Leetcode3：无重复字符的最长子串：

给定一个字符串 `s` ，请你找出其中不含有重复字符的 **最长子串** 的长度。

示例 1:

> **输入：** s = "abcabcbb"
> **输出：** 3
> **解释：** 因为无重复字符的最长子串是 "abc"，所以其长度为 3。

示例 2:

> **输入：** s = "bbbbb"
> **输出：** 1
> **解释：** 因为无重复字符的最长子串是 "b"，所以其长度为 1。

示例 3:

> **输入：** s = "pwwkew"
> **输出：** 3
> **解释：** 因为无重复字符的最长子串是 "wke"，所以其长度为 3。
> 请注意，你的答案必须是 **子串** 的长度，"pwke" 是一个*子序列*，不是子串。

算法思路：双指针算法：

1：定义两个指针 i ，j，（i <= j），表示当前扫描的子串是 [i , j] ，扫描过程中维护一个哈希表，表示 [i , j] 中每个字符出现的次数，线性扫描时，每次循环的流程如下：

1）：指针 j 向后移动一位，同时将哈希表中 s [j] 的个数 + 1

2）：假设 j 移动前的区间 [i, j] 中没有重复字符，则 j 移动后，只有 s [j] 可以出现两次，因此不断向后移动 i，直至区间 [i, j] 中 s [j] 的个数为 1

时间复杂度分析：O (n)

```java
class Solution {
    public int lengthOfLongestSubstring(String s) {
        HashMap<Character, Integer> map = new HashMap<>();
        int res = 0;
        for(int i = 0, j = 0; j < s.length(); j ++){
            map.put(s.charAt(j), map.getOrDefault(s.charAt(j), 0) + 1);
            while(map.get(s.charAt(j)) > 1){
```

```
                map.put(s.charAt(i), map.get(s.charAt(i)) - 1);
                i ++;
            }
            res = Math.max(res, j - i + 1);
        }
        return res;
    }
}
```