Find the Increasing subsequence of length three with maximum product

Given a sequence of non-negative integers, find the subsequence of length 3 having maximum product with the numbers of the subsequence being in ascending order.

Examples:

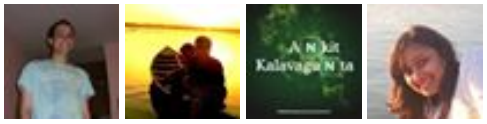
```
Input:
arr[] = {6, 7, 8, 1, 2, 3, 9, 10}
Output:
8 9 10

Input:
arr[] = {1, 5, 10, 8, 9}
Output: 5 8 9
```

Since we want to find the maximum product, we need to find following two things for every element in the given sequence:
LSL: The largest smaller element on left of given element
LGR: The largest greater element on right of given element.

Once we find LSL and LGR for an element, we can find the product of element with its LSL and LGR (if they both exist). We calculate this product for every element and return maximum of all products.

A **simple method** is to use nested loops. The outer loop traverses every element in sequence.



Inside the outer loop, run two inner loops (one after other) to find LSL and LGR of current element. Time complexity of this method is $O(n^2)$.

We can do this **in O(nLogn) time**. For simplicity, let us first create two arrays LSL[] and LGR[] of size n each where n is number of elements in input array arr[]. The main task is to fill two arrays LSL[] and LGR[]. Once we have these two arrays filled, all we need to find maximum product LSL[i]*arr[i]*LGR[i] where 0 < i < n-1 (Note that LSL[i] doesn't exist for i = 0 and LGR[i] doesn't exist for i = n-1).

We can **fill LSL[]** in O(nLogn) time. The idea is to use a Balanced Binary Search Tree like AVL. We start with empty AVL tree, insert the leftmost element in it. Then we traverse the input array starting from the second element to second last element. For every element currently being traversed, we find the floor of it in AVL tree. If floor exists, we store the floor in LSL[], otherwise we store NIL. After storing the floor, we insert the current element in the AVL tree.

We can **fill LGR[]** in O(n) time. The idea is similar to this post. We traverse from right side and keep track of the largest element. If the largest element is greater than current element, we store it in LGR[], otherwise we store NIL.

Finally, we run a O(n) loop and **return maximum of LSL[i]*arr[i]*LGR[i]**

Overall complexity of this approach is O(nLogn) + O(n) + O(n) which is O(nLogn). Auxiliary space required is O(n). Note that we can avoid space required for LSL, we can find and use LSL values in final loop.

This article is contributed by **Amit Jain**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Geometric Algorithms

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without

Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3

Writing code in comment? Please use **ideone.com** and share the link here.

98 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Dhruv Sharma • 2 days ago

u don't need to find LSL in nlogn...use the same concept as RSL and obtain it

• Reply • Share ›

guest • 2 months ago

void max_mul(int arr[],int s)

{

int m1,m2,m3;

int i;

m1=m2=m3=0;

for(i=0;i<s;i++) {="" if(m1="" &&="" arr[i]="">m1)

{

if(m2 && arr[i]>m2)

{

if(m3 && arr[i]>m3)

{

see more

• Reply • Share ›

guest ➚ guest • 2 months ago

void max_mul(int arr[],int s)

{

int m1,m2,m3;

int i;

m1=m2=m3=0;

705

Subscribe

Recent Comments

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 4 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 7 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 32 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 34 minutes ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 59 minutes ago

newCoder3006 Code without using while loop. We can do it...

Find subarray with given sum · 1 hour ago

for(i=0;i<s;i++) {="" if(m1="" &&="" arr[i]="">m1)

{

if(m2 && arr[i]>m2)

{

if(m3 && arr[i]>m3)

{

see more

• Reply • Share ›

pretentious.bastard • 2 months ago

C# code

/*Find increasing subsequence of length 3 with max product*/

public static List<int> SubsequenceWithMaxProduct(int[] arr, int seqSize)

{

/*the strategy would be to create another array that contains the largest numb
element,

* curr elem inclusive. Then select the seqSize largest numbers of this array.*/

int[] maxToLeftInc = new int[arr.Length];

int maxToLeft = arr[0];

maxToLeftInc[0] = maxToLeft;

see more

Reply • Share ›

Meenal • 4 months ago

Can we sort the array in O(nlogn) and take product of last three elements?

