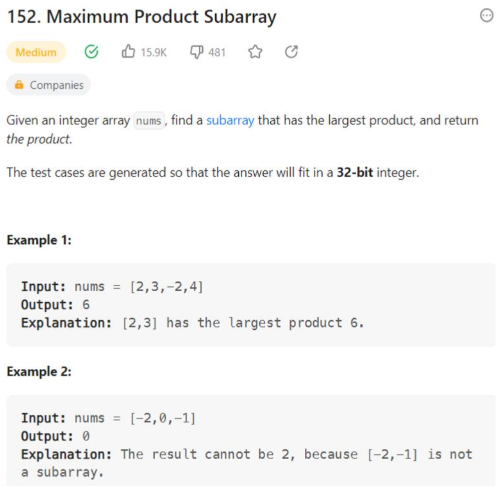
42. Maximum Product Subarray

Wednesday, May 24, 2023 11:06 AM



152. Maximum Product Subarray

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.
```

Maximum Product Subarray

↳ contigous part of array.

arr [] = [2, 3, -2, 4]   [2, -2] X (Subsequences)

ans = 6

1) Brute Force

- Generate all sub array.

```
maxi = INT_MIN
for (i = 0 ; i < n ; i + +)
{
    for (j = i ; j < n ; j + +)
    {   product = 1
        for ( k = i ⟶ j )     // this will give subarray
        {
            product = product * arr [k] ;
            }
            maxi = max(maxi, product)
        }
    }
```

TC ⇒ $O(n^3)$ S.C ⇒ $O(1)$

2) Better Solution :

```
for(i=0; i<n; i++)
{ product = 1
  for(j=i; j<n; j++)
  {   prod = prod + arr[j];
      maxi = max(maxi, prod)
  }
}
```

TC ⇒ $O(n^2)$ S.C ⇒ $O(1)$

3) Optimal Approach

Observation → use this always. in interview

Kadanes' Algo

arr[] = [2, 3, -2, 4]

ans = 6

1. +ve ⇒ multiple everyone

2. even -ve, rest +ve ⇒ multiple everyone

3. odd -ve rest +ve ⇒ will give -ve.

2. even -ve, rest +ve → multiple everyon

3. odd -ve, rest +ve ⇒ will give -ve.

we need max product, so you can do remove one -ve nos of odd nos of -ves will leave we with even nos of -ves.

4. if it has '0', multiplication will be '0'.

e.g. | -2 3 4 -1 | 0 | -2 3 1 4 | 0 | 4 6 7 |

i ll take this array and check

{ 2 3 -2 4 }

max = ~~INT_MIN~~ ~~2~~ 6

this is max {2 3}

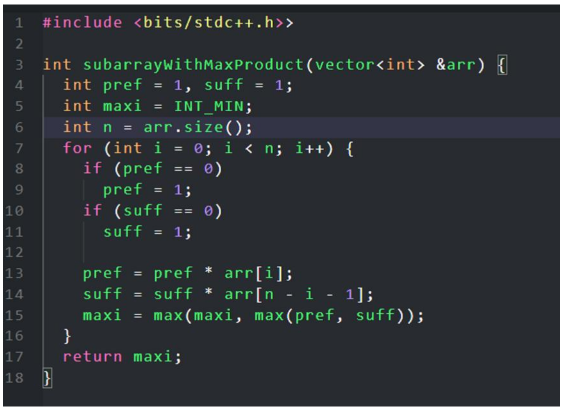
prex = ~~1~~ ~~2~~ ~~6~~ ~~-12~~ -48

suff = ~~1~~ ~~4~~ ~~-8~~ ~~-24~~ ~~-48~~

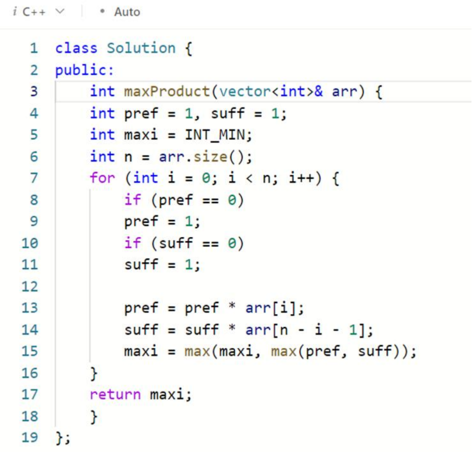
Now if there is '0' in the array and if you encounter '0' while multiplying then change the prefix to '1' again. (starting up new).

<u>Pseudo</u>

```
pref = 1 , suff = 1
for( i = 0 → n-1)
{
    if ( pre == 0) pre = 1      } //handling '0' in
    if (suff == 0) suff = 1     }  the array.

    pref = pref × arr[i]
    suff = suff × arr[n-i-1]; //as suff is from back.
    maxi = max(maxi, max(pref, suff))
}
return maxi;
```



```cpp
#include <bits/stdc++.h>>

int subarrayWithMaxProduct(vector<int> &arr) {
  int pref = 1, suff = 1;
  int maxi = INT_MIN;
  int n = arr.size();
  for (int i = 0; i < n; i++) {
    if (pref == 0)
      pref = 1;
    if (suff == 0)
      suff = 1;

    pref = pref * arr[i];
    suff = suff * arr[n - i - 1];
    maxi = max(maxi, max(pref, suff));
  }
  return maxi;
}
```



```cpp
class Solution {
public:
    int maxProduct(vector<int>& arr) {
    int pref = 1, suff = 1;
    int maxi = INT_MIN;
    int n = arr.size();
    for (int i = 0; i < n; i++) {
        if (pref == 0)
        pref = 1;
        if (suff == 0)
        suff = 1;

        pref = pref * arr[i];
        suff = suff * arr[n - i - 1];
        maxi = max(maxi, max(pref, suff));
    }
    return maxi;
    }
};
```

T.C => O(n)   S.C => O(1)