1750. Minimum Length of String After Deleting Similar Ends

Given a string `s` consisting only of characters `'a'`, `'b'`, and `'c'`. You are asked to apply the following algorithm on the string any number of times:

1. Pick a **non-empty** prefix from the string `s` where all the characters in the prefix are equal.
2. Pick a **non-empty** suffix from the string `s` where all the characters in this suffix are equal.
3. The prefix and the suffix should not intersect at any index.
4. The characters from the prefix and suffix must be the same.
5. Delete both the prefix and the suffix.

Return *the **minimum length** of* `s` *after performing the above operation any number of times (possibly zero times)*.

Example 1:

```
Input: s = "ca"
Output: 2
Explanation: You can't remove any characters, so the string stays as is.
```

Example 2:

```
Input: s = "cabaabac"
Output: 0
Explanation: An optimal sequence of operations is:
- Take prefix = "c" and suffix = "c" and remove them, s = "abaaba".
- Take prefix = "a" and suffix = "a" and remove them, s = "baab".
- Take prefix = "b" and suffix = "b" and remove them, s = "aa".
- Take prefix = "a" and suffix = "a" and remove them, s = "".
```

Example 3:

```
Input: s = "aabccabba"
Output: 3
Explanation: An optimal sequence of operations is:
- Take prefix = "aa" and suffix = "a" and remove them, s = "bccabb".
- Take prefix = "b" and suffix = "bb" and remove them, s = "cca".
```

Constraints:

- `1 <= s.length <= 10<sup>5</sup>`
- `s` only consists of characters `'a'`, `'b'`, and `'c'`.

```python
class Solution:
    def minimumLength(self, s: str) -> int:
        if len(s)==1:
            return 1
        if s[0] != s[-1]:
            return len(s)
        i = 0
        j = len(s) - 1

        while i <= j:
            if s[i] == s[j] and i!=j:
                i += 1
                j -= 1
            elif s[i - 1] == s[i]:
                i += 1
            elif s[j] == s[j + 1]:
                j -= 1
            else:
                return j-i+1
        return 0
```