计数二进制子串

给定一个字符串 s，统计并返回具有相同数量 0 和 1 的非空（连续）子字符串的数量，并且这些子字符串中的所有 0 和所有 1 都是成组连续的。

重复出现（不同位置）的子串也要统计它们出现的次数。

示例 1：

输入： s = "00110011"

输出： 6

解释： 6 个子串满足具有相同数量的连续 1 和 0 ：**"0011"**、**"01"**、**"1100"**、**"10"**、**"0011"** 和 **"01"** 。

注意，一些重复出现的子串（不同位置）要统计它们出现的次数。

另外，**"00110011"** 不是有效的子串，因为所有的 **0**（还有 **1** ）没有组合在一起。

示例 2：

输入： s = "10101"

输出： 4

解释： 有 4 个子串：**"10"**、**"01"**、**"10"**、**"01"** ，具有相同数量的连续 1 和 0

```
class Solution:
    def countBinarySubstrings(self, s: str) -> int:
         n = len(s)
         if n<=1:
             return 0
        #记录当前重复字符串的个数
         curLen = 1
         #保存前面字符的个数
         preLen = 0
        ##记录子串的个数
         count = 0

         for i in range(1,n):
             if s[i]==s[i-1]:
                curLen+=1
             else:
                 preLen=curLen #当前的字符与前面的字符不相同时，将前面字符的个数保存
                curLen=1
```

```
            if preLen>=curLen:##preLen>=curLen:很重要，if only 等于，则只有
0011 的情况，当存在>情况时，就存在 01 这种情况。仔细读一下代码即可
                count+=1


        return count
```