1 Reply • Share ›

Ankit Chaudhary Meenal • 4 months ago

this is wrong approach :
eg : 5 10 8 9

According to u : 5 8 9 10
ans= 8*9*10 = 720
but this is not increasing subsequence

correct answer : 5*8*9

Reply • Share ›

Meenal Ankit Chaudhary • 4 months ago

My bad.. Thanks for clarification

Reply • Share ›

Guest • 4 months ago

Is is wrong to sort the array in O(nlogn) and take product of last three element

Reply • Share ›

sambhavsharma Guest • 3 months ago

You need to return the subsequence:
for sequence of numbers: 5 10 8 9

if you sort the array, you would get : 8 9 10 which is not a subsequence

instead 5 8 9 is the correct answer as it is clearly a subsequence.

Reply • Share ›

Roohi Syeda • 4 months ago

Can we use RMQ (Max) to find LSL, even this gives time complexity of O(nlog O(nlogn) not better than AVL

Reply • Share ›

Google Roohi Syeda • 3 months ago

My bad.. This doesn't work..

Reply • Share ›

Davis • 4 months ago

Sort the array in O(nlogn), or you can make a copy and sort. Pick the last/first sorted.

Reply • Share ›

guest Davis • 4 months ago

You should combine this with a reverse hashmap to check if the eleme
Ex: 6,7,8,10,9 correspond to indices 0,1,2,3,4 (Use hashmap(Java) or
Sorted array - 6,7,8,9,10 correspond to indices 0,1,2,4,3

Select last three element from the back and check if it is an increasing 3 elements and so on.

8,9,10 => 2,4,3 (non-increasing)
7,8,9 => 1,2,4 (increasing subsequence) - which is the answer.

1 Reply • Share ›

Ankit Chaudhary guest • 4 months ago

ur approach is wrong

Aaquib Khwaja ➦ Ankit Chaudhary • 4 months ago

How is the approach wrong, can u please explain ?

Reply • Share ›

coderor • 5 months ago

```
int main()

{ int arr[] = {1, 5, 10, 8, 9};

node * root = createAVL(arr, 5);

if there are more than 3 nodes

cout << maxProduct(root);

else

cout << "Not present 3 increasing number\n";

}

int maxProduct( node *root)

{

int count = countNodes(root->right);
```

see more

Reply • Share ›

Harendra Singh • 5 months ago

all the ones having solution of O(n) are wrong,

Reply • Share ›

Guantana Chikoslavia • 5 months ago

@GeeksForGeeks
This can be achieved in O(n) with O(n) space. Maintain a stack in increasing c
{code}
for(i=0; i<n; i++){="" while(stack.isempty()="=" false="" &&="" stack.top()="">
{
stack.pop();
}
stack.push();
//calculateproduct of last three elements in stack and comare it with maxSoFa
}
{code}

1 ▲ | ▼ • Reply • Share ›

Sourin Sutradhar • 6 months ago

This can be easily solved in O(n) time complexity and using O(n) auxiliary spa

1 ▲ | ▼ • Reply • Share ›

Harendra Singh ➦ Sourin Sutradhar • 5 months ago

can u tell us how?

▲ | ▼ • Reply • Share ›

Aravindan B • 7 months ago

Hi why cant we fill LSL[] in the same way as LGL[]. Any logic behind using an A
logic used

▲ | ▼ • Reply • Share ›

Harendra Singh ➦ Aravindan B • 5 months ago

if you just keep track of the lowest number until current number (like we
the largest number until current number) then we we cannot be sure th
smaller number than current number. Lets take an example

numbers 5 10 2 6 26
smaller no until current -1 5 5 2 2

now for 26 the smaller no we found is 2 but it is not the largest among 26

if we do this naively we have search all the way back from current no t current no, that will take O(n) time so we use a self balancing tree.

