696 Count Binary Substrings

2018年5月3日 20:47

Give a string s, count the number of non-empty (contiguous) substrings that have the same number of 0's and 1's, and all the 0's and all the 1's in these substrings are grouped consecutively.
Substrings that occur multiple times are counted the number of times they occur.
Example 1:
Input: "00110011"
Output: 6
Explanation: There are 6 substrings that have equal number of consecutive 1's and 0's: "0011", "01", "1100", "10", "0011", and "01".
Notice that some of these substrings repeat and are counted the number of times they occur.
Also, "00110011" is not a valid substring because **all** the 0's (and 1's) are not grouped together.
Example 2:
Input: "10101"
Output: 4
Explanation: There are 4 substrings: "10", "01", "10", "01" that have equal number of consecutive 1's and 0's.
Note:

- s.length will be between 1 and 50,000.
- s will only consist of "0" or "1" characters.

来自 <https://leetcode.com/problems/count-binary-substrings/description/>

给定一个字符串 s，计算具有相同数量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都是组合在一起的。
重复出现的子串要计算它们出现的次数。
示例 1 :
输入: "00110011"
输出: 6
解释: 有6个子串具有相同数量的连续1和0：“0011”，“01”，“1100”，“10”，“0011”和“01”。
请注意，一些重复出现的子串要计算它们出现的次数。
另外，“00110011”不是有效的子串，因为所有的0（和1）没有组合在一起。
示例 2 :
输入: "10101"
输出: 4
解释: 有4个子串：“10”，“01”，“10”，“01”，它们具有相同数量的连续1和0。
注意:

- s.length 在1到50,000之间。
- s 只包含“0”或“1”字符。

Solution for Python3:

```python
class Solution1:
    def countBinarySubstrings(self, s):
        """
        :type s: str
        :rtype: int
        """
        groups = [1]
        for i in range(1, len(s)):
            if s[i-1] != s[i]:
                groups.append(1)
            else:
                groups[-1] += 1
        ans = 0
        for i in range(1, len(groups)):
            ans += min(groups[i-1], groups[i])
        return ans

class Solution2:
    def countBinarySubstrings(self, s):
        """
        :type s: str
        :rtype: int
        """
        groups = [len(list(v)) for _, v in itertools.groupby(s)]
        return sum(min(a, b) for a, b in zip(groups, groups[1:]))

class Solution3:
    def countBinarySubstrings(self, s):
        """
        :type s: str
        :rtype: int
        """
        ans, prev, cur = 0, 0, 1
        for i in range(1, len(s)):
            if s[i-1] != s[i]:
                ans += min(prev, cur)
                prev, cur = cur, 1
            else:
                cur += 1
        return ans + min(prev, cur)
```

Solution for C++:

```cpp
class Solution1 {
public:
    int countBinarySubstrings(string s) {
        int groups[s.length()];
        int t = 0;
        groups[t] = 1;
        for (int i = 1; i < s.length(); i++) {
            if (s[i-1] != s[i])
                groups[++t] = 1;
            else
                groups[t]++;
        }
        int ans = 0;
        for (int i = 1; i <= t; i++)
            ans += min(groups[i-1], groups[i]);
        return ans;
    }
};

class Solution2 {
public:
    int countBinarySubstrings(string s) {
        int ans = 0, prev = 0, cur = 1;
        for (int i = 1; i < s.length(); i++) {
            if (s[i-1] != s[i]) {
                ans += min(prev, cur);
                prev = cur;
                cur = 1;
            } else {
                cur++;
            }
        }
        return ans + min(prev, cur);
    }
};
```