Find a sorted subsequence of size 3 in linear time

Given an array of n integers, find the 3 elements such that a[i] < a[j] < a[k] and i < j < k in 0(n) time. If there are multiple such triplets, then print any one of them.

Examples:

```
Input:  arr[] = {12, 11, 10, 5, 6, 2, 30}
Output: 5, 6, 30

Input:  arr[] = {1, 2, 3, 4}
Output: 1, 2, 3 OR 1, 2, 4 OR 2, 3, 4

Input:  arr[] = {4, 3, 2, 1}
Output: No such triplet
```

Source: Amazon Interview Question

Hint: Use Auxiliary Space

Solution:
1) Create an auxiliary array smaller[0..n-1]. smaller[i] should store the index of a number which is smaller than arr[i] and is on left side of arr[i]. smaller[i] should contain -1 if there is no such element.
2) Create another auxiliary array greater[0..n-1]. greater[i] should store the index of a number which is greater than arr[i] and is on right side of arr[i]. greater[i] should contain -1 if there is no such element.
3) Finally traverse both smaller[] and greater[] and find the index i for which both smaller[i] and greater[i] are not -1.

Google™ Custom Search

GeeksforGeeks

53,521 people like GeeksforGeeks.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

```
#include<stdio.h>

// A function to fund a sorted subsequence of size 3
void find3Numbers(int arr[], int n)
{
   int max = n-1; //Index of maximum element from right side
   int min = 0; //Index of minimum element from left side
   int i;

   // Create an array that will store index of a smaller
   // element on left side. If there is no smaller element
   // on left side, then smaller[i] will be -1.
   int *smaller = new int[n];
   smaller[0] = -1;  // first entry will always be -1
   for (i = 1; i < n; i++)
   {
       if (arr[i] <= arr[min])
       {
          min = i;
          smaller[i] = -1;
       }
       else
          smaller[i] = min;
   }

   // Create another array that will store index of a
   // greater element on right side. If there is no greater
   // element on right side, then greater[i] will be -1.
   int *greater = new int[n];
   greater[n-1] = -1;  // last entry will always be -1
   for (i = n-2; i >= 0; i--)
   {
       if (arr[i] >= arr[max])
       {
          max = i;
          greater[i] = -1;
       }
       else
          greater[i] = max;
   }

   // Now find a number which has both a greater number on
   // right side and smaller number on left side
   for (i = 0; i < n; i++)
   {
       if (smaller[i] != -1 && greater[i] != -1)
```

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 stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

```
        {
            printf("%d %d %d", arr[smaller[i]],
                             arr[i], arr[greater[i]]);
            return;
        }
    }

    // If we reach number, then there are no such 3 numbers
    printf("No such triplet found");

    // Free the dynamically alloced memory to avoid memory leak
    delete [] smaller;
    delete [] greater;

    return;
}
```

```
// Driver program to test above function
int main()
{
    int arr[] = {12, 11, 10, 5, 6, 2, 30};
    int n = sizeof(arr)/sizeof(arr[0]);
    find3Numbers(arr, n);
    return 0;
}
```

Output:

