Maximum Product Subarray

Given an array that contains both positive and negative integers, find the product of the maximum product subarray. Expected Time complexity is O(n) and only O(1) extra space can be used.

Examples:

```
Input: arr[] = {6, -3, -10, 0, 2}
Output:   180  // The subarray is {6, -3, -10}

Input: arr[] = {-1, -3, -10, 0, 60}
Output:   60  // The subarray is {60}

Input: arr[] = {-2, -3, 0, -2, -40}
Output:   80  // The subarray is {-2, -40}
```

The following solution assumes that the given input array always has a positive ouput. The solution works for all cases mentioned above. It doesn't work for arrays like {0, 0, -20, 0}, {0, 0, 0}.. etc. The solution can be easily modified to handle this case.

It is similar to Largest Sum Contiguous Subarray problem. The only thing to note here is, maximum product can also be obtained by minimum (negative) product ending with the previous element multiplied by this element. For example, in array {12, 2, -3, -5, -6, -2}, when we are at element -2, the maximum product is multiplication of, minimum product ending with -6 and -2.

C/C++

```
// C program to find Maximum Product Subarray
#include <stdio.h>

// Utility functions to get minimum of two integers
int min (int x, int y) {return x < y? x : y; }

// Utility functions to get maximum of two integers
int max (int x, int y) {return x > y? x : y; }

/* Returns the product of max product subarray.
   Assumes that the given array always has a subarray
   with product more than 1 */
int maxSubarrayProduct(int arr[], int n)
```

```
{
    // max positive product ending at the current position
    int max_ending_here = 1;

    // min negative product ending at the current position
    int min_ending_here = 1;

    // Initialize overall max product
    int max_so_far = 1;

    /* Traverse through the array. Following values are
       maintained after the i'th iteration:
       max_ending_here is always 1 or some positive product
                       ending with arr[i]
       min_ending_here is always 1 or some negative product
                       ending with arr[i] */
    for (int i = 0; i < n; i++)
    {
        /* If this element is positive, update max_ending_here.
           Update min_ending_here only if min_ending_here is
           negative */
        if (arr[i] > 0)
        {
            max_ending_here = max_ending_here*arr[i];
            min_ending_here = min (min_ending_here * arr[i], 1);
        }

        /* If this element is 0, then the maximum product
           cannot end here, make both max_ending_here and
           min_ending_here 0
           Assumption: Output is alway greater than or equal
                       to 1. */
        else if (arr[i] == 0)
        {
            max_ending_here = 1;
            min_ending_here = 1;
        }

        /* If element is negative. This is tricky
           max_ending_here can either be 1 or positive.
           min_ending_here can either be 1 or negative.
           next min_ending_here will always be prev.
           max_ending_here * arr[i] next max_ending_here
           will be 1 if prev min_ending_here is 1, otherwise
           next max_ending_here will be prev min_ending_here *
           arr[i] */
        else
        {
            int temp = max_ending_here;
            max_ending_here = max (min_ending_here * arr[i], 1);
            min_ending_here = temp * arr[i];
        }

        // update max_so_far, if needed
        if (max_so_far <  max_ending_here)
          max_so_far  =  max_ending_here;
    }

    return max_so_far;
}

// Driver Program to test above function
int main()
{
    int arr[] = {1, -2, -3, 0, 7, -8, -2};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Maximum Sub array product is %d",
            maxSubarrayProduct(arr, n));
    return 0;
}
```

