1653. Minimum Deletions to Make String Balanced

You are given a string `s` consisting only of characters `'a'` and `'b'`.

You can delete any number of characters in `s` to make `s` **balanced**. `s` is **balanced** if there is no pair of indices `(i,j)` such that `i < j` and `s[i] = 'b'` and `s[j]= 'a'`.

Return *the* ***minimum*** *number of deletions needed to make* `s` ***balanced***.

Example 1:

```
Input: s = "aababbab"
Output: 2
Explanation: You can either:
Delete the characters at 0-indexed positions 2 and 6 ("aababbab" -> "aaabbb"), or
Delete the characters at 0-indexed positions 3 and 6 ("aababbab" -> "aabbbb").
```

Example 2:

```
Input: s = "bbaaaaabb"
Output: 2
Explanation: The only solution is to delete the first two characters.
```

Constraints:

- `1 <= s.length <= 105`
- `s[i]` is `'a'` or `'b'`.

1653. Minimum Deletions to Make String Balanced

```
/*
   Brute force
    Time complexity: O(2n)=O(n)
    Extra space complexity: O(n)
*/
typedef std::vector<int> vi;

class Solution {
public:
    int minimumDeletions(std::string s) {
        int n=s.size();
        vi suff_a_in_right(n,0);
        for(int i=n-2;i>=0;--i){
            suff_a_in_right[i]+=(s[i+1]=='a')?suff_a_in_right[i+1]+1:suff_a_in_right[i+1];
        }

        int b_in_left=0,ans=INT_MAX;

        for(int i=0;i<n;++i){
            ans=std::min(ans,b_in_left+suff_a_in_right[i]);
            if(s[i]=='b') b_in_left++;
        }

        return ans;
    }
};
```

1653. Minimum Deletions to Make String Balanced

```
/*
   Brute force: optimize space
    Time complexity: O(2n)=O(n)
    Extra space complexity: O(1)
*/
typedef std::vector<int> vi;

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

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

        int b_in_left=0,a_in_left=0,ans=INT_MAX;

        for(int i=0;i<n;++i){
            if(s[i]=='a') a_in_left++;
            ans=std::min(ans,b_in_left+(total_a-a_in_left));
            if(s[i]=='b') b_in_left++;
        }

        return ans;
    }
};
```