


## 696. Count Binary Substrings

Solved 

Easy Topics Companies Hint

Given a binary string `s`, return the number of non-empty 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:** `s = "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.

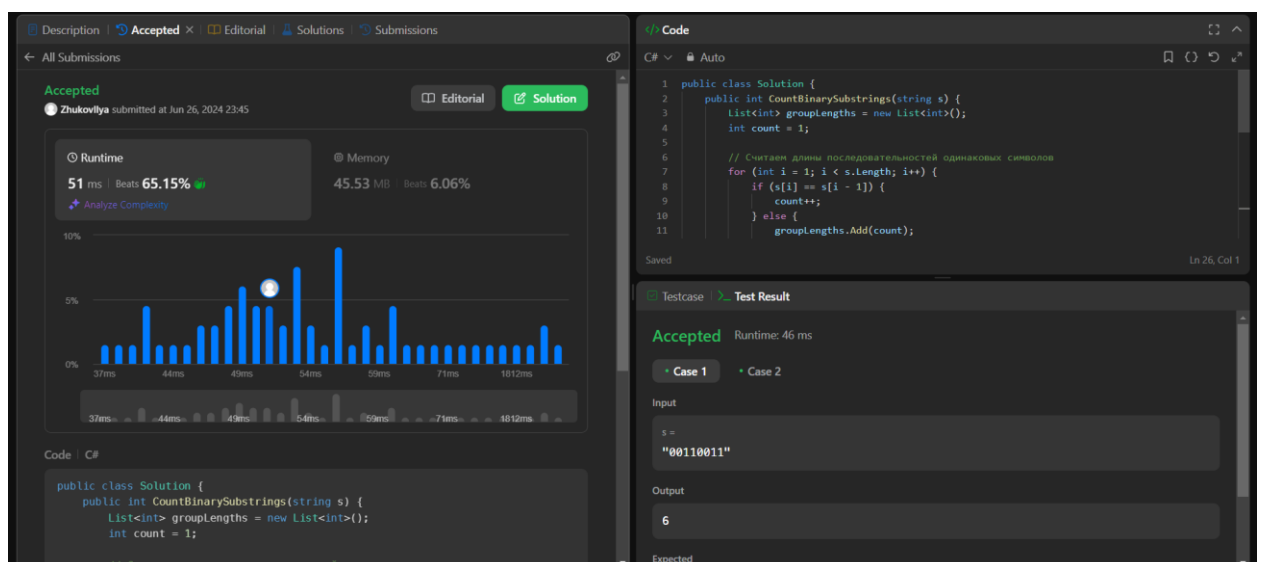
Also, "00110011" is not a valid substring because all the 0's (and 1's) are not grouped together.

### Example 2:

**Input:** `s = "10101"`

**Output:** 4

**Explanation:** There are 4 substrings: "10", "01", "10", "01" that have equal number of consecutive 1's and 0's.



The screenshot displays a coding platform interface. On the left, the 'Accepted' status is shown for a submission by 'ZhukovIlya' at Jun 26, 2024 23:45. Performance metrics include Runtime (51 ms, 65.15% beat) and Memory (45.53 MB, 6.06% beat). A bar chart shows the distribution of runtime results. Below the chart, the C# code for the solution is visible. On the right, the 'Code' editor shows the same C# code. Below the code, the 'Testcase' section shows 'Accepted' status with a runtime of 46 ms. The input is `s = "00110011"` and the output is `6`.

```
public class Solution {
    public int CountBinarySubstrings(string s) {
        List<int> groupLengths = new List<int>();
        int count = 1;

        // Считаем длины последовательностей одинаковых символов
        for (int i = 1; i < s.Length; i++) {
            if (s[i] == s[i - 1]) {
                count++;
            } else {
                groupLengths.Add(count);
                count = 1;
            }
        }
        groupLengths.Add(count);

        int result = 0;
        for (int i = 0; i < groupLengths.Count - 1; i++) {
            result += Math.Min(groupLengths[i], groupLengths[i + 1]);
        }

        return result;
    }
}
```

Код:

```
public class Solution
{
    public int CountBinarySubstrings(string s)
    {
        List<int> groupLengths = new List<int>();
        int count = 1;

        // Считаем длины последовательностей одинаковых символов
        for (int i = 1; i < s.Length; i++)
```

```
{
    if (s[i] == s[i - 1])
    {
        count++;
    }
    else
    {
        groupLengths.Add(count);
        count = 1;
    }
}
groupLengths.Add(count);

int result = 0;
// Находим пары соседних групп и добавляем минимумы их длин к результату
for (int i = 1; i < groupLengths.Count; i++)
{
    result += Math.Min(groupLengths[i - 1], groupLengths[i]);
}

return result;
}
}
```