Find the Minimum length Unsorted Subarray, sorting which makes the complete array sorted

Given an unsorted array arr[0..n-1] of size n, find the minimum length subarray arr[s..e] such that sorting this subarray makes the whole array sorted.

Examples:
1) If the input array is [10, 12, 20, 30, 25, 40, 32, 31, 35, 50, 60], your program should be able to find that the subarray lies between the indexes 3 and 8.

2) If the input array is [0, 1, 15, 25, 6, 7, 30, 40, 50], your program should be able to find that the subarray lies between the indexes 2 and 5.

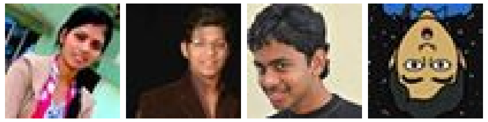
Solution:
1) Find the candidate unsorted subarray
a) Scan from left to right and find the first element which is greater than the next element. Let *s* be the index of such an element. In the above example 1, *s* is 3 (index of 30).
b) Scan from right to left and find the first element (first in right to left order) which is smaller than the next element (next in right to left order). Let *e* be the index of such an element. In the above example 1, e is 7 (index of 31).

2) Check whether sorting the candidate unsorted subarray makes the complete array sorted or not. If not, then include more elements in the subarray.
a) Find the minimum and maximum values in *arr[s..e]*. Let minimum and maximum values be *min* and *max*. *min* and *max* for [30, 25, 40, 32, 31] are 25 and 40 respectively.
b) Find the first element (if there is any) in *arr[0..s-1]* which is greater than *min*, change *s* to index of this element. There is no such element in above example 1.
c) Find the last element (if there is any) in *arr[e+1..n-1]* which is smaller than max, change *e* to

Google™ Custom Search

GeeksforGeeks

53,522 people like GeeksforGeeks.



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

index of this element. In the above example 1, e is changed to 8 (index of 35)

3) Print *s* and *e*.

Implementation:

```
#include<stdio.h>

void printUnsorted(int arr[], int n)
{
  int s = 0, e = n-1, i, max, min;

  // step 1(a) of above algo
  for (s = 0; s < n-1; s++)
  {
    if (arr[s] > arr[s+1])
      break;
  }
  if (s == n-1)
  {
    printf("The complete array is sorted");
    return;
  }

  // step 1(b) of above algo
  for(e = n - 1; e > 0; e--)
  {
    if(arr[e] < arr[e-1])
      break;
  }

  // step 2(a) of above algo
  max = arr[s]; min = arr[s];
  for(i = s + 1; i <= e; i++)
  {
    if(arr[i] > max)
      max = arr[i];
    if(arr[i] < min)
      min = arr[i];
  }

  // step 2(b) of above algo
  for( i = 0; i < s; i++)
  {
    if(arr[i] > min)
```

