Problem 3501: Maximize Active Section with Trade II

Problem Information

Difficulty: Hard
Acceptance Rate: 18.37%
Paid Only: No
Tags: Array, String, Binary Search, Segment Tree

Problem Description

You are given a binary string `s` of length `n`, where:

* `'1'` represents an **active** section. * `'0'` represents an **inactive** section.

You can perform **at most one trade** to maximize the number of active sections in `s`. In a trade, you:

* Convert a contiguous block of `'1'`s that is surrounded by `'0'`s to all `'0'`s. * Afterward, convert a contiguous block of `'0'`s that is surrounded by `'1'`s to all `'1'`s.

Additionally, you are given a **2D array** `queries`, where `queries[i] = [li, ri]` represents a substring `s[li...ri]`.

For each query, determine the **maximum** possible number of active sections in `s` after making the optimal trade on the substring `s[li...ri]`.

Return an array `answer`, where `answer[i]` is the result for `queries[i]`.

Note

* For each query, treat `s[li...ri]` as if it is **augmented** with a `'1'` at both ends, forming `t = '1' + s[li...ri] + '1'`. The augmented `'1'`s **do not** contribute to the final count. * The queries are independent of each other.

Example 1:

Input: s = "01", queries = [[0,1]]

Output: [1]

Explanation:

Because there is no block of `'1'`s surrounded by `'0'`s, no valid trade is possible. The maximum number of active sections is 1.

Example 2:

Input: s = "0100", queries = [[0,3],[0,2],[1,3],[2,3]]

Output: [4,3,1,1]

Explanation:

* Query `[0, 3]` -> Substring `"0100"` -> Augmented to `"101001"` Choose `"0100"`, convert `"0100"` -> `"0000"` -> `"1111"`. The final string without augmentation is `"1111"`. The maximum number of active sections is 4.

* Query `[0, 2]` -> Substring `"010"` -> Augmented to `"10101"` Choose `"010"`, convert `"010"` -> `"000"` -> `"111"`. The final string without augmentation is `"1110"`. The maximum number of active sections is 3.

* Query `[1, 3]` -> Substring `"100"` -> Augmented to `"11001"` Because there is no block of `'1'`s surrounded by `'0'`s, no valid trade is possible. The maximum number of active sections is 1.

* Query `[2, 3]` -> Substring `"00"` -> Augmented to `"1001"` Because there is no block of `'1'`s surrounded by `'0'`s, no valid trade is possible. The maximum number of active sections is 1.

Example 3:

Input: s = "1000100", queries = [[1,5],[0,6],[0,4]]

Output: [6,7,2]

Explanation:

* Query `[1, 5]` -> Substring `"00010"` -> Augmented to `"1000101"` Choose `"00010"`, convert `"00010"` -> `"00000"` -> `"11111"`. The final string without augmentation is `"1111110"`. The maximum number of active sections is 6.

* Query `[0, 6]` -> Substring `"1000100"` -> Augmented to `"110001001"` Choose `"000100"`, convert `"000100"` -> `"000000"` -> `"111111"`. The final string without augmentation is `"1111111"`. The maximum number of active sections is 7.

* Query `[0, 4]` -> Substring `"10001"` -> Augmented to `"1100011"` Because there is no block of `'1'`s surrounded by `'0'`s, no valid trade is possible. The maximum number of active sections is 2.

Example 4:

Input: s = "01010", queries = [[0,3],[1,4],[1,3]]

Output: [4,4,2]

Explanation:

* Query `[0, 3]` -> Substring `"0101"` -> Augmented to `"101011"` Choose `"010"`, convert `"010"` -> `"000"` -> `"111"`. The final string without augmentation is `"11110"`. The maximum number of active sections is 4.

* Query `[1, 4]` -> Substring `"1010"` -> Augmented to `"110101"` Choose `"010"`, convert `"010"` -> `"000"` -> `"111"`. The final string without augmentation is `"01111"`. The maximum number of active sections is 4.

* Query `[1, 3]` -> Substring `"101"` -> Augmented to `"11011"` Because there is no block of `'1'`s surrounded by `'0'`s, no valid trade is possible. The maximum number of active sections is 2.

Constraints:

* `1 <= n == s.length <= 105` * `1 <= queries.length <= 105` * `s[i]` is either `'0'` or `'1'`. * `queries[i] = [li, ri]` * `0 <= li <= ri < n`

Code Snippets

C++:

```
class Solution {
public:
vector<int> maxActiveSectionsAfterTrade(string s, vector<vector<int>>&
queries) {

}
};
```

Java:

```
class Solution {
public List<Integer> maxActiveSectionsAfterTrade(String s, int[][] queries) {

}
}
```

Python3:

```
class Solution:
def maxActiveSectionsAfterTrade(self, s: str, queries: List[List[int]]) ->
List[int]:
```