LeetCode 238 - Product of Array Except Self (Full Explanation)

LeetCode 238 - Product of Array Except Self

Problem Understanding

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].

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

Example:

Input: nums = [1,2,3,4]

Output: [24,12,8,6]

---

1. Brute Force Approach (O(n^2))

Idea:

For each element nums[i], multiply all other elements except nums[i].

Dry Run:

nums = [1,2,3,4]

i | nums[i] | Product of others | Result

0 | 1 | 2x3x4 = 24 | 24

1 | 2 | 1x3x4 = 12 | 12

2 | 3 | 1x2x4 = 8 | 8

3 | 4 | 1x2x3 = 6 | 6

Output = [24,12,8,6]

Python Code:

```
def productExceptSelf(nums):
```

LeetCode 238 - Product of Array Except Self (Full Explanation)

```
    n = len(nums)
    ans = []
    for i in range(n):
        prod = 1
        for j in range(n):
            if i != j:
                prod *= nums[j]
        ans.append(prod)
    return ans
```

C++ Code:

```
vector<int> productExceptSelf(vector<int>& nums) {
    int n = nums.size();
    vector<int> ans(n);
    for(int i=0;i<n;i++){
        int prod = 1;
        for(int j=0;j<n;j++){
            if(i!=j) prod *= nums[j];
        }
        ans[i] = prod;
    }
    return ans;
}
```

Java Code:

```
class Solution {
    public int[] productExceptSelf(int[] nums) {
        int n = nums.length;
        int[] ans = new int[n];
        for (int i = 0; i < n; i++) {
```

LeetCode 238 - Product of Array Except Self (Full Explanation)

```
            int prod = 1;
            for (int j = 0; j < n; j++) {
                if (i != j) prod *= nums[j];
            }
            ans[i] = prod;
        }
        return ans;
    }
}
```

---

2. Optimal Approach - Prefix and Suffix Product (O(n))

Idea:

We calculate prefix[i] = product of elements to the left of i

and suffix[i] = product of elements to the right of i.

Then answer[i] = prefix[i] * suffix[i]

Dry Run Example: nums = [1,2,3,4]

Prefix pass:

prefix = [1, 1, 2, 6]

Suffix pass updates final result:

answer = [24,12,8,6]

Python Code:

```
def productExceptSelf(nums):
    n = len(nums)
    ans = [1] * n
    prefix = 1
    for i in range(n):
        ans[i] = prefix
```

LeetCode 238 - Product of Array Except Self (Full Explanation)

```
        prefix *= nums[i]
    suffix = 1
    for i in range(n-1, -1, -1):
        ans[i] *= suffix
        suffix *= nums[i]
    return ans
```

C++ Code:

```
vector<int> productExceptSelf(vector<int>& nums) {
    int n = nums.size();
    vector<int> ans(n, 1);
    int prefix = 1;
    for (int i = 0; i < n; i++) {
        ans[i] = prefix;
        prefix *= nums[i];
    }
    int suffix = 1;
    for (int i = n - 1; i >= 0; i--) {
        ans[i] *= suffix;
        suffix *= nums[i];
    }
    return ans;
}
```

Java Code:

```
class Solution {
    public int[] productExceptSelf(int[] nums) {
        int n = nums.length;
        int[] ans = new int[n];
        int prefix = 1;
```

LeetCode 238 - Product of Array Except Self (Full Explanation)

```
        for (int i = 0; i < n; i++) {
            ans[i] = prefix;
            prefix *= nums[i];
        }
        int suffix = 1;
        for (int i = n - 1; i >= 0; i--) {
            ans[i] *= suffix;
            suffix *= nums[i];
        }
        return ans;
    }
}
```

Time and Space Complexity

Approach	Time	Space	Notes
Brute Force	O(n^2)	O(1)	Simple but slow
Prefix-Suffix	O(n)	O(1)	Efficient and elegant