152. Maximum Product Subarray

Medium 17.1K 524

Companies

Given an integer array `nums`, find a subarray that has the largest product, and return *the product*.

The test cases are generated so that the answer will fit in a **32-bit** integer.

Example 1:

```
Input: nums = [2,3,-2,4]
Output: 6
Explanation: [2,3] has the largest product 6.
```

Example 2:

```
Input: nums = [-2,0,-1]
Output: 0
Explanation: The result cannot be 2, because [-2,-1] is not a subarray.
```

Constraints:

- `1 <= nums.length <= 2 * 10^4`
- `-10 <= nums[i] <= 10`
- The product of any prefix or suffix of `nums` is **guaranteed** to fit in a **32-bit** integer.

Accepted **1.1M** | Submissions **3.1M** | Acceptance Rate **34.9%**

Bruteforce:

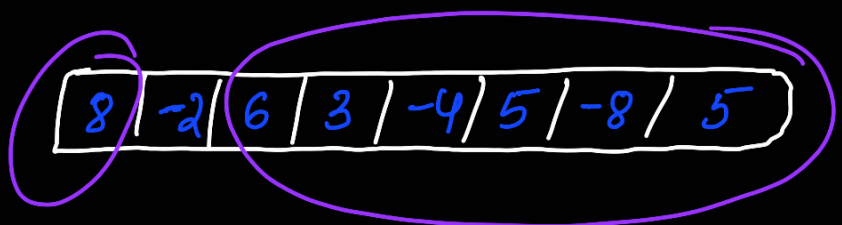
$T(n) : O(n^2)$

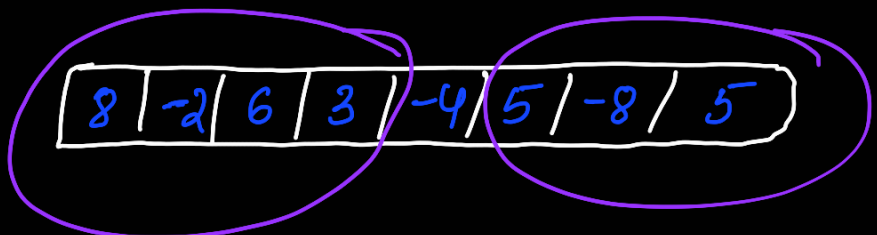
$S(n) : O(n)$

Approach 1: Using prefix and Suffix

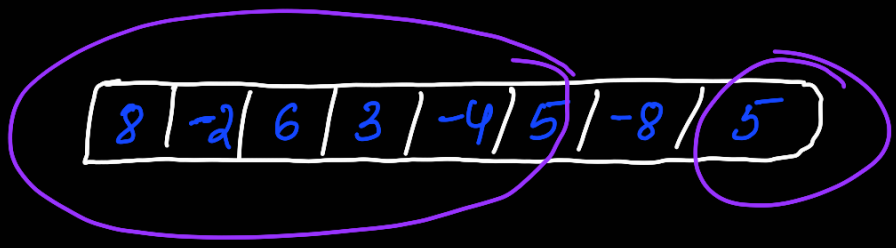
The key observations on this question are

- If all elements are positive then maxProduct Subarray will be whole array.
- If there are even no. of negative elements without any zeroes then also the whole array will be the maxProduct sub array.
- If the array contains odd no. of negative elements then either prefix or Suffix of one of the negative element could be the max product Subarray

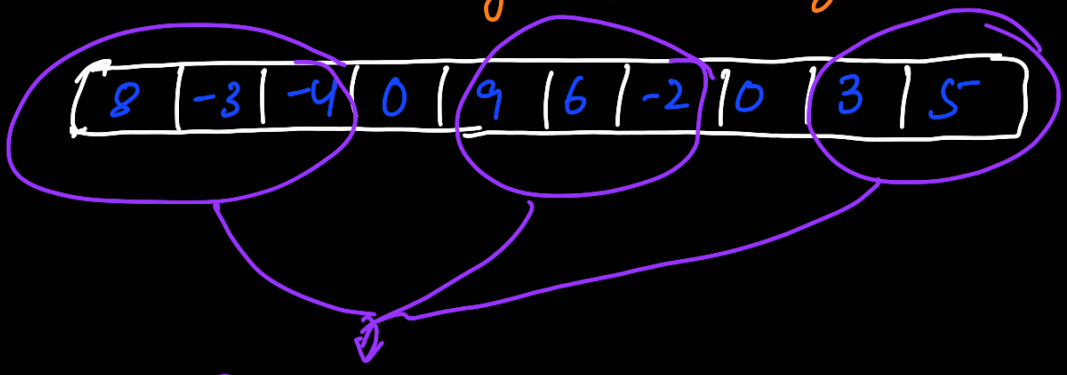




here as we can see both prefix & suffix are negative but we don't need to worry about this case becoz " for an array with odd no of negative elements the max product will always contain even no. of negative elements." So this case would never be the answer.



→ If the array contains zeroes



The max would be either of these 3 subarrays or it could be zero if all these 3 subarray product comes out to be negative.

With all these observations we could come up with the solution.

```
prefix = 1
Suffix = 1
ans = -∞
for( i:0 to n-1)
{
    if( prefix becomes zero)  prefix = 1
    if( Suffix becomes zero)  Suffix = 1
    To handle when array has zeroes
    prefix = prefix * nums[i]
    Suffix = Suffix * nums[n-i-1]

    ans = max( ans, max( pre, suf))
}
```