LeetCode 238 [Product of Array Except Self]

Given an 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 ~~prefix~~ 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 operator.

i: [1, 2, 3, 4]
o: [24, 12, 8, 6]

2*3*4 1*3*4

It's not possible to 1*2*3*4 = 24

24/1 = 24, 24/2 = 12, 24/3 = 8, ...

Brute Force

nums: [1, 2, 3, 4]

Itarate the array for every single character. Its $\overset{time}{O(n^2)}$

Better approach

nums: [1, 2, 3, 4] → 1

For left and righ edge we consider 1 because anything multiply by 1 is the number itself.

pre:	1	1	2	6
post	24	12	4	1

Ans: 24, 12, 8, 6

Time: O(n)
Space: O(n)

Optimal approach

nums: [1, 2, 3, 4]

pre: 1 2 6
post: 1 4 12 24

Ans: | 1 | 1 | 2 | 6 |
24 12 8 6

4×1 = 4
3×4 = 12
2×12 = 24

Time: O(2n)
Space: O(1)

Java Solution

```
public int[] productExceptSelf(int[] nums) {
    int[] result = new int[nums.length];

    Arrays.fill(result, 1);

    int pre = 1, post = 1

    For(int i = 0; i < nums.length; i++){
        ~~result[i] = result[i] * post;~~
        result[i] = pre;
        pre = nums[i] * pre;
    }

    For(int i = nums.length - 1; i >= 0; i--) {
        result[i] = result[i] * post;
        post = post * nums[i];
    }

    return result;
}
```

Kotlin solution

```
Fun productExceptSelf(nums: IntArray): IntArray {
    val result = IntArray(nums.size) {1}

    var pre = 1
    var post = 1

    For (i in nums.indices){
        result[i] = pre
        pre = nums[i] * pre
    }
    For (i in nums.size -1 downTo 0) {
        result[i] = result[i] * post;
        post = post * nums[i]
    }

    return result
}
```