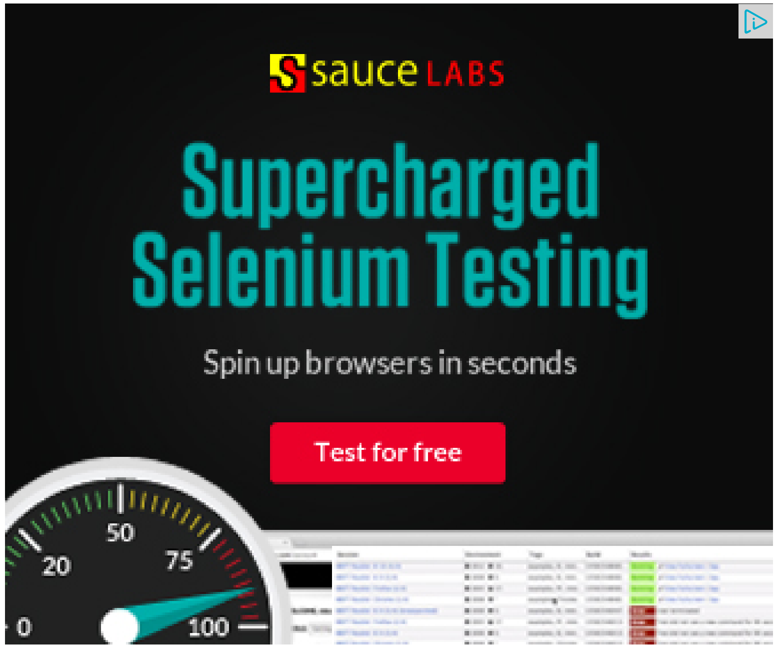
```
5 6 30
```

Time Complexity: O(n)
Auxliary Space: O(n)

Source: How to find 3 numbers in increasing order and increasing indices in an array in linear time

Exercise:
1. Find a subsequence of size 3 such that arr[i] < arr[j] > arr[k].
2. Find a sorted subsequence of size 4 in linear time

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above



705 Subscribe

Recent Comments

Aman Hi, Why arent we checking for conditions...
Write a C program to Delete a Tree. · 18 minutes ago

kzs please provide solution for the problem...
Backtracking | Set 2 (Rat in a Maze) · 21 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...
Root to leaf path sum equal to a given number · 46 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...
Count trailing zeroes in factorial of a number · 48 minutes ago

newCoder3006 If the array contains negative numbers also. We...
Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it...
Find subarray with given sum · 1 hour ago

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

4 Tweet 0 0

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

63 Comments **GeeksforGeeks**

Sort by Newest

Join the discussion…

xxmajia • 3 months ago

for finding any length of sequence, we can just sort the array using O(nlgn) tim
LIS on it

Reply • Share ›

Sumit Monga • 3 months ago

for doing it for 4 numbers , i < j < k < l such that arr[i] < arr[j] < arr[k] < arr[l],we
and then between iterate for j and k in between using a list(serving purpose of
understand it:

void find4Numbers(int arr[], int n)

{

int max = n-1; //Index of maximum element from right side for k

int min = 0; //Index of minimum element from left side for j

int i;

// Create an array that will store index of a smaller

// element on left side. If there is no smaller element

// on left side, then smaller[i] will be -1.

see more

Reply • Share ›

Sumit Monga • 3 months ago

for doing it for 4 numbers , i < j < k < l such that arr[i] < arr[j] < arr[k] < arr[l],we
and then between iterate for j and k in between using a list(serving purpose of
understand it:

void find4Numbers(int arr[], int n)

{

int max = n-1; //Index of maximum element from right side for k

int min = 0; //Index of minimum element from left side for j

int i;

// Create an array that will store index of a smaller

// element on left side. If there is no smaller element

// on left side, then smaller[i] will be -1.

int *smaller = new int[n];

see more

Reply • Share ›

asunel • 7 months ago

Can be done using only one extra array !!!

See http://ajscodeblog.blogspot.in...

Reply • Share ›

nishant08 • 9 months ago

```
 plz tell me if there is a problem
#include<stdio.h>
main()
{
        int arr[10],i,j=0,n,t[3];
        printf("enter no. of elements- ");
        scanf("%d", &n);
        for(i=0;i<n;i++)
        {
                printf("enter data- ");
                scanf("%d", &arr[i]);
        }
        for(i=0;i<n;i++)
        {
                if(j==0)
                t[j]=arr[i];

                if(arr[i]<arr[i+1])
```

see more

Reply • Share ›

Nikunj Banka • 10 months ago

see this:

http://stackoverflow.com/quest...

