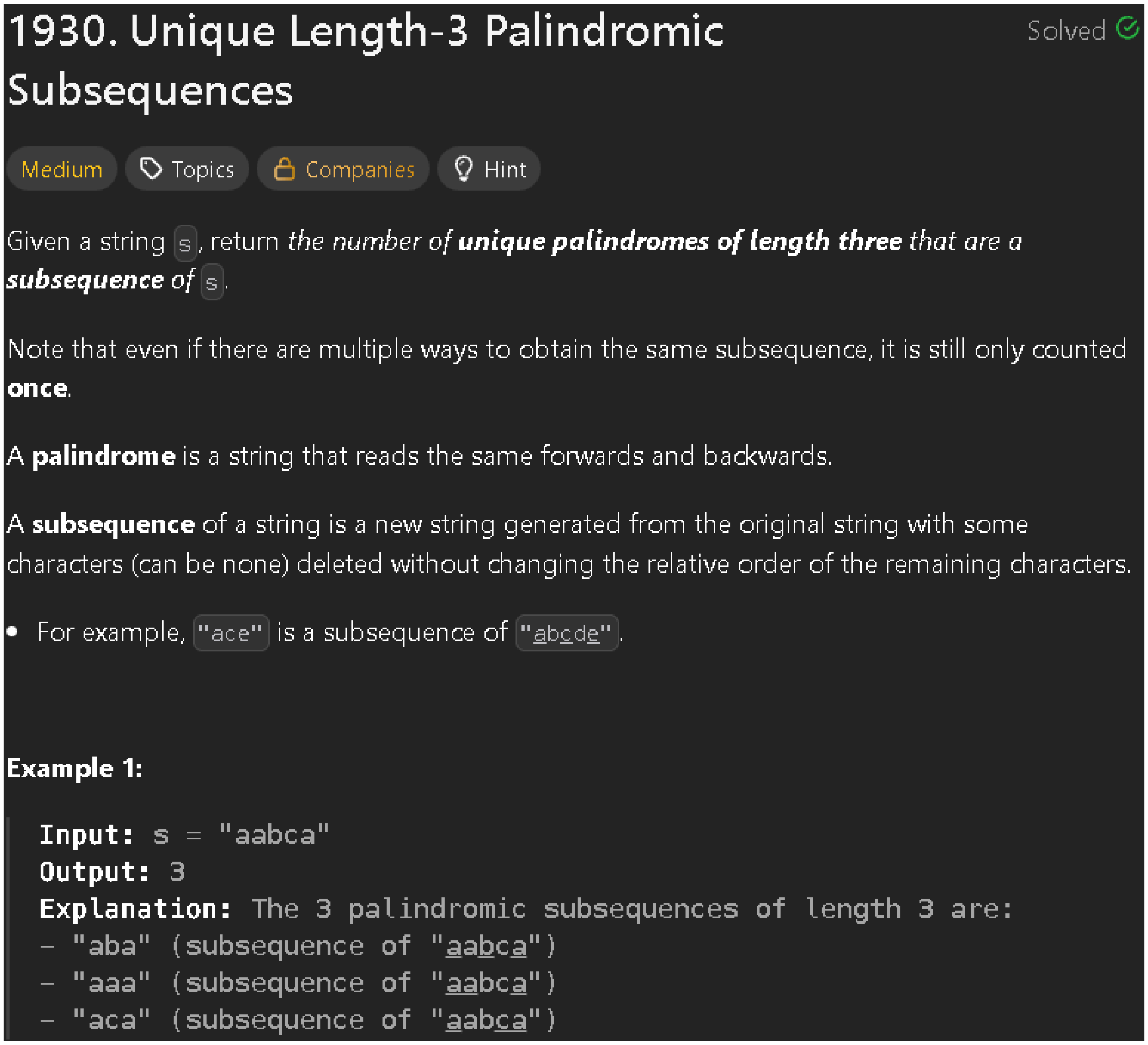


1930. Unique Length-3 Palindromic Subsequences

Solved ✓

Medium | Topics | Companies | Hint

Given a string `s`, return *the number of* ***unique palindromes of length three*** *that are a* ***subsequence*** *of* `s`.

