Problem 2193: Minimum Number of Moves to Make Palindrome

Problem Information

Difficulty: Hard
Acceptance Rate: 52.40%
Paid Only: No
Tags: Two Pointers, String, Greedy, Binary Indexed Tree

Problem Description

You are given a string `s` consisting only of lowercase English letters.

In one **move** , you can select any two **adjacent** characters of `s` and swap them.

Return _the**minimum number of moves** needed to make_ `s` _a palindrome_.

Note that the input will be generated such that `s` can always be converted to a palindrome.

Example 1:

Input: s = "aabb" **Output:** 2 **Explanation:** We can obtain two palindromes from s, "abba" and "baab". - We can obtain "abba" from s in 2 moves: "a _**ab**_ b" -> "ab _**ab**_ " -> "abba". - We can obtain "baab" from s in 2 moves: "a _**ab**_ b" -> "_**ab**_ ab" -> "baab". Thus, the minimum number of moves needed to make s a palindrome is 2.

Example 2:

Input: s = "letelt" **Output:** 2 **Explanation:** One of the palindromes we can obtain from s in 2 moves is "lettel". One of the ways we can obtain it is "lete _**lt**_ " -> "let _**et**_ l" -> "lettel". Other palindromes such as "tleelt" can also be obtained in 2 moves. It can be shown that it is not possible to obtain a palindrome in less than 2 moves.

Constraints:

* `1 <= s.length <= 2000` * `s` consists only of lowercase English letters. * `s` can be converted to a palindrome using a finite number of moves.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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