# 题目

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 。

提示：

1 <= s.length <= 10^5

s[i] 为 '0' 或 '1'

# 分析

## 方法一：按字符分组

思路：

我们可以遍历字符串 s，对于每个位置 i，我们可以从该位置开始向左和向右扩展，直到遇到不同的字符或到达字符串边界为止。在扩展的过程中，我们可以统计连续相同字符的数量，并将较小的数量加入答案中。

代码：

class Solution {

public:

int countBinarySubstrings(string s) {

int ans = 0; // 最终结果

int cur = 1; // 当前连续相同字符的数量

int pre = 0; // 前一个连续相同字符的数量

// 从第二个字符开始遍历

for (int i = 1; i < s.size(); ++i) {

// 如果当前字符和前一个字符相同，则增加当前连续相同字符的数量

if (s[i] == s[i - 1]) {

cur++;

} else { // 如果当前字符和前一个字符不同

ans += min(cur, pre); // 将较小的数量加入答案中

pre = cur; // 更新前一个连续相同字符的数量

cur = 1; // 重置当前连续相同字符的数量为1

}

}

// 将最后一个连续相同字符的数量加入答案中

ans += min(cur, pre);

return ans;

}

};