Leetcode Dynamic Programming (DP) Algorithms Problem Solving

How do I solve maximum product subarray problems?

> Find the contiguous subarray within an array (containing at least one number) which has the largest product.
> For example, given the array [2,3,-2,4],
> the contiguous subarray [2,3] has the largest product = 6.

As far as I understand there is a DP solution, which I can not understand. Any help?
Here is a link to the problem: Maximum Product Subarray

6 Answers

Gautam Kumar, CS Grad
Updated Nov 24, 2014

When the array has only positive elements then the product of all elements will be answer.
Problem becomes interesting and complex simultaneously when there are negative elements.

The way I looked at this problem is as follows.
You have three choices to make at any position in array.

1. You can get maximum product by multiplying the current element with maximum product calculated so far. (might work when current element is positive).
2. You can get maximum product by multiplying the current element with minimum product calculated so far. (might work when current element is negative).
3. Current element might be a starting position for maximum product sub array

so you have to maintain current maximum product and current minimum product.

```
curr_max_prod = A[0];
prev_max_prod = A[0];
prev_min_prod = A[0];
ans = A[0];
for i=1 to  n-1
{
   curr_max_prod = MAX ( prev_max_prod*A[i],
```

Related Questions

What is the solution for the maximum product subarray problem?

What is the mathematical proof of my solution to "Maximum Product Subarray" problem?

In "Introduction to Algorithms" 3rd edition, why is buying stock a maximum-subarray problem?

The maximum subarray problem has both a linear time algorithm and an nlogn divide and conquer algorithm. Are there parallel versions with the ...

What are some applications of maximum subarray problems?

What is the most time efficient way to find maximum subarray sum?

How can I count the no. of subarrays that have the product of elements smaller than a number D?

How do I solve the following problem: given an unsorted array of integers (negative and positive) find the maximum product of three elements o...

How does the following algorithm work: maximum circular subarray sum, GeeksforGeeks?

How can I find the maximum subarray with linear running time?

More Related Questions

```
    curr_min_prod = MIN ( prev_max_prod*A[i],
                          prev_min_prod*A[i],
                          A[i] );
    Ans = MAX(ans, curr_max_prod);
    prev_max_prod = curr_max_prod;
    prev_min_prod = curr_min_prod;
}
return ans;
```

Above algorithm requires O(n) time and constant space, very similar to Kadane's algorithm.

17.2k Views · 116 Upvotes · View Timeline

Timon Manfred Gehr
Updated Oct 28, 2015

If the array has only one element, return that element.
Otherwise, compute for each index the non-positive and non-negative products of maximum magnitude of a non-empty contiguous subarray ending there. Return the largest non-negative product encountered.

You can compute the maximum magnitude products as follows (we use 0 as a sentinel value):
Let $A_0, \dots A_{n-1}$ denote the elements of your array, then
$p_0 = m_0 = 0$,

$$p_{i+1} = \begin{cases} \max(1, p_i) \cdot A_i & \text{if } A_i > 0 \\ m_i \cdot A_i & \text{otherwise} \end{cases},$$

$$m_{i+1} = \begin{cases} \max(1, p_i) \cdot A_i & \text{if } A_i < 0 \\ m_i \cdot A_i & \text{otherwise} \end{cases}.$$

The answer is

$$\begin{cases} A_0 & \text{if } n = 1 \\ \max\limits_{i \in [n]} p_i & \text{otherwise} \end{cases}.$$

Implementation:

```
int p=0,m=0,r=0;
if(n==1) r=A[0];
else for(int i=0;i<n;i++){
  p=max(1,p)*A[i], m*=A[i];
  if(p<0) swap(p,m);
  r=max(p,r);
}
```

(This solution works if the array entries are arbitrary real numbers. For the more specific problem where all entries are integers, an alternative solution would be to split the array at values 0 and to return the maximum prefix/suffix product among the resulting subarrays.)

7.5k Views · 14 Upvotes · View Timeline

I used some what different approach and got AC at GeeksForGeeks(could not find any other OJ to test the efficiency and correctness of my code for this problem).

Algorithm :

1. We will deal with different subarrays separated by 0. As two or more subarrays with a zero between them will always give a combined product as zero.
2. We will maintain a variable that store the 'product obtained so far' after the last zero was encountered. That is once we encounter a zero we start taking product from the next element with initializing the variable maintaining the 'product obtained so far' to 1. By this we can get the total running product of each subarray separated by zero.
3. We maintain another variable that counts the total number of negative elements.
4. One more variable is used that stores the product till 'first negative element' in each subarray separated by zero(its use will be explained later).
5. Now, suppose we are at an element with index i, if total number of negative elements till here are even then : 'max product ending at i' = product so far.
   else if the number of negative elements are odd then : 'max product ending at i = max product so far / product till first negative element.
   (removing the first negative element will make the total negative elements even).
6. The global maximum can be obtained by taking the max of 'max product for each subarray ending at index i'.
7. The algorithm runs in a time complexity O(N).

