Problem 3746: Minimum String Length After Balanced Removals

Problem Information

Difficulty: Medium
Acceptance Rate: 78.06%
Paid Only: No
Tags: String, Stack, Counting

Problem Description

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

You are allowed to repeatedly remove **any substring** where the number of `'a'` characters is equal to the number of `'b'` characters. After each removal, the remaining parts of the string are concatenated together without gaps.

Return an integer denoting the **minimum possible length** of the string after performing any number of such operations.

Example 1:

Input: s = `"aabbab"`

Output: 0

Explanation:

The substring `"aabbab"` has three `'a'` and three `'b'`. Since their counts are equal, we can remove the entire string directly. The minimum length is 0.

Example 2:

Input: s = `"aaaa"`

Output: 4

Explanation:

Every substring of `"aaaa"` contains only `'a'` characters. No substring can be removed as a result, so the minimum length remains 4.

Example 3:

Input: s = `"aaabb"`

Output: 1

Explanation:

First, remove the substring `"ab"`, leaving `"aab"`. Next, remove the new substring `"ab"`, leaving `"a"`. No further removals are possible, so the minimum length is 1.

Constraints:

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

Code Snippets

C++:

```
class Solution {
public:
int minLengthAfterRemovals(string s) {

}
};
```

Java:

```
class Solution {
public int minLengthAfterRemovals(String s) {

}
```

```
}
```

Python3:

```
class Solution:
def minLengthAfterRemovals(self, s: str) -> int:
```