3. Longest Substring Without Repeating Characters

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.
```

Example 4:

```
Input: s = ""
Output: 0
```

Constraints:

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

```
class Solution:
    def lengthOfLongestSubstring(self, s: str) -> int:
        if len(s)==0 or len(s)==1:
            return len(s)
        ans = 0
```

```
        i = -1
        j = -1
        freq = {}
        while True:
            f1,f2 = False,False

            while i<len(s)-1:
                f1 = True
                i = i+1
                ch = s[i]
                freq[ch] = freq.get(ch,0)+1

                if freq[ch]==2:
                    break
                else:
                    pAns = i-j
                    ans = max(ans,pAns)

            while j<i:
                f2 = True
                j = j+1
                ch = s[j]
                freq[ch] = freq.get(ch,0)-1
                if freq[ch]==1:
                    break

            if f1 is False and f2 is False:
                break
        return ans
```