238. Product of Array Except Self

Given an integer array `nums`, return *an array* `answer` *such that* `answer[i]` *is equal to the product of all the elements of* `nums` *except* `nums[i]`.

The product of any prefix or suffix of `nums` is **guaranteed** to fit in a **32-bit** integer.

You must write an algorithm that runs in `O(n)` time and without using the division operation.

Example 1:

```
Input: nums = [1,2,3,4]
Output: [24,12,8,6]
```

Example 2:

```
Input: nums = [-1,1,0,-3,3]
Output: [0,0,9,0,0]
```

Constraints:

- `2 <= nums.length <= 10<sup>5</sup>`
- `-30 <= nums[i] <= 30`
- The product of any prefix or suffix of `nums` is **guaranteed** to fit in a **32-bit** integer.

Follow up: Can you solve the problem in `O(1)` extra space complexity? (The output array **does not** count as extra space for space complexity analysis.)

```python
class Solution:
    def productExceptSelf(self, nums: List[int]) -> List[int]:
        ans = [0]*len(nums)
        productR = [0]*len(nums)
        productL = [0]*len(nums)
        prod = 1
        for i in range(len(nums)):
            prod = prod*nums[i]
            productR[i] = prod
        prod = 1
        for i in range(len(nums)-1,-1,-1):
            prod = prod*nums[i]
            productL[i] = prod
        for i in range(1,len(nums)-1):
            ans[i] = productR[i-1]*productL[i+1]
        ans[0] = productL[1]
        ans[-1] = productR[-2]
        return ans
```

FOLLOW-UP:

```
class Solution:
    def productExceptSelf(self, nums: List[int]) -> List[int]:
        # ans = [0]*len(nums)
        # productR = [0]*len(nums)
        productL = [0]*len(nums)
        prod = 1
        # for i in range(len(nums)):
        #     prod = prod*nums[i]
        #     productR[i] = prod
        for i in range(len(nums)-1,-1,-1):
            prod = prod*nums[i]
            productL[i] = prod
        prod = 1
        for i in range(0,len(nums)-1):
            productL[i] = prod*productL[i+1]
            prod = prod*nums[i]
        productL[-1] = prod
        return productL
```