Problem 1960: Maximum Product of the Length of Two Palindromic Substrings

Problem Information

Difficulty: Hard
Acceptance Rate: 30.63%
Paid Only: No
Tags: String, Rolling Hash, Hash Function

Problem Description

You are given a **0-indexed** string `s` and are tasked with finding two **non-intersecting palindromic** substrings of **odd** length such that the product of their lengths is maximized.

More formally, you want to choose four integers `i`, `j`, `k`, `l` such that `0 <= i <= j < k <= l < s.length` and both the substrings `s[i...j]` and `s[k...l]` are palindromes and have odd lengths. `s[i...j]` denotes a substring from index `i` to index `j` **inclusive**.

Return _the**maximum** possible product of the lengths of the two non- intersecting palindromic substrings._

A **palindrome** is a string that is the same forward and backward. A **substring** is a contiguous sequence of characters in a string.

Example 1:

Input: s = "ababbb" **Output:** 9 **Explanation:** Substrings "aba" and "bbb" are palindromes with odd length. product = 3 * 3 = 9.

Example 2:

Input: s = "zaaaxbbby" **Output:** 9 **Explanation:** Substrings "aaa" and "bbb" are palindromes with odd length. product = 3 * 3 = 9.

Constraints:

* `2 <= s.length <= 105` * `s` consists of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
long long maxProduct(string s) {

}
};
```

Java:

```
class Solution {
public long maxProduct(String s) {

}
}
```

Python3:

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