1422. Maximum Score After Splitting a String

Given a string `s` of zeros and ones, *return the maximum score after splitting the string into two* ***non-empty*** *substrings* (i.e. **left** substring and **right** substring).

The score after splitting a string is the number of **zeros** in the **left** substring plus the number of **ones** in the **right** substring.

Example 1:

```
Input: s = "011101"
Output: 5
Explanation:
All possible ways of splitting s into two non-empty substrings are:
left = "0" and right = "11101", score = 1 + 4 = 5
left = "01" and right = "1101", score = 1 + 3 = 4
left = "011" and right = "101", score = 1 + 2 = 3
left = "0111" and right = "01", score = 1 + 1 = 2
left = "01110" and right = "1", score = 2 + 1 = 3
```

Example 2:

```
Input: s = "00111"
Output: 5
Explanation: When left = "00" and right = "111", we get the maximum score = 2 + 3 =
5
```

Example 3:

```
Input: s = "1111"
Output: 3
```

Constraints:

- `2 <= s.length <= 500`
- The string `s` consists of characters `'0'` and `'1'` only.

1422. Maximum Score After Splitting a String

```
/*
   Prefix array + Suffix array
   Time compelxity: O(5n)
   Space complexity: O(2n)
*/
class Solution {
public:
   int maxScore(std::string s){
      int n=s.size();

      std::vector<int> prefix_0(n);
      prefix_0[0]=int(s[0]=='0');
      for(int i=1;i<n;++i) prefix_0[i]=prefix_0[i-1]+int(s[i]=='0');

      std::vector<int> suffix_1(n);
      suffix_1[n-1]=int(s[n-1]=='1');
      for(int i=n-2;i>=0;--i) suffix_1[i]=suffix_1[i+1]+int(s[i]=='1');

      int ans=0;
      for(int i=0;i<n-1;++i){
         ans=std::max(ans,prefix_0[i]+suffix_1[i+1]);
      }
      return ans;
   }
};
```

1422. Maximum Score After Splitting a String

```
/*
   Space optimization
   Time compelxity: O(2n)
   Space complexity: O(1)
*/
class Solution {
public:
   int maxScore(std::string s){
      int n=s.size();

      int ones_in_right=0;
      for(auto& c: s){
         if(c=='1') ones_in_right++;
      }

      int zeros_in_left=0,ans=0;
      for(int i=0;i<n-1;++i){
         if(s[i]=='0') zeros_in_left++;
         else ones_in_right--;
         ans=std::max(ans,zeros_in_left+ones_in_right);
      }
      return ans;
   }
};
```

1422. Maximum Score After Splitting a String

```
/*
   One pass
   Time compelxity: O(n)
   Space complexity: O(1)
*/
class Solution {
public:
   int maxScore(std::string s){
      int n=s.size();
      int zeros_in_left=0,ones_in_left=0,ans=INT_MIN;
      for(int i=0;i<n-1;++i){
         if(s[i]=='0') zeros_in_left++;
         else ones_in_left++;
         ans=std::max(ans,zeros_in_left-ones_in_left);
      }
      return ans+ones_in_left+int(s[n-1]=='1');
   }
};
```