Maximum Product Subarray

solved by	Senan
Platform	GeeksForGeeks
difficulty	Medium
tags	Prefix Sum
language	C++
solved on	@25/11/2024
link	https://www.geeksforgeeks.org/problems/maximum-product-subarray3604/1
Completion	☑

Intuition

The problem requires finding the contiguous subarray within an array that has the largest product. A key observation is that the product can change drastically due to the presence of negative numbers and zeros:

- **Negative numbers:** Multiplying two negatives gives a positive product, so tracking the smallest negative product can help.
- **Zeros:** They break the product continuity, resetting the product.

To efficiently solve the problem, the **two-pass approach** is utilized:

- **Left to right pass:** Accumulate the product while tracking the maximum.
- **Right to left pass:** Repeat the same process but starting from the end of the array. This helps handle cases where the maximum product is achieved by a subarray ending at the last element.

By considering both passes, we ensure that the product of all possible subarrays is evaluated.

Approach

1. Initialize `left` and `right` products to 1, and `maxi` (to store the maximum product) to `INT_MIN`.
2. Traverse the array twice:
   - **First Pass (Left to Right):** Multiply each element with `left`. If `left` becomes zero (due to a zero in the array), reset `left` to 1.
   - **Second Pass (Right to Left):** Multiply each element from the end of the array with `right`. Similarly, reset `right` to 1 when encountering a zero.
3. In each pass, update `maxi` with the maximum of `maxi`, `left`, and `right`.
4. Return `maxi` as the maximum product.

Complexity

Time Complexity:

- **O(n):** We perform a single traversal of the array from both directions.

Space Complexity:

- **O(1):** Only a few variables are used for tracking the product and maximum.

Code

```
class Solution {
  public:
    int maxProduct(vector<int> &arr) {
        int n = arr.size();
        int left = 1;
        int right = 1;
        int maxi = INT_MIN;

        for (int i = 0; i < n; i++) {
            left *= arr[i];
            right *= arr[n - i - 1];
            maxi = max({maxi, left, right});

            if (left == 0) left = 1;
            if (right == 0) right = 1;
        }

        return maxi;
    }
};
```