# 01.01.2025

## *Problem Understanding*:

You are given a binary string `s` consisting of `'0'`s and `'1'`s. The goal is to split this string into two **non-empty substrings** such that the **score** is maximized.

**Score Definition:**

- The score is calculated as the sum of:
  a. Number of `'0'`s in the **left substring**.
  b. Number of `'1'`s in the **right substring**.

**Objective:**
Maximize the score after the split.

---

### Code Walkthrough:

```
#include<bits/stdc++.h>
using namespace std;

class Solution {
public:
    int maxScore(string s) {
        int n = s.size();

        // Step 1: Count the total number of '1's
in the string
        int ones = 0;
        for(int i = 0; i < n; i++){
```

```
            if(s[i] == '1'){
                ones++;
            }
        }

        // Step 2: Initialize variables to track
zero count and maximum score
        int zeroes = 0;
        int maxScore = 0;

        // Step 3: Simulate the split by iterating
from the start to the second last character
        // Note: The split must happen before the
last character, so loop till n-1
        for(int i = 0; i < n - 1; i++){
            if(s[i] == '1')
                ones--;       // Moving '1' to the
left, decrease the count of right-side ones
            else
                zeroes++;     // Moving '0' to the
left, increase the count of left-side zeroes

            // Step 4: Calculate the score and
update the maximum score
            maxScore = max(maxScore, zeroes +
ones);
        }

        // Step 5: Return the maximum score
obtained
        return maxScore;
    }
};
```

---

## Key Insight:

- The string is never physically split.
- Instead, the code simulates the split by iterating over the string and keeping track of the counts of `'0'` and `'1'`.

## Explanation with an Example:

For `s = "011101"`:

- **Total ones = 4**
- Initialize `zeroes = 0`, `maxScore = 0`

**Iteration Steps:**

- `i = 0`:
  a. `s[0] = 0 → zeroes = 1`, `maxScore = max(0, 1 + 4) = 5`
- `i = 1`:
  a. `s[1] = 1 → ones = 3`, `maxScore = max(5, 1 + 3) = 5`
- `i = 2`:
  a. `s[2] = 1 → ones = 2`, `maxScore = max(5, 1 + 2) = 5`
- `i = 3`:
  a. `s[3] = 1 → ones = 1`, `maxScore = max(5, 1 + 1) = 5`
- `i = 4`:
  a. `s[4] = 0 → zeroes = 2`, `maxScore = max(5, 2 + 1) = 5`

**Final Answer:** `5`

---

## Why This Works:

- **Efficiency:** Only one full pass is used to count the `1`s, and another for the simulation (`O(n)` time complexity).
- **Space Complexity:** Only a few variables are used (`O(1)` space complexity).

---