Run on IDE

Java

```
// Java program to find maximum product subarray
import java.io.*;

class ProductSubarray {

    // Utility functions to get minimum of two integers
    static int min (int x, int y) {return x < y? x : y; }

    // Utility functions to get maximum of two integers
    static int max (int x, int y) {return x > y? x : y; }

    /* Returns the product of max product subarray.
       Assumes that the given array always has a subarray
       with product more than 1 */
    static int maxSubarrayProduct(int arr[])
    {
        int n = arr.length;
        // max positive product ending at the current position
        int max_ending_here = 1;

        // min negative product ending at the current position
        int min_ending_here = 1;

        // Initialize overall max product
        int max_so_far = 1;

        /* Traverse through the array. Following
           values are maintained after the ith iteration:
           max_ending_here is always 1 or some positive product
                           ending with arr[i]
           min_ending_here is always 1 or some negative product
                           ending with arr[i] */
        for (int i = 0; i < n; i++)
        {
            /* If this element is positive, update max_ending_here.
                Update min_ending_here only if min_ending_here is
                negative */
            if (arr[i] > 0)
            {
                max_ending_here = max_ending_here*arr[i];
                min_ending_here = min (min_ending_here * arr[i], 1);
            }

            /* If this element is 0, then the maximum product cannot
               end here, make both max_ending_here and min_ending
              _here 0
               Assumption: Output is alway greater than or equal to 1. */
            else if (arr[i] == 0)
            {
                max_ending_here = 1;
                min_ending_here = 1;
            }
            /* If element is negative. This is tricky
               max_ending_here can either be 1 or positive.
               min_ending_here can either be 1 or negative.
               next min_ending_here will always be prev.
               max_ending_here * arr[i]
               next max_ending_here will be 1 if prev
               min_ending_here is 1, otherwise
               next max_ending_here will be
                           prev min_ending_here * arr[i] */
            else
            {
                int temp = max_ending_here;
                max_ending_here = max (min_ending_here * arr[i], 1);
                min_ending_here = temp * arr[i];
            }

            // update max_so_far, if needed
            if (max_so_far <  max_ending_here)
                max_so_far  =  max_ending_here;
        }

        return max_so_far;
    }

    public static void main (String[] args) {

        int arr[] = {1, -2, -3, 0, 7, -8, -2};
        System.out.println("Maximum Sub array product is "+
                           maxSubarrayProduct(arr));
    }
}/*This code is contributed by Devesh Agrawal*/
```

Run on IDE

Python

```
# Python program to find maximum product subarray

# Returns the product of max product subarray.
# Assumes that the given array always has a subarray
# with product more than 1
def maxsubarrayproduct(arr):

    n = len(arr)

    # max positive product ending at the current position
    max_ending_here = 1

    # min positive product ending at the current position
    min_ending_here = 1

    # Initialize maximum so far
    max_so_far = 1

    # Traverse throughout the array. Following values
    # are maintained after the ith iteration:
    # max_ending_here is always 1 or some positive product
    # ending with arr[i]
    # min_ending_here is always 1 or some negative product
    # ending with arr[i]
    for i in range(0,n):
        # If this element is positive, update max_ending_here.
        # Update min_ending_here only if min_ending_here is
        # negative
        if arr[i] > 0:
            max_ending_here = max_ending_here*arr[i]
            min_ending_here = min (min_ending_here * arr[i], 1)

        # If this element is 0, then the maximum product cannot
        # end here, make both max_ending_here and min_ending_here 0
        # Assumption: Output is alway greater than or equal to 1.
        elif arr[i] == 0:
            max_ending_here = 1
            min_ending_here = 1

        # If element is negative. This is tricky
        # max_ending_here can either be 1 or positive.
        # min_ending_here can either be 1 or negative.
        # next min_ending_here will always be prev.
        # max_ending_here * arr[i]
        # next max_ending_here will be 1 if prev
        # min_ending_here is 1, otherwise
        # next max_ending_here will be prev min_ending_here * arr[i]
        else:
            temp = max_ending_here
            max_ending_here = max (min_ending_here * arr[i], 1)
            min_ending_here = temp * arr[i]
        if (max_so_far <  max_ending_here):
            max_so_far  =  max_ending_here
    return max_so_far

# Driver function to test above function
arr = [1, -2, -3, 0, 7, -8, -2]
print "Maximum product subarray is",maxsubarrayproduct(arr)

# This code is contributed by Devesh Agrawal
```

Run on IDE

Output:

```
Maximum Sub array product is 112
```

Time Complexity: O(n)

Auxiliary Space: O(1)

This article is compiled by **Dheeraj Jain** and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

182 Comments Category: Arrays

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Count triplets with sum smaller than a given value
- Find zeroes to be flipped so that number of consecutive 1's is maximized
- Reorder an array according to given indexes
- Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed
- Find maximum average subarray of k length

(Login to Rate and Mark)

2.4 Average Difficulty : **2.4/5.0**
Based on **7** vote(s)

Add to TODO List

Mark as DONE

Like Share 13 people like this. Be the first of your friends.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

182 Comments GeeksforGeeks 1 Login

Recommend 4 Share Sort by Newest

Join the discussion

Crackhacker • 15 days ago

Cleaner and easier: http://code.geeksforgeeks.org/...

