178. 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

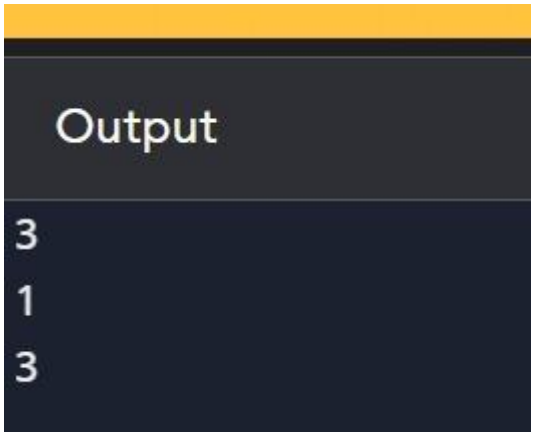
Explanation: The answer is "wke", with the length of 3.

Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.
Constraints: ● 0 <= s.length <= 5 * 104 ● s consists of English letters, digits, symbols and spaces.

Program:

```
def length_of_longest_substring(s):
    start = maxLength = 0
    used_chars = {}

    for i, char in enumerate(s):
        if char in used_chars and start <= used_chars[char]:
            start = used_chars[char] + 1
        else:
            maxLength = max(maxLength, i - start + 1)

        used_chars[char] = i

    return maxLength

# Test the function
s1 = "abcabcbb"
s2 = "bbbbb"
s3 = "pwwkew"

print(length_of_longest_substring(s1))  # Output: 3
print(length_of_longest_substring(s2))  # Output: 1
print(length_of_longest_substring(s3))  # Output: 3
```

Output:



Timecomplexity: O(n^2)