Reply • Share ›

aspire • 10 months ago

I have a solution with O(n) time complexity and O(1) space.
Please correct me if i am wrong.

```
 #include<stdio.h>
```

```
int find3Size(int a[],int n)
{
    int mn=0,mx=n-1,i=1,j=n-2;
    if(n<3)
        return -1;

    while(j>=i)
    {
        if(a[j]>=a[mx])
            mx=j--;
        else
            j--;
        if(a[i]<=a[mn])
```

see more

1 ^ | v • Reply • Share ›

Sourabh ➦ aspire • 9 months ago

Yes you are missing Half of the range to check.

```
/* Paste your code here (You may delete these lines if not wr
```

^ | v • Reply • Share ›

skulldude • 10 months ago

In the above code, I think it is not necessary to check whether the first and last elements as it will always be -1.

Below is the code to solve the problem.

```
// minLeft[i] - Minimum element from a[0] to a[i]
// maxRight[i] - Maximum element from a[len-1] to a[i]

void find3Numbers(int a[],int len){
        if(!a || len<=2){
                printf("Invalid\n");
                return;
        }

        int *minLeft=new int[len],*maxRight=new int[len];
        minLeft[0]=a[0],maxLeft[len-1]=a[len-1];
```

see more

Reply • Share ›

aspire skulldude • 10 months ago

I have a solution with O(n) time complexity and O(1) space.
Please correct me if i am wrong.

```
 #include<stdio.h>

int find3Size(int a[],int n)
{
    int mn=0,mx=n-1,i=1,j=n-2;
    if(n<3)
        return -1;

    while(j>=i)
    {
        if(a[j]>=a[mx])
            mx=j--;
        else
            j--;
        if(a[i]<=a[mn])
```

see more

Reply · Share ›

Ganesh · a year ago

You can find java code here:

[sourcecode language="JAVA"]
/**
* Given an array of n integers, find the 3 elements such that a[i] < a[j] < a[k] an
* If there are multiple such triplets, then print any one of them.
* Example: Input: arr[] = {12, 11, 10, 5, 6, 2, 30} Output: 5, 6, 30
* @author GAPIITD
*
*/
public class SortedSubsequenceOfSize3 {

public static void main(String[] args) {
int arr[] = {12, 11, 10, 5, 6, 2, 30};
int small[] = new int[arr.length];
int large[] = new int[arr.length];
small[0] = large[arr.length - 1] = -1;
int min = arr[0], max = arr[arr.length - 1];

see more

Reply · Share ›

azee · a year ago

Can't we user longest increasing subsequence logic for this problem. And usi

print all subsequences with length more than or equal to 3. In this way we get
subsequence of size 3. If I am wrong, please let me know where this logic mig

^ | ⌄ • Reply • Share ›

Paparao Veeragandham ➦ azee • a year ago

It will take O(nlogn) time.

```
/* Paste your code here (You may delete these lines if not wr
```

^ | ⌄ • Reply • Share ›

Paparao Veeragandham ➦ Paparao Veeragandham • a year ago

Please post solution for sorted sequence of size 4 in linear time

^ | ⌄ • Reply • Share ›

VikasG • 2 years ago

I think this will work. Let me know if you can spot any issues.

```
void printSubSequence3(int array[], int n) {
   //error checking
   assert( n >= 3 );

   int p = -1; // index of min0
   int q = -1; // index of min1
   int r = -1; // index of min2

   for (int i = 1; i < n ; i++) {
       //compare i with i - 1
       if (array[i] > array[i-1]) { //found increasing subSequence
           if ( p == -1 && q == -1) {
               //Our new subSequence is array[i-1], array[i], ?
               q = i;
               p = i - 1;
```

```
        } else {
```

see more

^ | ⌄ • Reply • Share ›

Amandeep Gautam • 2 years ago

I think that this algorithm will also work. If any mistakes please point out.

```
// initialize four variables to 0.
int i=0, j=0, k=0, counter=0
//start parsing the array.
while ( a[i] j
if ( a[counter] > a[j] ) {
//print the answer
break;
}
else if ( a[counter] > a[i] && a[counter] < a[j] ) {
j=counter;
if ( k!=0 ) {
i=k;
k=0;
}
}
else if ( (a[counter] < a[i]) || (a[counter]< a[k]) ) {
k=counter;
}
```

^ | ⌄ • Reply • Share ›

Amandeep Gautam ➦ Amandeep Gautam • 2 years ago

The above algorithm is o(n) time and O(1) space.

