3. Longest Substring Without Repeating Character

Created	@July 14, 2020 5:22 AM
Difficulty	Medium
LC Url	https://leetcode.com/problems/longest-substring-without-repeating-characters/
Importance	
Tag	NEET Sliding Window Two pointers
Video	https://www.youtube.com/embed/wiGpQwVHdE0

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

Example 1:

```
Input: s = "abcabcbb"
Output: 3
Explanation: The answer is "abc", with the length of 3.
```

Example 2:

```
Input: s = "bbbbb"
Output: 1
Explanation: The answer is "b", with the length of 1.
```

Example 3:

```
Input: s = "pwwkew"
Output: 3
Explanation: The answer is "wke", with the length of 3.
Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.
```

Constraints:

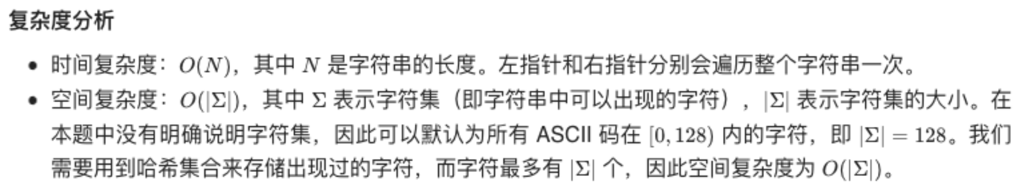
- `0 <= s.length <= 5 * 10 4`
- `s` consists of English letters, digits, symbols and spaces.

Solution

```
class Solution:
    def lengthOfLongestSubstring(self, s: str) -> int:
        # https://leetcode.cn/problems/longest-substring-without-repeating-characters/solution/wu-zhong-fu-zi-fu-de-zui-chang-zi-chuan-by-l

        cur_chars = set()
        n = len(s)

        right, length = -1, 0
        for left in range(n):
            if left != 0:
                cur_chars.remove(s[left - 1])

            while right + 1 < n and s[right + 1] not in cur_chars:
                cur_chars.add(s[right + 1])
                right += 1

            length = max(length, right - left + 1)
        return length
```

力扣

https://leetcode.cn/problems/longest-substring-without-repeating-characters/solution/wu-zhong-fu-zi-fu-de-zui-chang-zi-chuan-by-leetc-2/



复杂度分析

- 时间复杂度：$O(N)$，其中 N 是字符串的长度。左指针和右指针分别会遍历整个字符串一次。
- 空间复杂度：$O(|\Sigma|)$，其中 Σ 表示字符集（即字符串中可以出现的字符），$|\Sigma|$ 表示字符集的大小。在本题中没有明确说明字符集，因此可以默认为所有 ASCII 码在 $[0,128)$ 内的字符，即 $|\Sigma|=128$。我们需要用到哈希集合来存储出现过的字符，而字符最多有 $|\Sigma|$ 个，因此空间复杂度为 $O(|\Sigma|)$。

```
class Solution:
    def lengthOfLongestSubstring(self, s: str) -> int:
        # 作者：seventeenth
        # 链接：https://leetcode.cn/problems/longest-substring-without-repeating-characters/solution/zen-yao-yong-hua-dong-chuang-kou-wei-he
        left, right, length = 0, 0, 0
        cur_chars = set()

        for i in range(len(s)):
            while s[i] in cur_chars:
                cur_chars.remove(s[left])
                left += 1
            cur_chars.add(s[i])
            length = max(length, right - left + 1)
            right += 1

        return length
```

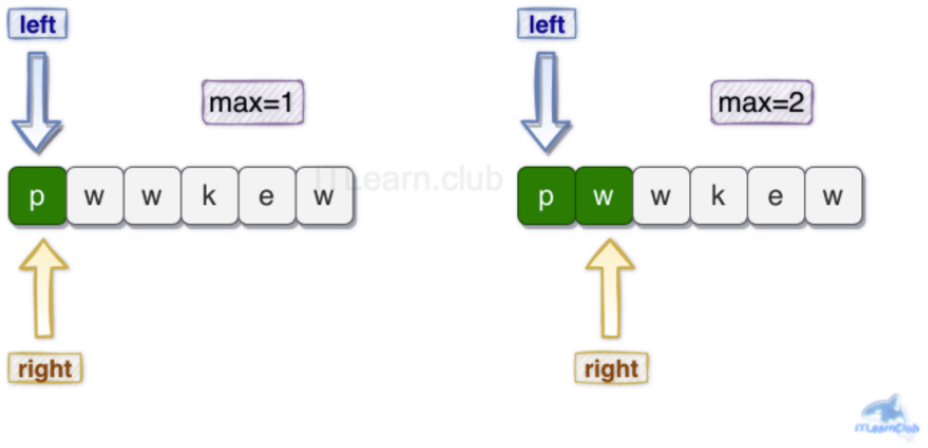
力扣

https://leetcode.cn/problems/longest-substring-without-repeating-characters/solution/zen-yao-yong-hua-dong-chuang-kou-wei-he-35418/