• Reply • Share ›

suraj • 16 days ago

this will work for every array

#include<stdio.h>

int maxSubarrayProduct(int a[], int n)

{

int max=a[0];

int prod,i,j,x=0;

for(i=1;i<n;i++) {="" if(a[i]="=0)" continue;="" if(a[i-1]="=0&amp;&amp;abs(a[i])">abs(max))

{

max=abs(a[i]);

x=i;

see more

• Reply • Share ›

Rashmi Mishra • 18 days ago

Simple and efficient code where we dont need to check whether the array element is positive or negative and will work for all cases.
Time complexity - O(n)
Space complexity - O(1)

http://code.geeksforgeeks.org/...

1 • Reply • Share ›

Raj Coding World Rashmi Mishra • 7 days ago

This seems super easy, did you check for all cases?

• Reply • Share ›

Rashmi Mishra Raj Coding World • 6 days ago

Yes , it works for all cases . Tried and tested code .

^ | ˅ • Reply • Share ›

Shambhavi Shinde • 18 days ago

Simpler approach with O(n) time complexity and O(1) space. maxm is maximum product till ith element, and present is the product of the array traversed so far.
int maxSubarrayProduct(int arr[], int n)
{
int maxm=INT_MIN,present=1,i;
for(i=0;i<n;i++) {="" if(arr[i]="=0)" {="" present="1;" continue;="" }="" maxm="max(maxm,present*arr[i]);" present*="arr[i];" }="" if(present="=1||present&lt;0)" return="" 1;="" return="" maxm;="" }="">

^ | ˅ • Reply • Share ›

StrictMath • a month ago

The use of 1 in the solution is counter-intuitive. Further, special handling is required for arrays which do not have a positive maximum. This can be rectified as follows:

```
for(int i=0; i < N; i++)
{
     if ( arr[i] < 0 )
     {
            int temp = min_ending_here;
            min_ending_here = min ( max_ending_here * arr[i], arr[i] );
            max_ending_here = max ( temp * arr[i], arr[i] );
     }
     else
     {
            min_ending_here = min ( min_ending_here * arr[i], arr[i] );
            max_ending_here = max ( max_ending_here * arr[i], arr[i] );
     }

     ret = max (ret, max_ending_here);
}
```

^ | ˅ • Reply • Share ›

Aman ➦ StrictMath • 24 days ago

But what happens when arr[i] == 0 ? min_ending_here becomes Zero, and then in further iterations it will always cause product to be zero ?

^ | ˅ • Reply • Share ›

Rahul Sarkar • a month ago

simple and easy approach-

int maxsubarray(int a[],int n){

int cur,max;
cur=max=1;
for(int i=0;i<n;i++){ cur="cur*a[i];" if(cur="=0)" cur="1;" if(cur="">max)
max=cur;
}
return max;
}

• Reply • Share ›

Unique Rahul Sarkar • 4 days ago

Wrong answer for following

input={2,0,-4,2,3}

• Reply • Share ›

Rahul Sarkar Unique • 3 days ago

thx bro

• Reply • Share ›

sandeepjindal • a month ago

it will give wrong answer for -4 0 -5 0

1 • Reply • Share ›

Aman sandeepjindal • 24 days ago

It will give 1.

