，只需要知道头指针的为止，并且使用一个 hashset 来保存扫描过的字符的位置值就可以了。这样做可以把算法优化为 O(n)线性复杂度。

需要保证重复元素是头指针指向的元素或者之后
更新重复元素的位置值

```python
class Solution(object):
    def lengthOfLongestSubstring(self, s):

        if s == None or len(s) == 0:
            return 0

        n = len(s)
        head = 0
        tail = 0
        dic = dict()
        res = 0

        while tail < n:
            if s[tail] in dic and head <= dic[s[tail]]: #需要保证重复元素是头指针指向的元素或者之后
                head = dic[s[tail]] + 1 #头指针移动到出现重复元素的下一个元素

            dic[s[tail]] = tail # 更新重复元素的位置值
            res = max(res, tail - head + 1)
            tail += 1

        return res
```

例如对于字符串 abcabcbb, 在扫描过程中
Head = 0
第一步： abca, 出现重复， head = 0 -> head = 1 指向 b
第二步： b, 出现重复，head = 1 -> head = 2 指向 c
第三步： c, 出现重复， head = 2 -> head = 3 指向 a
第四步： b, 出现重复， head = 3 -> head = 5 指向 c