Note that even if there are multiple ways to obtain the same subsequence, it is still only counted **once**.

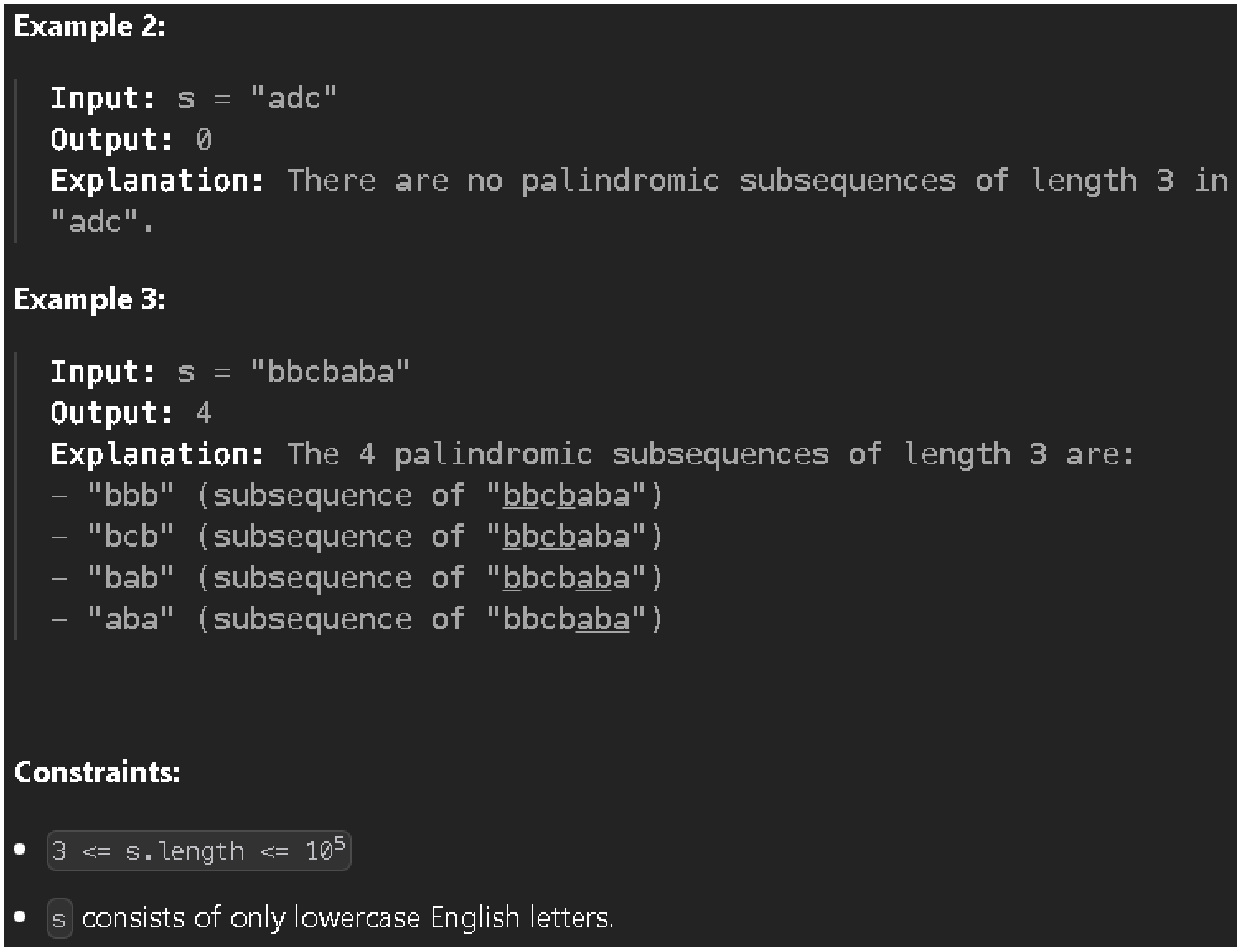
A **palindrome** is a string that reads the same forwards and backwards.

A **subsequence** of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters.

- For example, `"ace"` is a subsequence of `"abcde"`.