C++ implementation :

```
#include <bits/stdc++.h>
#define isneg(x) ((x < 0 ? 1 : 0))
int ar[10];
int prodTillFirstNeg(int index, int elems, int &remove){
    remove = 1;
    for(int i = index; i < elems and ar[i] != 0; ++i){
        remove *= ar[i];
        if(ar[i] < 0)
            return i;
    }
    return elems;
}
int getResl(int elems){
    int max_prod = 0, so_far = 1, current, negs = 0, remove, index = pro
    for(int i = 0; i < elems; ++i){
        if(ar[i] == 0){
            so_far = 1;
            negs = 0;
            index = prodTillFirstNeg(i + 1, elems, remove);
            continue;
        }
        negs += isneg(ar[i]);
        so_far *= ar[i];
        if(negs % 2 == 0)
            current = so_far;
        else{
            if(index == i)
                current = so_far / remove * ar[i];
            else
                current = so_far / remove;
        }
        max_prod = std :: max(max_prod, current);
    }
    return max_prod;
}
void solve(){
    int elems;
    std::cin >> elems;
    for(int i = 0; i < elems; ++i)
```

```
int main(int argc, char const *argv[]){
    int test;
    std::cin >> test;
    while(test--)
        solve();
    return 0;
}
```

535 Views · 2 Upvotes · View Timeline

Upvote 2 Downvote Comment

Varun Vinayak
Answered Feb 25

The below function returns the maximum product subarray:

```
static int getMaximumProductSubarray(int[] arr)
{
int currentPositive = arr[0];
int currentNegative = arr[0];
int overallPositive = arr[0];
int overallNegative = arr[0];
for(int i = 1; i < arr.length; i++)
{
int temp = currentPositive;
currentPositive = Math.max(arr[i], Math.max(currentPositive * arr[i], cu
currentNegative = Math.min(arr[i], Math.min(temp * arr[i], currentNegati
overallPositive = Math.max(overallPositive, currentPositive);
overallNegative = Math.min(overallNegative, currentNegative);
}
return overallPositive;
}
```

568 Views · 3 Upvotes · View Timeline

Upvote 3 Downvote Comment

Anonymous
Answered Jun 25, 2015

What about this solution
Steps of algorithm

1.calculate product so far from left side & maintain in max product.
2.when you get zero in product update product with 1.

2.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

4k Views · 1 Upvote · View Timeline

Upvote 1 Downvote Comment

Phanindra Saggurthi, F.R.I.E.N.D and Polymath in progress....
Updated Oct 5, 2016

try below approach...

If array has only one element, return that element other wise you've to iterate all over the array to find out maximum sub array from each position.
For the input, [2 3 -2 4], you need to use two loops cause we need to find sub array from each position.

int prev_max=1;

for(int col=row;col<A.size();col++){

if(row == col)

prev_max=A[col];

else

prev_max=prev_max*A[col];

max_elem=max(prev_max,max_elem);

}

}

For row-0,col-0 - the maximum we can have in this position is 2,and set prev_max as 2 cause we need for further calculations in the row, max_elem-2

For row-0,col-1 - the maximum value we can get by compute by prev_max*array[1]i.e., is 6 ,max_elem-6

we are comparing our maximum value for each position:: max_elem=max(prev_max,max_elem);

For row-0,col-2 - the maximum value we can get by compute by prev_max*array[2]i.e., is -12 —->max_elem-6

For row-0,col-3 - the maximum value we can get by compute by prev_max*array[3]i.e., is -12 —->max_elem-6

For row-1,col-1 - the maximum we can have in this position is 3,prev_max-2,max_elem-6

For row-1,col-2 - the maximum we can have in this position is prev_max*array[2],prev_max-(-6),max_elem-6

For row-1,col-2 - the maximum we can have in this position is prev_max*array[2],prev_max-(-6),max_elem-6

For row-1,col-3 - the maximum we can have in this position is prev_max*array[3],prev_max-(-24),max_elem-6

For row-2,col-2 - the maximum we can have in this position is -2 , prev_max=(-2),max_elem-6

For row-2,col-3 - the maximum we can have in this position is prev_max*array[3],prev_max-(-24),max_elem-6

For row-3,col-3 - the maximum we can have in this position is 4 , prev_max=(4),max_elem-6

here now you can return max_elem...

EDIT: the above one is O(n*n) run time complexity, the below runs in O(n) .

Thanx to Gautam Kumar

int maxProduct(vector<int>& nums) {

int ans=nums[0];

int cur_max_prod=nums[0];

int cur_min_prod=nums[0];

int prev_max_prod=nums[0];

int prev_min_prod=nums[0];

ums[i]));

cur_min_prod=std::min(prev_max_prod*nums[i],std::min(prev_min_prod*nums[i],nums[i]));

ans=max(ans,cur_max_prod);

prev_max_prod=cur_max_prod;

prev_min_prod=cur_min_prod;

}

return ans;

}

1.6k Views · View Timeline

Upvote Downvote Comment

1 Answer Collapsed (Why?)

Top Stories from Your Feed

Answer · Mathematics

Are all Chinese people good at math?

Kevin D. Aslan, Entrepreneur, Author, Podcaster

Answered Jun 8 · Upvoted by Shubham Yadav, Master's Mathematics, Sardar Vallabhbhai National Institute of Technology, Surat (2020)

No, but they have a small advantage that can turn into a big one: their language. In Malcolm Gladwell's Outliers, the

Read In Feed

Answer · Operating Systems · Topic you might like

Why has nobody created a 128-bit operating system yet?

John R. Jorgensen, Computer Security professional, 2 graduate degrees in computers, 25+ years in IT

Updated May 30 · Upvoted by Metta Ramesh, Masters Computer Programming, Osmania University (2009)

As has been already mentioned there

Architectural component	64-bit Windows	32-bit Windows
Virtual memory	16 terabytes	4 GB
Paging file size	256 terabytes	16 terabytes
Hyperspace	8 GB	4 MB
Paged pool	128 GB	470 MB
Non-paged pool	128 GB	256 MB
System cache	1 terabyte	1 GB
System PTEs	128 GB	660 MB

Read In Feed

Answer · Computer Programming

What are some mistakes you can make as a programmer that will get you fired immediately?

Don Ramos, Likes to read, loves history

Answered Jun 7 · Upvoted by Tim Broberg, 25 years a hardware and software developer, still finding new mistakes to make and Oladeji Abubakar Adebowale, Mobile Application Developer,

I once had a colleague who was newly married and thought that it would be romantic to

Read In Feed