I hope its clear now. :)

Reply • Share ›

Kaife ➚ Aravindan B • 6 months ago

Since we have to find number less then current number but greater an which is less then current

Reply • Share ›

Yukang ➚ Aravindan B • 6 months ago

yes, I am also confusing about that, we can scan from left to right for fi

Reply • Share ›

Yukang ➚ Yukang • 6 months ago

Oh，no, we need to keep the result in order.

Reply • Share ›

Chen Pang • 7 months ago

Instead of treating each element in the iteration as the middle element, why no

For each element, the maximum product it can produce, is itself with two large order. We can find them by populating the following two arrays:

LGR: The largest greater element on right of given element. If there is no great

set LGR[i] = -1

SGR (second largest greater right): This is the largest greater element on the
If there is no such element, set SGR[i] = -1

Both LGR and SGR can be populated easily in O(n) time, thus making the ove

For array {6, 7, 8, 1, 9, 2, 3, 10}
LGR is { 10, 10, 10, 10, 10, 10, 10, -1 }
SGR is { 9, 9, 9, 9, -1, 3, -1, -1}

Then we can easily generate the result: (8, 9, 10).

In actual implementation, LGR and SGR should store index instead of values.
SGR[i], LGR[SGR[i]]

5 ^ | v • Reply • Share ›

Yukang ➚ Chen Pang • 6 months ago

actually this is not the right solution, you have to give a sequence resul

1 5 10 8 9

LGR 10 10 -1 9 -1

SGR 9 9 -1 -1 -1

you method give ans: 450

5 10 9

but the true ans is : 5 8 9

^ | v • Reply • Share ›

Chen Pang ➚ Yukang • 6 months ago

No, LGR for element 10 is -1, because 9 is smaller than 10. (Tl

the same, except null value is used instead of -1). Therefore, S
cannot use 9 either. That is the condition I put in the algorithm.
having value -1 in LGR should be simply ignored.

The correct values in your case would be:
1 5 10 8 9
LGR is: [10, 10, -1, 9, -1]
SGR is: [8, 8, -1, -1, -1]

In fact in the implementation, you should store index instead of
LGR is : [2, 2, -1, 4, -1]
SGR is : [3, 3, -1, -1, -1]

The triplets at i-th element is: i, SGR[i], LGR[SGR[i]]. Note, it is

So you can get the correct sequence:
i (1) => 5
SGR[i] (3) => 8
LGR[3] (4) => 9

1 ^ | v • Reply • Share ›

rk ➚ Chen Pang • 6 months ago

Can u please tell the answer for this example :
a = [10,20,1235,40,50]. I dont think this algo will work for

^ | v • Reply • Share ›

Chen Pang ➚ rk • 6 months ago

Indeed, the algorithm is wrong

1 ^ | v • Reply • Share ›

Hariprasadmce ➚ Chen Pang • 5 months ago

I think your algo should work

-1 10 20 20 40
1235 1235 -1 50 -1

- 10*20*1235 - 20 * 40 *50 -

Answer is 10 * 20 * 1235

1 ^ | v • Reply • Share ›

Guest • 7 months ago

Can't delete comment?

^ | v • Reply • Share ›

Sandeep Sharma • 7 months ago

Another possible solution for filling LSL[] though(O(nlogn)) is to construct segr minimum element range queries for index 0 to n

^ | v • Reply • Share ›

Harendra Singh ➦ Sandeep Sharma • 5 months ago

I dont think that would work, becoz it will just give the min of the range l smaller no than current no
Please reply if I am wrong.

^ | v • Reply • Share ›

Govind • 7 months ago

Simply we can sort the array first in O(nlogn). Then start from Last element of distinct element. O(nlogn)+o(n)

Arrays.sort(int arr[]);

int i=arr.length-1; int j=0;

int product=1;

```
if(i>0 && arr[i] > arr[i-1])
{
Product=product*arr[i];
j++;
}
i--;
}
```


Harendra Singh Govind • 5 months ago

sorry bro but that's completely wrong
lets take an eg
100 10 20 30 1 2 3
ans we 10*20*30
your approach => 1 2 3 10 20 30 100 => 100*30*20

see when you sort the list you loose the original order which is imp for
to be in a order.

hope that's clear.