^ | ⌄ • Reply • Share ›

Ifenjoy9 • 2 years ago

```
 // find the first adjacent elements in ascending order and co
// compare the two pairs of adjacent elements
// A[x] > A[i] => i, x, x + 1
// A[x+1] > A[j] => i, j, x + 1
// set i to x and j to x + 1 and otherwise continue
public void find(int[] A)
{
        int i = -1;
        int j = -1;
        int k = -1;
        int x = 0;
        int l = A.length;
        while(x < l - 2 && A[x+1] <= A[x]) x++;
        if(x < l - 2) {
                i = x;
                j = x + 1;
                x++;
                while(x < l - 1) {
```

see more

Reply • Share ›

Nirdesh ➦ Ifenjoy9 • a year ago

Your soluation will not work for input :

{ 1,12, 11, 2, -2,-5,10 }

where expected result is : 1,2,10 but your program prints nothings.

Reply • Share ›

nitin gupta iitian • 2 years ago

/* Paste your code here (You may delete these lines if not writing code) */
[@geeksforgeeks
Its very easy dud
it like a maximum and minimum problem

Logic is very simple
actually we have to first find the minimum value .once we encounter minimum value
greater than previous value
{12, 11, 10, 5, 6, 2, 30,5,3,32,2,40};
first we found minimum value 11 then 10 then 5 (since 6 > 5)
now we find greater value's which is greater to previous one
we have 5 now next is 6 next is 30 next is 32 and 40

EASY NA!
I DON'T KNOW WHY YOU NEED TWO ARRAY ?
]

Reply • Share ›

andrew nitin gupta iitian • 2 years ago
this will be wrong in case:

{12, 11, 10, 5, 6, 2, 3, 4}

```
/* Paste your code here (You may delete these lines if not wr:
```

Reply • Share ›

NITIN GUPTA IITIAN • 2 years ago
@geeksforgeeks

its very easy dud
it like a maximum and minimum problem

simplest Logic is
actually we have to first find the minimum value .once we encounter minimum value
greater than previous value
{12, 11, 10, 5, 6, 2, 30,5,3,32,2,40};
first we found minimum value 11 then 10 then 5 (since 6 > 5)
now we find greater value's which is greater to previous one
we have 5 now next is 6 next is 30 next is 32 and 40

EASY NA!

```
/* Paste your code here (You may delete these lines if not writing co
```

Reply • Share ›

lohith • 2 years ago

```
 import java.util.ArrayList;


public class subsequence3 {


        public static int array[] = {5,2,6,3,4};

        public static void main(String str[]){

                ArrayList<Integer> inn = new ArrayList<Integer>();
                for(int i=0;i<array.length;i++){
                        inn.add(i);
                }
```

```
            ArrayList<Integer> resu = Subsequence(inn,0);

            for(int i=0;i<resu.size();i++){
```

see more

Reply • Share ›

lei • 2 years ago

build auxiliary array will be o(n*n); so average it is not o(n)

Reply • Share ›

ashu lei • 2 years ago

you can build auxiliary array using stack in o(n) time.

```
/* Paste your code here (You may delete these lines if not wr
```

Reply • Share ›

ashu lei • 2 years ago

you can build your auxillary array using stack. it will take o(n) time.

```
/* Paste your code here (You may delete these lines if not wr
```

Reply • Share ›

Ashu • 2 years ago

I think approach for 4 increasing sorted sub-sequence is same , you need to s
a[i] in greater[i] and last smaller element of a[i] in smaller[i].

then for every index you can check if for index i , (smaller[i]!=-1 and smaller[sr
smaller[i]!=-1 and greater[i]!=-1 and greater[greater[i]])

```
/* Paste your code here (You may delete these lines if not writing c
```