• Reply • Share ›

Anurag Prabhakar • 2 months ago

Simpler implementation:
int ans = nums[0], currmax=nums[0], currmin =nums[0], prevmax =nums[0] , prevmin = nums[0];
for(int i=1;i<nums.size();i++) {=""
currmax="MAX(prevmax*nums[i],prevmin*nums[i],nums[i]);"
currmin="MIN(prevmax*nums[i],prevmin*nums[i],nums[i]);" ans="max(ans," currmax);=""
prevmax="currmax;" prevmin="currmin;" }="">

• Reply • Share ›

Mallikarjun Patil • 3 months ago

Java implementation:

public class MAximumProductSubarray {

public static void main(String[] args) {

MAximumProductSubarray mAximumProductSubarray = new MAximumProductSubarray();

// int a[]={6, -3, -10, 0, 2};

// int a[]={-1, -3, -10, 0, 60};

// int a[]={-2, -3, 0, -2, -40};

int a[]={1, -2, -3, 0, 7, -8, -2};

mAximumProductSubarray.printMaxSum(a);

}

private void printMaxSum(int[] a) {

int curMax=1;

int finalMax=1;

for(int i=0;i<a.length;i++){ curmax="curMax*a[i];" if(curmax="=0){" curmax="1;" }="" if(finalmax<curmax){="" finalmax="curMax;" }="" }="" system.out.println("value="" of="" maximum="" contiguous="" subarray="" is="" "+finalmax);="" }="" }="">

• Reply • Share ›

Coder12 • 3 months ago

hello! can anyone give some test cases which can prove my solution wrong ...i think its best n short solution http://ideone.com/bsmMfl

• Reply • Share ›

Monika Daryani Coder12 • 3 months ago

{-1, -3, -10, 0}. answer is 30, but your solution gives 3

• Reply • Share ›

Vijay • 3 months ago

Who will provide the sub array ? I am not understanding the entire question here.

• Reply • Share ›

Aman Vijay • 24 days ago

You don't need to return the subarray. Just need the maximum possible product of a subarray.

• Reply • Share ›

Preeti Rani • 3 months ago

This code handles the negative output also.

```
int maxProduct(vector<int>& nums) {
int maxlocal = nums[0] , minlocal = nums[0] , global = nums[0];

for(int i = 1; i< nums.size() ; i++)
{
int temp = maxlocal;
maxlocal = max(max(temp*nums[i] , nums[i]), minlocal*nums[i]);
minlocal = min(min(temp*nums[i] , nums[i]), minlocal*nums[i]);
global = max(global,maxlocal);
}
return global;
}
```

• Reply • Share ›

Fattepur Mahesh • 3 months ago

http://code.geeksforgeeks.org/...

• Reply • Share ›

Anirudh Agrawal Fattepur Mahesh • 3 months ago

5
6 -3 10 0 2

Dude your code fails for this input ,

• Reply • Share ›

Teja • 4 months ago

Keep track of products from left, products from right. Select the max product.
When zero is encountered, the product_so_far will be 1.

• Reply • Share ›

Avat

This comment was deleted.

Tej Guest • 3 months ago

For the array, -2, -3, 0, -2, -40
1) Product so far from left is -2 6 1 -2 80
2) Product so far from right 6 -3 1 80 -40
Get the max of all i.e 80.

• Reply • Share ›

Parminder Bharti • 4 months ago

Java Code... Try this

https://ideone.com/w50uMu

• Reply • Share ›

seema • 4 months ago

@geeksforgeeks
plz check your solution for following inputs, your program is not giving correct answer.
1. {0}
2. {-2}
3.{-2,0,-1}

1 • Reply • Share ›

Subhabrata Das seema • 3 months ago

To quote:
"The following solution assumes that the given input array always has a positive ouput. The solution works for all cases mentioned above. It doesn’t work for arrays like {0, 0, -20, 0}, {0, 0, 0}.. etc. The solution can be easily modified to handle this case."

2 • Reply • Share ›

priti • 4 months ago

#include <stdio.h>
/* Returns the product of max product subarray. */
int maxSubarrayProduct(int arr[], int n)
{
int i ,mul=1 ,max = -999;
for(i=0; i<n ;i++)="" {="" mul="mul*" arr[i];="" if(arr[i]="=" 0="")="" {="" mul="1;" }="" if(max="" <="" mul="" &&="" arr[i]!="0")="" {="" max="mul;" }="" }="" return="" max;="" }="" driver="" program="" to="" test="" above="" function="" int="" main()="" {="" int="" arr[]=" {-2," -3,="" 0,="" -2,="" -40};="" int="" n="sizeof(arr)/sizeof(arr[0]);" printf("maximum="" sub="" array="" product="" is="" %d",="" maxsubarrayproduct(arr,="" n));="" return="" 0;="" }="">

• Reply • Share ›

sahil • 4 months ago

I think answer is wrong for below case:
-40,-2,-6,-4,-100

expected :96000
geeks:4800

• Reply • Share ›

SID sahil • 4 months ago

Sahip! geeks answer is fine.

