class Solution:

    def lengthOfLongestSubstring(self, s: str) -> int:

        char\_index\_map = {}  # Stores the last seen index of each character

        max\_length = 0

        left = 0  # Left pointer of the sliding window

        for right in range(len(s)):

            current\_char = s[right]

            # If the current character is already in the map and its last seen index

            # is within the current window (i.e., greater than or equal to 'left'),

            # then we need to move the 'left' pointer to the right of the repeated character's last occurrence.

            if current\_char in char\_index\_map and char\_index\_map[current\_char] >= left:

                left = char\_index\_map[current\_char] + 1

            # Update the last seen index of the current character

            char\_index\_map[current\_char] = right

            # Calculate the current window length and update max\_length

            max\_length = max(max\_length, right - left + 1)

        return max\_length