Example 1:

```
Input: s = "aabca"
Output: 3
Explanation: The 3 palindromic subsequences of length 3 are:
- "aba" (subsequence of "aabca")
- "aaa" (subsequence of "aabca")
- "aca" (subsequence of "aabca")
```


Example 2:

```
Input: s = "adc"
Output: 0
Explanation: There are no palindromic subsequences of length 3 in
"adc".
```

Example 3:

```
Input: s = "bbcbaba"
Output: 4
Explanation: The 4 palindromic subsequences of length 3 are:
- "bbb" (subsequence of "bbcbaba")
- "bcb" (subsequence of "bbcbaba")
- "bab" (subsequence of "bbcbaba")
- "aba" (subsequence of "bbcbaba")
```

Constraints:

- `3 <= s.length <= 10`^{5}
- `s` consists of only lowercase English letters.

Python:

```
class Solution:
    def countPalindromicSubsequence(self, inputString):
        # Arrays to store the minimum and maximum occurrences of each character in the input
string
        min_exist = [float('inf')] * 26
        max_exist = [float('-inf')] * 26

        # Populate min_exist and max_exist arrays
        for i in range(len(inputString)):
            char_index = ord(inputString[i]) - ord('a')
            min_exist[char_index] = min(min_exist[char_index], i)
            max_exist[char_index] = max(max_exist[char_index], i)

        # Variable to store the final count of unique palindromic subsequences
        unique_count = 0

        # Iterate over each character in the alphabet
        for char_index in range(26):
            # Check if the character has occurred in the input string
            if min_exist[char_index] == float('inf') or max_exist[char_index] == float('-inf'):
                continue  # No occurrences, move to the next character

            # Set to store unique characters between the minimum and maximum occurrences
            unique_chars_between = set()

            # Iterate over the characters between the minimum and maximum occurrences
            for j in range(min_exist[char_index] + 1, max_exist[char_index]):
                unique_chars_between.add(inputString[j])

            # Add the count of unique characters between the occurrences to the final count
            unique_count += len(unique_chars_between)

        # Return the total count of unique palindromic subsequences
        return unique_count
```

JavaScript:

```
var countPalindromicSubsequence = function(s) {
    let res = 0;
    const uniq = new Set(s);

    for (const c of uniq) {
        const start = s.indexOf(c);
        const end = s.lastIndexOf(c);

        if (start < end) {
            res += new Set(s.slice(start + 1, end)).size;
        }
    }

    return res;
};
```

Java:

```
class Solution {
    public int countPalindromicSubsequence(String inputString) {
        // Arrays to store the minimum and maximum occurrences of each character in the input
string
        int[] minExist = new int[26];
        int[] maxExist = new int[26];
        for (int i = 0; i < 26; i++) {
            minExist[i] = Integer.MAX_VALUE;
            maxExist[i] = Integer.MIN_VALUE;
        }

        // Populate minExist and maxExist arrays
        for (int i = 0; i < inputString.length(); i++) {
            int charIndex = inputString.charAt(i) - 'a';
            minExist[charIndex] = Math.min(minExist[charIndex], i);
            maxExist[charIndex] = Math.max(maxExist[charIndex], i);
        }

        // Variable to store the final count of unique palindromic subsequences
        int uniqueCount = 0;

        // Iterate over each character in the alphabet
        for (int charIndex = 0; charIndex < 26; charIndex++) {
            // Check if the character has occurred in the input string
            if (minExist[charIndex] == Integer.MAX_VALUE || maxExist[charIndex] ==
Integer.MIN_VALUE) {
                continue; // No occurrences, move to the next character
            }

            // Set to store unique characters between the minimum and maximum occurrences
            HashSet<Character> uniqueCharsBetween = new HashSet<>();

            // Iterate over the characters between the minimum and maximum occurrences
            for (int j = minExist[charIndex] + 1; j < maxExist[charIndex]; j++) {
                uniqueCharsBetween.add(inputString.charAt(j));
            }

            // Add the count of unique characters between the occurrences to the final count
            uniqueCount += uniqueCharsBetween.size();
        }

        // Return the total count of unique palindromic subsequences
        return uniqueCount;
    }
}
```