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")
```

Example 2:

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

Example 3:

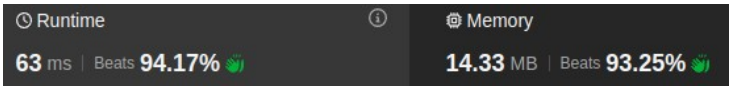
```
Input: s = "bbcbaba"
Output: 4
Explanation: The 4 palindromic subsequences of length 3 are:
- "bbb" (subsequence of "bbcbaba")
- "bcb" (subsequence of "bbcbaba")
- "bab" (subsequence of "bbcbaba")
- "aba" (subsequence of "bbcbaba")
```

Constraints:

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

1930. Unique Length-3 Palindromic Subsequences

```
/*
   Counting
   Time compelxity: O(26.2.n+26n+26.2)=O(78n+52)=O(n)
   Space compelxity: O(26+2.26)=O(78)=O(1)
*/
```




```
typedef std::pair<int,int> ii;
typedef std::vector<ii> vii;

typedef std::vector<int> vi;
typedef std::vector<vi> vvi;

class Solution {
public:
   // Function to get first and last position of every letter in s
   vii get_first_last_pos(std::string& s){
      int n=s.size();
      vii positions(26);
      for(int i=0;i<26;++i){
         positions[i]={-1,-1};
         int left=0,right=n-1;
         while(left<=right &&(positions[i].first==-1 || positions[i].second==-1)){
            if(s[left]-'a'==i){
               positions[i].first=left;
            }
            if(s[right]-'a'==i){
               positions[i].second=right;
            }
            if(positions[i].first==-1) left++;
            if(positions[i].second==-1) right--;
         }
      }
      return positions;
   }
```

```
int countPalindromicSubsequence(std::string s) {
        vii positions=get_first_last_pos(s);

        vvi freq(26,vi(26,0));

        // For each letter ine the alphbet
        for(int i=0;i<26;++i){
            // If it apprears in s
            int left=positions[i].first;
            int right=positions[i].second;
            if(left!=-1 && right!=-1 && right-left>1){
                // Mark its middle letter
                for(int j=left+1;j<=right-1;++j){
                    freq[i][s[j]-'a']=1;
                }
            }
        }

        // Determine the number of length-3 palindromic subsequences
        int ans=0;
        for(int i=0;i<26;++i){
            for(int j=0;j<26;++j){
                ans+=freq[i][j];
            }
        }
        return ans;
    }
};
```