Product of Array Except Self

2nd Review	@January 13, 2026
Date Solved	@January 12, 2026
Difficulty	Medium
# Review	1
Status	Solved
Topic/Pattern	Array and String

Link → https://neetcode.io/problems/products-of-array-discluding-self/question?list=neetcode150

Problem

- You are given an integer array `nums`.
- Return an array `answer` such that:

  answer[i] = product ofall elements of numsexcept nums[i]d

- You **must not use division**.
- The solution must run in **O(n)** time.

Example

Input	Output	Reason
`[1,2,3,4]`	`[24,12,8,6]`	Each index multiplies all other elements
`[1,2,4,6]`	`[48,24,12,8]`	Product of left × product of right
`[0,1,2,3]`	`[6,0,0,0]`	Only index with zero gets non-zero product

Approaches

1. Brute Force — Multiply Everything Except Index

```
defproductExceptSelf(nums):
    n =len(nums)
    result = []

for iinrange(n):
        prod =1
for jinrange(n):
if i != j:
                prod *= nums[j]
        result.append(prod)

return result
```

- **Time:** O(n²) — nested loops
- **Space:** O(n) — output array
- **Notes:** Simple logic but **too slow** for large inputs

---

2. Using Division (❌ Not Allowed)

```
defproductExceptSelf(nums):
    total =1
    zero_count = nums.count(0)

for numin nums:
if num !=0:
            total *= num

    result = []
for numin nums:
if zero_count >1:
            result.append(0)
elif zero_count ==1:
            result.append(totalif num ==0else0)
else:
            result.append(total // num)
```

```
return result
```

- **Time:** O(n)
- **Space:** O(n)
- **Notes:**

  ❌ Uses division

  ❌ Invalid per problem constraints

  ❌ Zero handling becomes messy

---

3. Optimal — Prefix & Suffix Products (Recommended)

```
defproductExceptSelf(nums):
    n =len(nums)
    answer = [1] * n

# Left products
    left =1
for iinrange(n):
        answer[i] = left
        left *= nums[i]

# Right products
    right =1
for iinrange(n -1, -1, -1):
        answer[i] *= right
        right *= nums[i]

return answer
```

- **Time: O(n)** — two linear passes
- **Space: O(1)** extra space (output array excluded)
- **Notes:**

  ✅ No division

  ✅ Handles zeros naturally

✅ Interview-preferred solution

How the Optimal Approach Works (High Level)

- First pass stores the **product of all elements to the left** of each index.
- Second pass multiplies the **product of all elements to the right**.
- Final value at each index = `left_product × right_product`.

Summary

Approach	Time	Space	Notes
Brute Force	$O(n^2)$	O(n)	Too slow
Division	O(n)	O(n)	❌ Not allowed
Prefix + Suffix	**O(n)**	**O(1)**	✅ Optimal & expected

Edge Cases

Input	Output	Reason
`[]`	`[]`	Empty array
`[5]`	`[1]`	No other elements
`[0,0]`	`[0,0]`	More than one zero
`[1,0,3,4]`	`[0,12,0,0]`	Only zero index has non-zero product

Tip

- When division is **not allowed**, think:
  - **Prefix products**
  - **Suffix products**
- This pattern avoids recomputation and handles zeros cleanly.
- Very common in **FAANG interviews**.