2315. Count Asterisks

Easy String

Leetcode Link

Problem Description

In this problem, we are given a string `s`. The string contains some asterisks `'*'` and vertical bars `'|'`. The catches are that every two consecutive vertical bars form a pair, and we should not count any asterisks that are between each pair of vertical bars. For example, in the string `"*|*|*|*|*"`, there are three pairs formed by the vertical bars: the first `'|*|'`, the second `'*|'`, and the last two `'|'`. The asterisks we want to count are only those that are not between the vertical bars that form pairs. Therefore, we need to find a way to count asterisks that are not enclosed by any pairs of vertical bars in the string `s`.

The task is to find the total number of such asterisks in the given string.

Intuition

To solve this problem, we can iterate through the string `s` and maintain a boolean flag to indicate whether we are between a pair of vertical bars. We can initialize this flag to `True`, since we start counting asterisks from the very beginning of the string, where there are no vertical bars to enclose them. When we encounter a vertical bar, we switch the state of the flag. This is because vertical bars always come in pairs; the first one in a pair opens the enclosed section, and the second one closes it.

So when the flag is `True`, we know that any asterisks we see are not enclosed in vertical bars, and we should count them. And when the flag is `False`, we're between a pair of vertical bars, so we do not count the asterisks.

The variable `ok` in the given solution represents this flag, and `ans` is the count of asterisks that are not enclosed by the vertical bars. We use the bitwise XOR operator `^` to toggle the value of `ok` when we encounter a vertical bar. If `ok` is currently `1` (or `True`), it becomes `0` (or `False`) after the operation, and vice versa.

This solution is efficient since it only requires a single pass through the string and does not require additional data structures or complex logic.

Solution Approach

The solution for this problem is straightforward and does not require the use of complex data structures. Instead, it effectively utilizes a simple iteration and a flag switching technique to track the state of being inside or outside the vertical bar pairs.

Here's a step-by-step approach to how the solution is implemented:

1. **Initialize Variables**: Two variables are initialized:
   - `ans`: An integer initialized to `0`, which will serve as the counter for the asterisks not enclosed in vertical bar pairs.
   - `ok`: A boolean-like flag (represented as `1` for `True` and `0` for `False`), which helps determine whether we are currently outside (able to count `'*'`) or inside (not counting `'*'`) the vertical bar pairs.
2. **Iterate Through the String**: We loop through each character `c` in the string `s`.
3. **Counting Asterisks**: If the current character `c` is an asterisk (`'*'`), we increment the `ans` variable by the value of `ok`. Since `ok` is used as a boolean flag, when it's `1` (indicating we are outside of any vertical bar pair), `ans` gets incremented. If `ok` is `0` (meaning we're within a pair), there's no increment.
4. **Toggle Flag on Vertical Bars**: If we encounter a vertical bar (`'|'`), we toggle the value of `ok` with the expression `ok ^= 1`. This makes use of the bitwise XOR operation which flips `ok` between `1` and `0`. Thus, when a `'|'` is encountered, if `ok` was `1`, it becomes `0`, and if it was `0`, it becomes `1`. This signifies the start and end of the paired vertical bars.
5. **Return the Result**: Once the iteration over the string is complete, `ans` holds the count of all asterisks outside the vertical bar pairs, and this value is returned as the final answer.

The elegance of this approach lies in its simplicity and efficiency, as it only scans through the string once and uses bitwise operations for state toggling. This algorithm operates in O(n) time complexity where n is the length of string `s` and O(1) space complexity since no additional space is proportional to the input size is required.

Example Walkthrough

Let's walk through a small example using the solution approach with the string `"**|*|***|**"`.

1. **Initialize Variables**: We start by initializing `ans` to `0` and `ok` to `1`.
2. **Iterate Through the String**: We proceed to iterate over each character in the given string.
3. Upon the first character which is `'*'`, `ans` is incremented by `1` as `ok` is `1` (outside of vertical bar pairs), giving us `ans = 1`.
4. The second character is also `'*'` and `ok` still equals `1`, so we increment `ans` again to `2`.
5. We encounter our first vertical bar `'|'`. We toggle `ok` using `ok ^= 1`. As `ok` was `1`, it now becomes `0`.
6. The next character is `'*'`. Since `ok` is now `0`, we do not change `ans`.
7. We come across another vertical bar `'|'`. We toggle `ok` again, which changes it back to `1`.
8. Now we have three asterisks `'***'`, but as `ok` is `1`, we do not count these; we're inside a pair from step 6 and 7.
9. The next character is a vertical bar `'|'`, so we must toggle `ok`. It changes from `1` to `0`, indicating the start of a new enclosed section.
10. We then find two asterisks `'**'`, and since `ok` is `0`, we do not count these either.
11. Finally, we reach the last character which is another vertical bar `'|'`. We toggle `ok`, changing it from `0` to `1`.
12. **Return the Result**: The iteration has ended, and the total count of asterisks outside the vertical bar pairs is `2` which is the value of `ans`.

