Intuition

The code aims to compute an array where each element is the product of all elements in the input array except itself. It employs a two-pass approach. In the first pass, it calculates the prefix products for each element, indicating the product of all elements to the left of the current element. In the second pass, it calculates the postfix products, representing the product of all elements to the right of the current element. The final result is obtained by multiplying the corresponding prefix and postfix products for each element. This approach eliminates the need for division and ensures linear time complexity. The algorithm uses a constant amount of extra space, creating an output array to store the results.

Approach

Given an input list `nums`, the goal is to compute an output list where each element is the product of all elements in `nums` except the one at that index.

1. **Left-to-Right Prefix Products:**

   - Initialize an array `answer` of the same length as `nums` with all elements set to 1.
   - Initialize a variable `prefix` to 1.
   - Iterate through `nums` from left to right.
   - At each index `i`, update `answer[i]` with the current `prefix`.
   - Multiply `prefix` by the current element `nums[i]`.

2. **Right-to-Left Postfix Products:**

   - Initialize a variable `postfix` to 1.
   - Iterate through `nums` from right to left.
   - At each index `i`, multiply `answer[i]` by the current `postfix`.
   - Multiply `postfix` by the current element `nums[i]`.

3. **Output:**

   - The `answer` array now contains the product of all elements except the one at the current index.

Complexity

- Time complexity: O(n)

- Space complexity: O(1)
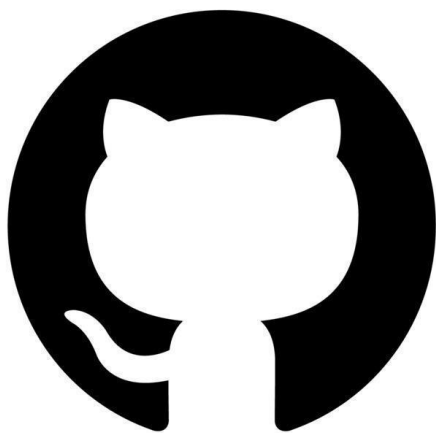
Code

```python
class Solution:
    def productExceptSelf(self, nums: List[int]) -> List[int]:
        answer = [1] * (len(nums))

        prefix = 1
        for i in range(len(nums)):
            answer[i] = prefix
            prefix *= nums[i]

        postfix = 1
        for i in range(len(nums) -1, -1, -1):
            answer[i] *= postfix
            postfix *= nums[i]

        return answer
```

If you want to see more solutions to coding problems, you can visit: