# 238. Product of Array Except Self

Given an integer array nums, return an array answer s.t. answer[i] is equal to the product of all elements in nums except nums[i]

Constraint: O(n) Time O(1) Space and no division

Ex. 1]
Input: [1,2,3,4]
Output: [24, 12, 8, 6]

Ex. 2]
Input: [7,0,2]
Output: [0, 14, 0]

Thoughts:

Mixture of prefix and suffix sums. ✓

Prefix: [1, 2, 6, 24]
Suffix: [24, 24, 12, 4]

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 1 | 2 | 6 | 24 |
| 24 | 24 | 12 | 4 |

1.) Using division

- Calculate the product, excluding 0s
- Insert into answer array 0 if 0 exists or prod / value
- Return answer

```
std::vector<int> productExceptSelf(std::vector<int> nums) {
    const auto count = std::count(nums.cbegin(), nums.cend(), {});
    if (count > 1) return std::vector<int>(nums.size());
    decltype(nums)::value_type product(1);
    for (const auto num : nums) {
        if (num == {}) {
            continue;
        }
```

```
            product *= num;
        }

        for (auto& num : nums) {
            if (count != 0) {
                num = num == 0 ? product : 0;
            }
            else {
                num = product / num;
            }
        }

        return nums;
    }
```

The way to do it w/ out division.

| | | | | | |
|---|---|---|---|---|---|
| Input: | 1 | 2 | 3 | 4 | 5 |
| Prefix: | 1 | 2 | 6 | 24 | 120 |
| Suffix: | 120 | 120 | 60 | 20 | 5 |

| | | | | | |
|---|---|---|---|---|---|
| Input: | 1 | 2 | 3 | 4 | 5 |
| Prefix: | 1 | 2 | 6 | 24 | 120 |
| Suffix: | 120 | 120 | 60 | 20 | 5 |

$i = 0$

$prefix[i-1] * suffix[i+1] = 120$

invalid!

| | | | | | |
|---|---|---|---|---|---|
| Input: | 1 | 2 | 3 | 4 | 5 |
| Prefix: | 1 | 2 | 6 | 24 | 120 |
| Suffix: | 120 | 120 | 60 | 20 | 5 |

$i = 1$

$answer[i] = prefix[i-1] * suffix[i+1]$

$= 60$

| | | | | | | |
|---|---|---|---|---|---|---|
| Input: | 1 | 2 | 3 | 4 | 5 | i = 2 |
| Prefix: | 1 | 2 | 6 | 24 | 120 | answer[i] = prefix[i-1] * suffix[i+1] |
| Suffix: | 120 | 120 | 60 | 20 | 5 | = 40 |

```
std::vector<int> productExceptSelf(std::vector<int> nums) {
    auto suffix{nums};
    auto prefix{nums};
    for (std::size_t i{1}; i < prefix.size(); ++i) {
        prefix[i] *= prefix[i-1];
    }

    for (std::size_t i = suffix.size()-2; i >= 0; --i) {
        suffix[i] *= suffix[i+1];
    }

    for (std::size_t i=0; i < prefix.size(); ++i) {
        int ans{1};
        if (i != 0) {
            ans *= prefix[i-1];
        }
        if (i != suffix.size()-1) {
            ans *= suffix[i+1];
        }
        suffix[i] = ans;
    }
    return suffix;
}
```