^ | ⌄ • Reply • Share ›

souvik • 2 years ago

O(n) time O(1) space for the 3 element problem

```
 void find(int x[],int n)
{
 int i=0,j=n,k=n,l;
 if(n<3)
  {
     printf("Not enough elements");
     return;
  }
  for(l=1;l<n;l++)
  {
    if(j==n)
      {
        if(x[l]>x[i])
           j=l;
         else
           i=l;
```

see more

^ | ⌄ • Reply • Share ›

anonymous ➦ souvik • 2 years ago

@souvik

You should always put comments in your code so that one can unders

^ | ⌄ • Reply • Share ›

Anurag Singh • 2 years ago
Exercise 1:
Find a subsequence of size 3 such that arr[i] arr[k]

Here we will calculate two smaller arrays, say smaller1 and smaller2.
smaller1 array will be exactly same as smaller array in solution of original prob
number on the left side)
smaller2 will be opposite, it will store the index of a smaller number in right sid

Now traverse smaller1 and smaller2 arrays, find an index i where
smaller1[i] != -1 && smaller2[i] != -1

And then expected output will be:
arr[smaller1[i]], arr[i], arr[smaller2[i]]

• Reply • Share ›

Aadarsh Anurag Singh • a year ago
Pretty simple. Liked it. We can also have a third auxialiary array which
min(whichever the case maybe) for every index.

```
/* Paste your code here (You may delete these lines if not wr
```

• Reply • Share ›

Anurag Singh • 2 years ago
Exercise 2:
Find a sorted subsequence of size 4 in linear time

Here we will calculate greater and smaller arrays as is.
Now if at all, there is an sorted subsequence of size 4, then there will be two d
smaller[i] != -1 && greater[i] != -1, also
smaller[j] != -1 && greater[j] != -1

i and j can be found by traversing forward and backward in same loop, and the

i and j can be found by traversing forward and backward in same loop. and the
arr[smaller[i]], arr[i], arr[j], arr[greater[j]]

if no such i and j found, then there is no sorted subsequence of size 4.

Reply • Share ›

Game • 2 years ago

Apparently you have not free'd the memory you new'd.

Reply • Share ›

geeksforgeeks ➦ Game • 2 years ago

@game: Thanks for pointing this out. We have added code to delete d
up!

Reply • Share ›

Ankush • 2 years ago

Any approach in linear time to size 4 increasing sorted sub-sequence problem

Reply • Share ›

Ashu ➦ Ankush • 2 years ago

I think approach for 4 increasing sorted sub-sequence is same , you n
element of a[i] in greater[i] and last smaller element of a[i] in smaller[i].

then for every index you can check if for index i , (smaller[i]!=-1 and sm
1) or(smaller[i]!=-1 and greater[i]!=-1 and greater[greater[i]])

```
/* Paste your code here (You may delete these lines if not wr
```

Reply • Share ›

Ankush • 2 years ago

Any approach to the sorted sub-sequence of size 4 problem??

Reply • Share ›

Nicam • 2 years ago

O(n) solution with O(1) space

```
public ArrayList<Integer> find3Numbers (int[] a) {
   if (a == null || a.length < 3) return null;
   int i = -1, j = -1;
   ArrayList<Integer> ans = new ArrayList<Integer>(3);

   for (int k = 1; k < a.length; ++k) {
           if (i != -1 && j != -1 && a[k] > a[j]) {
                   ans.add(a[i]);
                   ans.add(a[j]);
                   ans.add(a[k]);
                   return ans;
           } else if (a[k] > a[k-1]) {
                   if (i != -1 && j != -1) {
                           i = (a[k-1] < a[i]) ? k-1 : i;
                           j = (a[k] < a[j]) ? k : j;
                   } else {
```

