Daily Coding Problem #2

Problem

This problem was asked by Uber.

Given an array of integers, return a new array such that each element at index `i` of the new array is the product of all the numbers in the original array except the one at `i`.

For example, if our input was `[1, 2, 3, 4, 5]`, the expected output would be `[120, 60, 40, 30, 24]`. If our input was `[3, 2, 1]`, the expected output would be `[2, 3, 6]`.

Follow-up: what if you can't use division?

Solution

This problem would be easy with division: an optimal solution could just find the product of all numbers in the array and

then divide by each of the numbers.

Without division, another approach would be to first see that the i^{th} element simply needs the product of numbers before i and the product of numbers after i. Then we could multiply those two numbers to get our desired product.

In order to find the product of numbers before i, we can generate a list of prefix products. Specifically, the i^{th} element in the list would be a product of all numbers including i. Similarly, we would generate the list of suffix products.

```
def products(nums):
    # Generate prefix products
    prefix_products = []
    for num in nums:
        if prefix_products:
            prefix_products.append(prefix_products[-1] * num)
        else:
            prefix_products.append(num)

    # Generate suffix products
    suffix_products = []
    for num in reversed(nums):
        if suffix_products:
            suffix_products.append(suffix_products[-1] * num)
        else:
            suffix_products.append(num)
    suffix_products = list(reversed(suffix_products))

    # Generate result
    result = []
    for i in range(len(nums)):
        if i == 0:
            result.append(suffix_products[i + 1])
```

```
        elif i == len(nums) - 1:
            result.append(prefix_products[i - 1])
        else:
            result.append(prefix_products[i - 1] * suffix_products[i + 1])
    return result
```

This runs in O(N) time and space, since iterating over the input arrays takes O(N) time and creating the prefix and suffix arrays take up O(N) space.