```swift
import Foundation
/// **剑指 Offer 48. 最长不含重复字符的子字符串**
/// 请从字符串中找出一个最长的不包含重复字符的子字符串，计算该最长子字符串的长度。
///
/// 示例 1:
/// 输入: "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: "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。**
class Solution {
    /// 设置一个`dp`用于表示动态规划表表示以每一个字符为结尾的最长不重复字符串的长度
    /// `dp[j]`表示以字符`s[j]`为结尾的最长不重复字符串的长度
    /// 显然`dp[j]`的最大值取决于与`s[j]`相同的字符所在的位置
    /// 标记左侧最近的相同字符为`s[i]`, 显然有`i < j`.
    ///
    /// `dp`的递推关系与`j-i`的长度有关系:
    /// - 当`dp[j-1] < j-i 时候 dp[j] = dp[j-1] + 1`
        此时说明最近的相同字符在当前最长字符之外
    /// - 当`dp[j-1] >= j-i 时候 dp[j] = j - i` 此时说明最近的相同字符在当前最长字符里面
    func lengthOfLongestSubstring(_ s: String) -> Int {
        /// charLatestPositionDict 的 key 表示每一个字符，value
            表示使用各字符最后一次出现的索引位置 。
        var charLatestPositionDict: [Character: Int] = [:]
        /// temp 用于存储当前字符的`dp[j]`数值, res 用于存储最终的结果
        var (temp, res): (Int, Int) = (0, 0)
        s.enumerated().forEach { j, c in
            /// 获取当前同当前字符最近的相同字符的索引
            /// 字典的默认返回值可以设置成任意的非负整数
            let i = charLatestPositionDict[c] ?? -1
            /// 更新各字符最后一次出现的索引位置
            charLatestPositionDict[c] = j
            /// 根据递推关系更新以当前字符为结尾的最长不重复字符串的长度
            temp = (temp < j-i) ? temp + 1 : j - i
            /// 获取最大的数值作为结果
            res = max(res, temp)
        }
        return res
    }
}
```

```
let solution = Solution()
print(solution.lengthOfLongestSubstring("abcabcbb"))
print(solution.lengthOfLongestSubstring("bbbbbbb"))
print(solution.lengthOfLongestSubstring("pwwkew"))
print(solution.lengthOfLongestSubstring(" "))
print(solution.lengthOfLongestSubstring(""))
print(solution.lengthOfLongestSubstring("dvdf"))
```