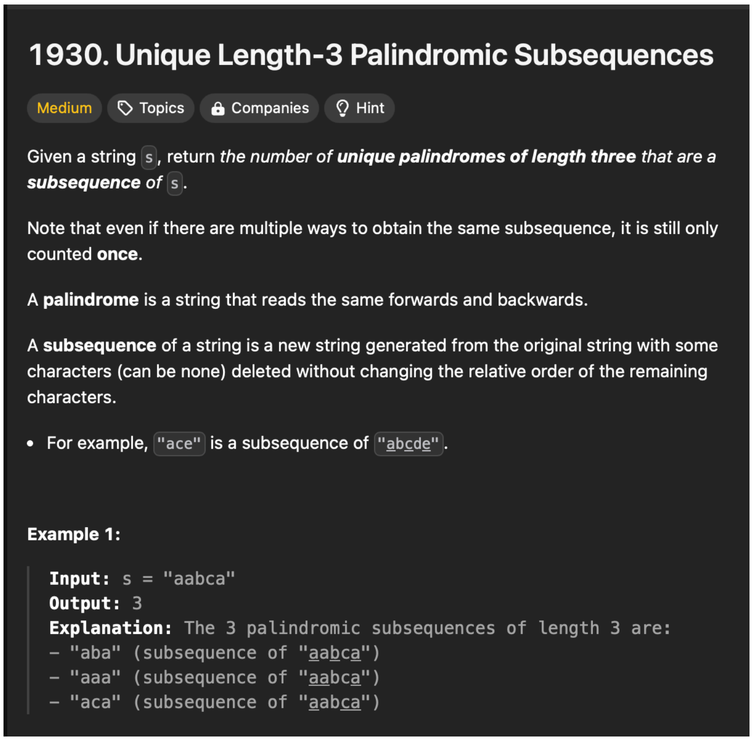


1930. Unique Length-3 Palindromic Subsequences

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.

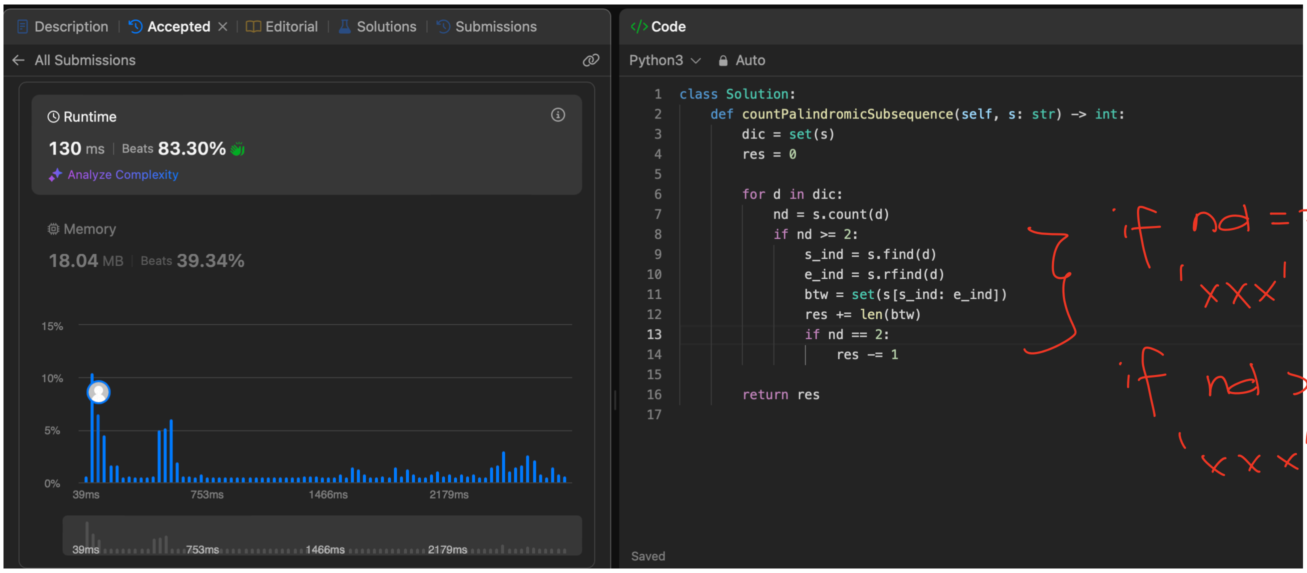
A **subsequence** of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters.

- 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")
```

```
dic = set ( s )

for d in dic :

    if count (d) >, 2

        btw = { }

?       btw += d_i ... d_{i+1}

        res += len(btw)
```



Runtime

130 ms | Beats 83.30%

Memory

18.04 MB | Beats 39.34%

```python
class Solution:
    def countPalindromicSubsequence(self, s: str) -> int:
        dic = set(s)
        res = 0

        for d in dic:
            nd = s.count(d)
            if nd >= 2:
                s_ind = s.find(d)
                e_ind = s.rfind(d)
                btw = set(s[s_ind: e_ind])
                res += len(btw)
                if nd == 2:
                    res -= 1

        return res
```

if nd == 2 :
'xxx' is not valid

if nd > 2 :
'xxx' is valid