see more

^ | ⌄ • Reply • Share ›

Nicam ➦ Nicam • 2 years ago

Corrected one logic bug, the code wasn't working on {5,2,6,3,4}, now s

```
public ArrayList<Integer> find3Numbers (int[] a) {
   if (a == null || a.length < 3) return null;
   int i = -1, j = -1;
   ArrayList<Integer> ans = new ArrayList<Integer>(3);

   for (int k = 1; k < a.length; ++k) {
          for (int k = 1; k < a.length; ++k) {
                  if (i != -1 && j != -1 && a[k] > a[j]) {
                          ans.add(a[i]);
                          ans.add(a[j]);
                          ans.add(a[k]);
                          return ans;
                  } else if (a[k] > a[k-1]) {
                          if (i != -1 && j != -1) {
                                  i = (a[k-1] < a[i]) ? k-1 : i;
                                  j = (a[k] < a[j]) ? k : j;
                          } else {
```

see more

^ | v • Reply • Share ›

Prem → Nicam • 2 years ago
@Nicam Can u pls explain ur logic?
^ | v • Reply • Share ›

Ram • 2 years ago
says

simple
^ | v • Reply • Share ›

doom • 2 years ago
Any ideas for exercise 2: Find a sorted subsequence of size 4 in linear time?
^ | v • Reply • Share ›

Hamid • 2 years ago
I think your original code is not taking care of "equal to" condition properly. for e
following input:

ȍ ȍ ȍ ȍ ȍ ȍ ȍ

I believe that you should <= instead of just < in following statement while filling
if (arr[i] < arr[min])

and similar logic needs to be followed while filling up greater array as well.

• Reply • Share ›

geeksforgeeks Hamid • 2 years ago

@Hamid:
Thanks for pointing this out. We have changed the condition. Keep it u

• Reply • Share ›

abc • 2 years ago

Just thought an algo with only one stack,

for i = 0 to N and stack.size a[i], if yes then replace top with a[i], otherwise we
an increasing subsequence add a[i], and set found = 1;
3. if found is 1 and a[i] a[top-1] replace top with a[i]

in the end if size of stack is 3 we have found one of the subsequence.

Sample run for
arr[] = {12, 11, 10, 5, 6, 2, 30}

s = 12
s= 11 (since a[i] < a[top])
s = 10
s = 5
s = 5,6 (a[i] < a[top], found = 1)
s= 5,6 (2 is not added since, 6 2
s = 5,6,30 (output)

Please comment and let me know cases for which this approach fails.

^ | ⌄ • Reply • Share ›

anonymous ➦ abc • 2 years ago

does not work for..
{12,11,10,5,6,2,3,4}

^ | ⌄ • Reply • Share ›

manish • 2 years ago

This is not working, i think the...solution should contain for arr[] = {1,2,3,4}
output = {1,2,3},{1,3,4},{1,2,4}and {2,3,4}

```
[sourcecode language="C++"]
#include <cstdio>
#include <iostream>
using namespace std;

void find3Numbers(int arr[],int n);
int main()
{
int arr[] = {1,2,3,4};
int arr1[] = {12,11,10,5,6,2,30};
find3Numbers(arr,sizeof(arr)/sizeof(arr[0]));
find3Numbers(arr1,sizeof(arr1)/sizeof(arr1[0]));
return 0;
}

void find3Numbers(int arr[],int n)
```

see more

^ | ⌄ • Reply • Share ›

GeeksforGeeks ➦ manish • 2 years ago

@manish: Please take a closer look at the problem statement. It says given in post prints {1, 2, 4} for your input which looks correct.

Reply • Share ›

@geeksforgeeks GeeksforGeeks • 2 years ago

thank you...i got it

Reply • Share ›

Saravan • 2 years ago

Solutions to find first occurrence in passed array!

```
public class FindTriplets {
 public static void main(String[] args) {
   int [] arr1 = {12, 11, 10, 5, 6, 2, 30};
   int [] arr2 = {1, 2, 3, 4};
   int [] arr3 = {4, 3, 2, 1};
   solution1(arr1);
   solution1(arr2);
   solution1(arr3);

   solution2(arr1);
   solution2(arr2);
   solution2(arr3);
 }

 private static void solution2(int [] arr) {
   int i = 0;
```

see more

Reply • Share ›

Load more comments

Subscribe   Add Disqus to your site