

# Product of Array Except Self

📌 Difficulty	Medium
📌 Category	Arrays
🔗 Question	<a href="https://leetcode.com/problems/product-of-array-except-self/">https://leetcode.com/problems/product-of-array-except-self/</a>
🔗 Solution	<a href="https://youtu.be/bNvIQI2wAjk">https://youtu.be/bNvIQI2wAjk</a>
🌟 Status	Done

## Question

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

### 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]
```

## Idea



One - pass → prefix and postfix and multiplying the result

## Solution

```
class Solution:
    def productExceptSelf(self, nums: List[int]) -> List[int]:
        # Initialize a result list with all elements set to 1
        res = [1] * (len(nums))

        # Initialize a prefix variable to store the product of elements to the left of the current element
        prefix = 1
        for i in range(len(nums)):
            # Store the product of elements to the left in the result array
            res[i] = prefix
            # Update the prefix product by multiplying it with the current element
            prefix *= nums[i]

        # Initialize a postfix variable to store the product of elements to the right of the current element
        postfix = 1
        for i in range(len(nums) - 1, -1, -1):
            # Update the result array with the product of elements to the right
            res[i] *= postfix
            # Update the postfix product by multiplying it with the current element
            postfix *= nums[i]

        return res
```

## Explanation

1. We define a class `Solution` with a method `productExceptSelf` that takes a list of integers called `nums` as input and returns a list of integers.
2. We initialize a result list `res` with the same length as `nums`. Initially, all elements in `res` are set to 1 because, for any element in `nums`, the product of all elements except itself is 1.
3. We use a `prefix` variable to keep track of the product of elements to the left of the current element. We start with `prefix` set to 1.
4. The first loop iterates through `nums` from left to right. For each element at index `i`, we store the current `prefix` value in the `res` array at index `i` and then update `prefix` by multiplying it with the current element `nums[i]`. This loop calculates the product of elements to the left of each element.
5. We initialize a `postfix` variable to keep track of the product of elements to the right of the current element. We start with `postfix` set to 1.

- The second loop iterates through `nums` from right to left (in reverse order). For each element at index `i`, we update the corresponding element in the `res` array by multiplying it with the current `postfix` value, effectively calculating the product of elements to the right of each element. Then, we update `postfix` by multiplying it with the current element `nums[i]`.
- After both loops have executed, the `res` array will contain the product of all elements in `nums` except the element at the same index. Thus, it represents the desired result.

