Product of Array Except itself

Given nums[n], return an array answer, such that answer[i] is equal to the product of all elements of nums except itself (nums[i])

Ex:- nums[] = (1, 2, 3, 4)

⇓

answer[] = (24, 12, 8, 6)

* We have to multiply every index value other than current idx for each index

Algorithm

① Initialize a "Suffix" variable → Suffix = 1

② for (n-1 ⟶ 1)

Do, prefixproduct (i) = prefixproduct [i-1] * Suffix

Suffix = suffix * nums (i)

What's going here?

let's breakdown

① At starting, create a prefixproduct and suffix product of our array

arr[] : (1, 2, 3, 4)

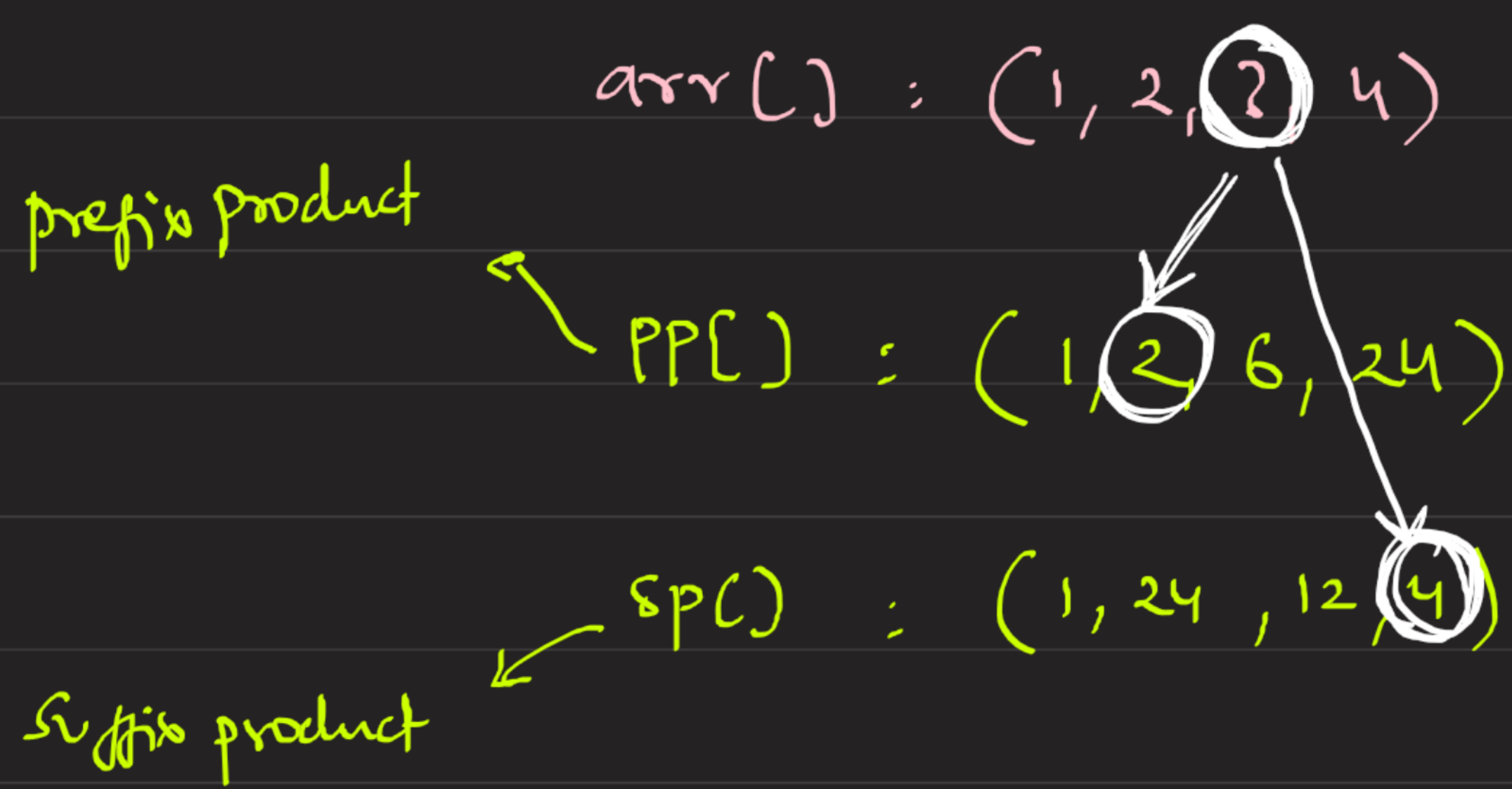prefix product

PP[] : (1, 2, 6, 24)

SP[] : (1, 24, 12, 4)

suffix product

$arr[] : (1, 2, 3, 4)$

prefix product

$PP[] : (1, 2, 6, 24)$

$SP() : (1, 24, 12, 4)$

Suffix product

At every index, between (1 to n-2)

See $\rightarrow$ $PP[i-1] \times SP[i+1]$

$= 2 \times 4 = 8$

Check our answer array, for index we having 3 in input array we will have 8 as answer

for $(i==0) \longrightarrow SP(i+1)$

for $(i==n-1) \longleftrightarrow PP(i-1)$

Creating 2 arrays (Prefix Product and Suffix Product) 👀

Bad Bro.... Optimize it!!!

Algorithm $\longrightarrow$ Now understand this,

① Initialize a "Suffix" variable $\longrightarrow$ (Suffix = 1)

② for $(n-1 \longrightarrow 1)$

Do, prefixproduct (i) = prefixproduct $[i-1]$ * Suffix

Suffix = Suffix * nums (i)

We don't really need 2 arrays, we can do this with one prefixproduct and a variable

(which stores required Suffix product)

* Create a Suffix Variable
        ↳ initializes to 1

* On every iteration update Suffix → By Suffix * arr[i]
        ↓
    ⟨ goes from Back (n-2 to 1) ⟩

$SP[i+1]$
previously it is↗

* Make ans for every index as
        ↳
    $PP(i) = PP[i-1] * \boxed{Suffix}$

* At last pp[] is Our answer

Dry run :

arr[] : (1, 2, 3, 4)

Prefix product [] : (1, 2, 6, 24) , Suffix = 1

i	Prefixproduct [i]	Suffix
3	PP(i-1) = 6 Suffix = 1 $\boxed{PP(i-1) * Suffix = 6}$	$\boxed{Suffix = Suffix * arr[i]} = 1 * 4 = 4$
2	2 * 4	4 * arr[2] = 12
1	1 * 12	12 * arr(1) = 24
0	24	

at (i==0) → prefixproduct [i] = Suffix