Abhay pandey • 7 months ago

O(n) solution:

```
#include<stdio.h>

void main() //main function begin

{

int arr[20]; //array maximum size
```

int max=0,p,k=-1; //variable declaration

int i,n;

printf("enter the range of array:");

scanf("%d",&n); //how much element you want to enter

printf("enter the element:");

for(i=0;i<n;i++) scanf("%d",&arr[i]);="" element="" insertion="" for(i="0;i<n-2;i

see more

• Reply • Share ›

Abhay pandey • 7 months ago

• Reply • Share ›

Neha Garg • 7 months ago

can we not fill lsl[] in 0[n] like lgr ????

3 • Reply • Share ›

Harendra Singh ➦ Neha Garg • 5 months ago

Nope

if you just keep track of the
lowest number until current number (like we are doing in of LGL, keepi
track of the largest number until current number) then we we cannot b
sure that the number is largest among all the smaller number than
current number. Lets take an example

numbers 5 10 2 6 26
smaller no until current -1 5 5 2 2

now for 26 the smaller no we found is 2 but it is not the largest among 26

if we do this naively we have search all the way back from current no to find the largest number smaller than current no, that will take O(n) time so we use a self balancing tree.

I hope its clear now. :)

dec C# • 8 months ago

static void CompareToMaxProduct(List<int> win, ref int max, ref int[] curr)

{

if (win == null)

throw new ArgumentNullException();

int currPr = 1;

win.ForEach(item => { currPr *= item; });

if(currPr > max)

{

max = currPr;

curr = win.ToArray();

}

tammana dec C# • 8 months ago

correct me if i m wrong..
it can be done in O(n) times.
consider
main(){
int a[]={15,8,12,9,65,88,96,6,84};
int b[3]={0,0,0};
int k=0;// maximum sub sequence of array of length 3
for(int i=0;i<a.length();i++) {="" if(b[2]<a[i])="" {="" b[k]="a[i];" k++;="" if(
{
k=0;
}
}
}
}//b[k] will give the maximum sub sequence in increasing order,k can b

1 • Reply • Share ›

Kai Luo • 8 months ago

What is the floor of an element

• Reply • Share ›

Harendra Singh Kai Luo • 5 months ago

100
50 170
25 60 120 200
10 30 55 65 118 125 180 210
floor if the right-most child of the left child ie largest element from left s
floor of 100 is 65
floor of 50 is 30
floor of 170 is 125
floor of 120 is 118

Hope that's clear now

Reply • Share ›

MS ➚ Harendra Singh • 2 months ago

Floor and Inorder Predecessor are the same right

1 Reply • Share ›

Neeraj Arora • 10 months ago

Maybe u have misunderstood the problem.

Reply • Share ›

Abhishek Kumar Maurya • 10 months ago

One more solution I have.
1. Sort the Array(O(nlogn)).
2. find prod=a[i-1]*a[i]*a[n-1]
3. check for the max prod and print the a[i-1], a[i], a[n-1].
Cmplexity will be same as O(nlogn) + O(n).

2 Reply • Share ›

Harendra Singh ➚ Abhishek Kumar Maurya • 5 months ago

sorry bro but that's completely wrong
lets take an eg
100 10 20 30 1 2 3
ans we 10*20*30
your approach => 1 2 3 10 20 30 100 => this will give 20*30*100

see when you sort the list you loose the original order which is imp for
to be in a order.

hope that's clear.

Reply • Share ›

Protocol Abhishek Kumar Maurya • 7 months ago

if u r sorting, then u r not finding increasing subsequence

• Reply • Share ›

Prashanth Abhishek Kumar Maurya • 8 months ago

It's not necessary that a[n-1] should be in the max prod. Also why are y
a[i-1]*a[i] ?

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site