647. Palindromic Substrings

Given a string `s`, return *the number of **palindromic substrings** in it*.

A string is a **palindrome** when it reads the same backward as forward.

A **substring** is a contiguous sequence of characters within the string.

Example 1:

```
Input: s = "abc"
Output: 3
Explanation: Three palindromic strings: "a", "b", "c".
```

Example 2:

```
Input: s = "aaa"
Output: 6
Explanation: Six palindromic strings: "a", "a", "a", "aa", "aa", "aaa".
```

Constraints:

- `1 <= s.length <= 1000`
- `s` consists of lowercase English letters.

```
def countSubstrings(self, s: str) -> int:
        dp = [[False]*len(s) for _ in range(len(s))]
        count = 0
        for gap in range(len(s)):
            i = 0
            j = gap

            while j<len(s):
                if gap==0:
                    dp[i][j] = True
                elif gap==1:
                    dp[i][j] = s[i]==s[j]
                else:
                    if s[i]==s[j] and dp[i+1][j-1]:
                        dp[i][j] = True
                    else:
```

```
                    dp[i][j] = False
            if dp[i][j]==True:
                count =count+1
            i  = i+1
            j = j+1
        return count
```