3 五重复字符的最长子串

Label：双指针
给定一个字符串，请你找出其中不含有重复字符的 最长子串 的长度。

```
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。

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。

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。
```

- 双指针（滑动窗口）

```java
class Solution {
    public int lengthOfLongestSubstring(String s) {

        if (s == null || s.length() == 0) {
            return 0;
        }

        HashMap<Character, Integer> map = new HashMap<>();
        char[] c = s.toCharArray();
        int maxLength = 0;

        for (int start = 0, end = 0; end < c.length; end++) {
            if (map.containsKey(c[end])) {
                // 如果是在 start 之前出现过的字符，则不需要改变 start，因为子串中就不包含它，也就不会重复
                start = Math.max(map.get(c[end]) + 1 , start);    // 因为后面肯定会包含这个重复的字符，所以把前面重复的剔除所以要 +1,
            }

            maxLength = Math.max(maxLength, end - start + 1);  // 计算长度
            map.put(c[end], end);
        }
        return maxLength;
    }
}
```

- 双指针 性能上的优化

```java
class Solution {
    public int lengthOfLongestSubstring(String s) {
        int n = s.length();
        if(n == 0) {
            return 0;
        }
        //  本题中字符串只含有英文字母，符号和数字，所以可以使用数组来代替哈希表，提高效率。
        int[] num = new int[128];
        int res = 0;
        int left = 0, right = 0;
        char[] cs = s.toCharArray();

        while(right < n) {
            //每次循环都将右侧指针向前移动一位，并将右侧指针所指向的字符的数量增加1
            //(byte) cs[right]表示将字符cs[right]转换为其所对应的ASCII码，在0~127之间，
            num[(byte) cs[right]]++;

            //如果此时右侧指针所对应的字符的数量超过1，表示已经有了重复字符，将左指针右移
            while(num[(byte) cs[right]] > 1) {
                num[(byte) cs[left++]]--;
            }
            //更新结果，取之前的结果与当前窗口长度的最大值
            res = Math.max(res, right - left + 1);

            right++;
        }
        return res;
    }
}
```