Throughout this example, we looped through the string and counted only the asterisks that are not enclosed by any pair of vertical bars by efficiently toggling our `ok` state upon every vertical bar encountered. The answer, in this case, is `2` since we have two asterisks at the beginning before encountering any pairs of vertical bars.

Python Solution

```python
class Solution:
    def countAsterisks(self, s: str) -> int:
        # Initialize a count for asterisks and a boolean flag for tracking the state
        asterisk_count = 0  # This will store the number of asterisks not enclosed in pairs of vertical bars
        is_outside_bars = True  # This flag will be True if we're outside vertical bars, False otherwise

        # Iterate over each character in the string
        for char in s:
            if char == "*":
                # Increment the count if we encounter an asterisk and we're outside the vertical bars
                if is_outside_bars:
                    asterisk_count += 1
            elif char == "|":
                # Toggle the flag when we encounter a vertical bar
                is_outside_bars = not is_outside_bars  # This will turn True to False, and False to True

        # Return the final count of asterisks outside of vertical bar pairs
        return asterisk_count
```

Java Solution

```java
class Solution {
    public int countAsterisks(String s) {
        // Initialize the answer to count asterisks not inside pair of bars
        int asteriskCount = 0;
        // ok will indicate the current state (outside the bars is 1, inside is 0)
        boolean outsideBars = true;

        // Iterate through each character in the string
        for (int i = 0; i < s.length(); ++i) {
            // Fetch the current character
            char ch = s.charAt(i);

            // If the character is an asterisk and we're currently outside bars
            if (ch == '*' && outsideBars) {
                // Increment the asterisk count
                asteriskCount++;
            } else if (ch == '|') {
                // Toggle the state between inside and outside bars
                outsideBars = !outsideBars;
            }
        }

        // Return the total number of asterisks counted outside bars
        return asteriskCount;
    }
}
```

C++ Solution

```cpp
class Solution {
public:
    // Function to count the number of asterisks (*) that are not between any pair of vertical bars (|)
    int countAsterisks(string s) {
        int count = 0;      // Initialize count of asterisks to 0
        bool outsideBar = true;  // Use a boolean to keep track of whether we are outside the bars (true) or inside (false)

        // Iterate through each character in the string
        for (char c : s) {
            if (c == '*') {
                // If the current character is an asterisk and we're outside the bars, increment our count
                count += outsideBar;
            } else if (c == '|') {
                // If the current character is a vertical bar, toggle the state of outsideBar
                outsideBar = !outsideBar;
            }
        }

        // Return the total count of asterisks
        return count;
    }
};
```

Typescript Solution

```typescript
// Counts the number of asterisks (*) in a given string that are not inside paired pipes (|).
// Paired pipes act as markers for sections that should be ignored when counting asterisks.
// The `countAsterisks` function takes a string and returns the count of valid asterisks.
function countAsterisks(s: string): number {
    // Variable to hold the count of asterisks
    let asterisksCount = 0;
    // Variable to track the current state;
    // 1 means we're considering asterisks,
    // 0 means we're between paired pipes.
    let shouldBeCounted = 1;
    // Iterate over each character in the string
    for (const char of s) {
        if (char === '*') {
            // Increment the count if the asterisk should be counted
            asterisksCount += shouldBeCounted;
        } else if (char === '|') {
            // Toggle the state when encountering a pipe
            shouldBeCounted ^= 1;
        }
    }
    // Return the count of asterisks found outside of paired pipes
    return asterisksCount;
}

```

Time and Space Complexity

The time complexity of the given code is `O(n)`, where `n` is the length of the input string `s`. Each character in the string is visited exactly once in a single pass, and constant-time operations are performed for each character.

The space complexity of the code is `O(1)` because the space used does not depend on the size of the input string. The variables `ans` and `ok` use a constant amount of space regardless of the input size.