🧩 Problem Statement (Simple Words)

You are given a string s consisting only of characters 'L' and 'R'.

A string is called **balanced** if:

Number of 'L' == Number of 'R'

Your task is to **split the string into the maximum number of balanced substrings** and return that count.

▯ Example

Input: s = "RLRRLLRLRL"

Output: 4

Possible splits:

"RL" | "RRLL" | "RL" | "RL"

Each substring has equal 'L' and 'R'.

● APPROACH 1: Using TWO Counters (Lcount & Rcount)

💡 Idea

- Traverse the string
- Count 'L' and 'R' separately
- Whenever Lcount == Rcount, one balanced string is formed

▯ Dry Run (Step by Step)

Input

s = "RLRRLLRLRL"

Index	Char	Lcount	Rcount	Balanced?	Result
0	R	0	1	✗	0
1	L	1	1	✅	1
2	R	1	2	✗	1
3	R	1	3	✗	1
4	L	2	3	✗	1

Index	Char	Lcount	Rcount	Balanced?	Result
5	L	3	3	✅	2
6	R	3	4	❌	2
7	L	4	4	✅	3
8	R	4	5	❌	3
9	L	5	5	✅	4

✔ Final Answer = 4

⏱ Complexity

- **Time:** O(n)
- **Space:** O(1)

💻 Code – TWO Counter Approach

C++

```
class Solution {
public:
    int balancedStringSplit(string s) {
        int L = 0, R = 0, result = 0;

        for (char c : s) {
            if (c == 'L') L++;
            else R++;

            if (L == R) result++;
        }
        return result;
    }
};
```

Java

```java
class Solution {
    public int balancedStringSplit(String s) {
        int L = 0, R = 0, result = 0;

        for (char c : s.toCharArray()) {
            if (c == 'L') L++;
            else R++;

            if (L == R) result++;
        }
        return result;
    }
}
```

Python

```python
class Solution:
    def balancedStringSplit(self, s: str) -> int:
        L = R = result = 0

        for c in s:
            if c == 'L':
                L += 1
            else:
                R += 1

            if L == R:
                result += 1

        return result
```

□ APPROACH 2: Using ONE Counter (Optimized)

💡 Key Observation

- Treat 'L' as **+1**
- Treat 'R' as **−1**
- When counter becomes **0**, a balanced string is found

▯ Dry Run (One Counter)

Index	**Char**	**Count**	**Balanced?**	**Result**
0	R	-1	❌	0
1	L	0	✅	1
2	R	-1	❌	1
3	R	-2	❌	1
4	L	-1	❌	1
5	L	0	✅	2
6	R	-1	❌	2
7	L	0	✅	3
8	R	-1	❌	3
9	L	0	✅	4

✔ Final Answer = **4**

⏱ Complexity

- **Time:** O(n)
- **Space:** O(1) ✅ (best)

💻 Code – ONE Counter Approach (Recommended)

C++

```
class Solution {
public:
    int balancedStringSplit(string s) {
        int count = 0, result = 0;

        for (char c : s) {
            if (c == 'L') count++;
            else count--;

            if (count == 0) result++;
        }
        return result;
    }
};
```

Java

```
class Solution {
    public int balancedStringSplit(String s) {
        int count = 0, result = 0;

        for (char c : s.toCharArray()) {
            if (c == 'L') count++;
            else count--;

            if (count == 0) result++;
        }
        return result;
    }
}
```

Python

```
class Solution:
    def balancedStringSplit(self, s: str) -> int:
        count = result = 0
        for c in s:
            count += 1 if c == 'L' else -1
            if count == 0:
                result += 1

        return result
```

🔥 Final Comparison

Approach	Variables	Space	Interview
Two Counters	L & R	O(1)	Good
One Counter	Single	O(1)	⭐ Best