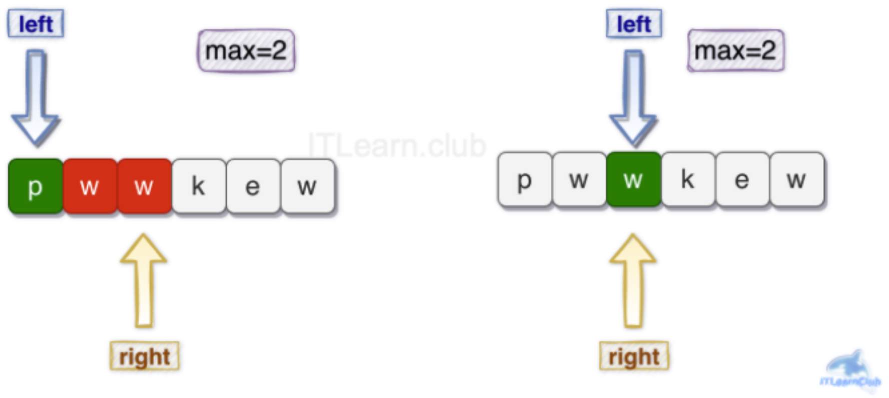
图解算法

> 定义 `left` 指针、 `right` 指针分别表示窗口的左端、右端；
>
> [`left` , `right`]区间内的字符串用 `HashSet` 实现判断重复操作，
>
> 随着[`left` , `right`]区间的变化对 `HashSet` 中的元素进行增减；
>
> 定义 `max` 变量用来存储 `不重复子串最大长度` 作为结果返回。

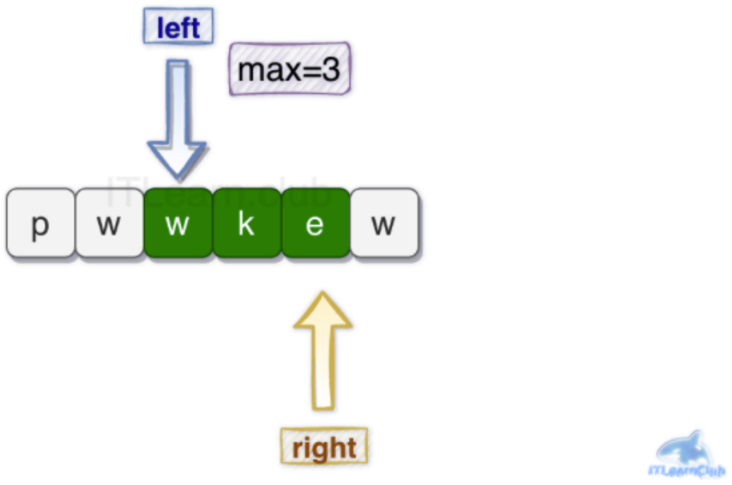
1. `left` 不变， `right` 向右移动，扩大[`left` , `right`]区间范围，同时更新 `max`



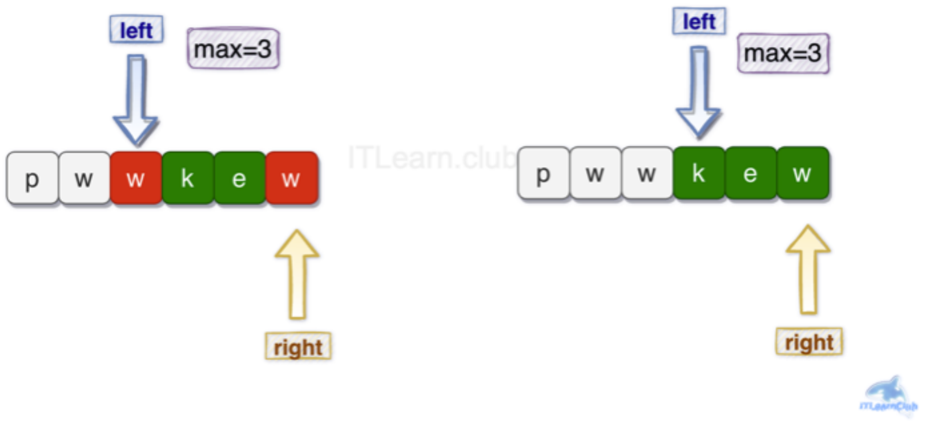
2. 继续移动 `right` 指针，发现区间内出现重复字符；移动 `left` 指针来消除重复



3. 发现更大的 不重复区间长度，更新 `max`



4. 继续移动 `right` 指针，发现区间内出现重复字符；移动 `left` 指针来消除重复



https://www.youtube.com/watch?v=9VcYiqTqzUY

```java
class Solution {
    public int lengthOfLongestSubstring(String s) {
        if (s == null || s.length() == 0) return 0;
        int left = 0, right = 0;
        int n = s.length();
        boolean[] used = new boolean[128];

        int max = 0;
        while (right < n) {
            if (used[s.charAt(right)] == false) {
                used[s.charAt(right)] = true;
                right++;
            } else {
                max = Math.max(max, right-left);
                while (left < right && s.charAt(right) != s.charAt(left)) {
                    used[s.charAt(left)] = false;
                    left++;
                }
                left++;
                right++;
            }
        }
        max = Math.max(max, right-left);
        return max;
    }
}
```