Please Note the wording of the Questin : It says "Maximum Product subarray" and you are considering it as "Maximum Product subsequence" .
U are removing element -2 from the array, so subarray left will be {-40} and {-6,-4,-100} both will not give the desired result, but if you remove -40 from the given array, subarray left will be {-2,-6,-4,-100} which will product to 4800.

1 • Reply • Share ›

d_geeks • 5 months ago

O(1) space, C program

http://ideone.com/JfLt2w

• Reply • Share ›

nish • 5 months ago

Time:O(n). space:O(1)
simple approach check it out !!
http://ideone.com/5SFhkH

• Reply • Share ›

Katamaran • 5 months ago

Much shorter and simpler to understand
http://yucoding.blogspot.com/2...

• Reply • Share ›

Guest • 6 months ago

http://code.geeksforgeeks.org/...

I think this would work in O(n) time and O(1) space

• Reply • Share ›

Victor • 6 months ago

```
static int maxproductSubArray(int [] array){
int length = array.length;
if(length == 1)
return array[0];
int max_so_far = array[0];
int max = array[0];
for(int i = 1; i < length; ++i){
max_so_far = max_so_far * array[i];

if(max_so_far > max)
max=max_so_far;
if(max_so_far == 0)
max_so_far = 1;
}
return max;
}
```

• Reply • Share ›

Victor • 6 months ago

http://ideone.com/HNUafK

• Reply • Share ›

MAYANK SHARMA • 6 months ago

Check this, its quiet easy,,,,,,plz suggest if there is anything wrong

http://ideone.com/HmE6Jd

• Reply • Share ›

Lokesh • 6 months ago

Check this !! This should be the easiest one. I tried with couple of samples, Hope i have not missed any scenario.
http://code.geeksforgeeks.org/...

Actual logic from https://changhaz.wordpress.com.... Changed it bit, to make it simpler.

• Reply • Share ›

soumyajit bhattacharyay • 6 months ago

easy c++ implementation

http://ideone.com/UZdrJU

• Reply • Share ›

harshit rastogi • 6 months ago

I ran against 184 test cases and all passed.
http://ideone.com/7h6E1I

• Reply • Share ›

Dman • 6 months ago

Implemented differently. Covers all the cases. Even when there is only one element.

https://ideone.com/8LG3Uj

2 • Reply • Share ›

Raj • 6 months ago

A C++ implementation covering all cases.
http://ideone.com/XxiGYO

• Reply • Share ›

contradiction • 6 months ago

solution handling all edge cases probably comment if any bug foung
http://ideone.com/XGBkkn

• Reply • Share ›

Klaus • 6 months ago

The above code ignores many cases.This one can be useful http://ideone.com/6A3zwA .

• Reply • Share ›

nikhil jain • 6 months ago

What about this solution
Steps of algorithm

Basically i have taken this idea from kaden algorithm for calculating maximum sum of of subarray. https://en.wikipedia.org/wiki/...

1.calculate product so far from left side & maintain in product variable.
2.each time compare product with max(here max is my answer).
3.when you get zero in product update product with 1.
4.After that,calculate product from right side & update max when product is greater than max.

my algo traverse two times the whole array.
example
-1,-2,-3,0,4
1.calculate from left side:
ans will be 4

2.calculate from right side:
ans will update with 6.

so final answer is 6
tell me if i am wrong

1 • Reply • Share ›

Rishu Agrawal → nikhil jain • 6 months ago

what about
1,4,-1,3,4,-5.-2

• Reply • Share ›

aditya • 7 months ago

all cases covered :D

http://ideone.com/82fE2Q

• Reply • Share ›

Shaan123 • 7 months ago

Simple and takes care of all cases.. http://ideone.com/h71U7p
Plz correct me if m wrong.

• Reply • Share ›

sifi Shaan123 • 7 months ago

Incorrect result for -1 0 -2 0

• Reply • Share ›

Load more comments