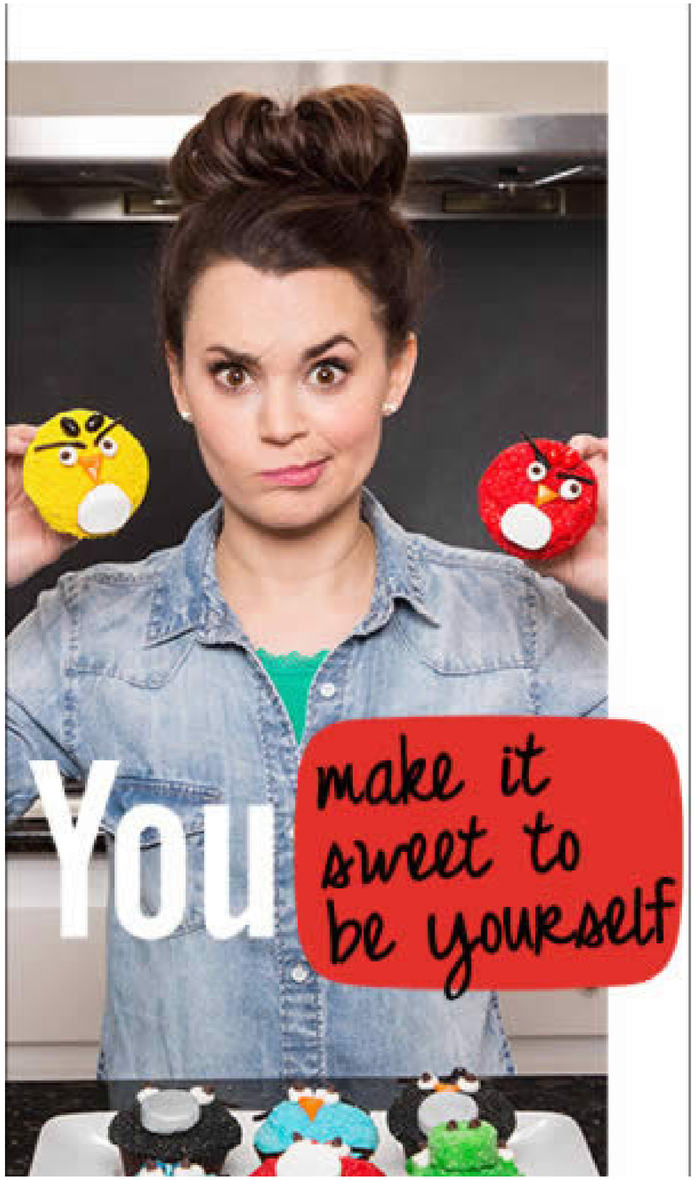
Popular Posts

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

```
      {
        s = i;
        break;
      }
    }

    // step 2(c) of above algo
    for( i = n -1; i >= e+1; i--)
    {
      if(arr[i] < max)
      {
        e = i;
        break;
      }
    }

    // step 3 of above algo
    printf(" The unsorted subarray which makes the given array "
           " sorted lies between the indees %d and %d", s, e);
    return;
  }

  int main()
  {
    int arr[] = {10, 12, 20, 30, 25, 40, 32, 31, 35, 50, 60};
    int arr_size = sizeof(arr)/sizeof(arr[0]);
    printUnsorted(arr, arr_size);
    getchar();
    return 0;
  }
```

Time Complexity: O(n)

Please write comments if you find the above code/algorithm incorrect, or find better ways to solve the same problem.



705 Subscribe

Recent Comments

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

Write a C program to Delete a Tree. · 24 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 28 minutes ago

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

Root to leaf path sum equal to a given number · 53 minutes ago

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

Count trailing zeroes in factorial of a number · 54 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...

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

7 Tweet 0 0

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

43 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Find subarray with given sum · 1 hour ago

Venu Gopal • 12 days ago

my solution is definitely not better than author's solution but nevertheless.
you can check a different approach, it is a simple code without using sort, bina
explained in detail with help of comments for better understanding of reader..
http://ideone.com/q7X3ES

groomnestle • 5 months ago

A (nlgn) solution is simpler: make a copy of array A into B, sort B using quicks
leftmost and rightmost elements that differ in A and B. The minimum subarray
elements.

Castle Age ➦ groomnestle • 4 months ago

time complexity nlgn and space complexity n
vs
time complexity n and space complexity constant

I would take latter anytime, plus it is pretty simple as well

Marsha Donna • 8 months ago

can sum1 pls give an eg to explain when part 2b of above solution arises??

2

Pushkar ➦ Marsha Donna • 7 months ago

Part 2(b) of the solution arises with the input {10,12,30,11,40,32,31,35,
example. so step 2(a) find min for [30,11,40,32,31] as 11. So step 2(b)
the index of 12. Hope you got it!!!

1

Marsha Donna Pushkar • 7 months ago

thanks ..how do v copy paste code from vc++ here..wenever i p
else...can u tel sum good material to understand asymptotic nc

Marsha Donna • 8 months ago

another way to implement part1 of the solution is as follows

```
void minunsort(int arr[],int n)
{
int diff[10],i,minind,maxind;
for(i=0;i<n-1;i++) {="" diff[i]="arr[i+1]-arr[i];" }="" for(i="0;i&lt;n-1;i++)" {="" if(diff
}="" for(i="n-2;i">=0;i++)
{

if(diff[i]<0)
{
maxind=i;
break;
}

}
maxind++;
printf("the subarray indices are %d and %d\n and the length is \n",minind,maxi

}
```

for the given eg the value held by minind would be 3 and maxind would be 7
but it uses auxillary space of O(n)

exor • 9 months ago

The last part where you scan the insertion point for min and max can be done
the starting and ending sequences are sorted)...

It doesn't reduce the complexity but it's technically more efficient.

^ | v •

kuldeepshandilya • 10 months ago

We can use binary search to find following indexes -
startIndexOfUnsortedSubArray = index of a number which is greater than its n

if there are are multiple such indexes, take the left most index

endIndexOfUnsortedSubArray = index of a number which is less than its previ

if there are are multiple such indexes, take the right most index

When there are more than one such unsorted arrays, the answer would be su

^ | v •

Ankit Gupta • 11 months ago

A really simple O(n) time and O(1) space solutions....

```
int func(int arr[], int n){//arr is the pointer to array and n is size of array.
int i=-1, j=-1;//i and j will finally store the required left and right index.
int max_so_far = arr[0], min_so_far = arr[n-1];.
for(int t=0;t<n;t++){.
max_so_far = MAX(max_so_far, arr[t]);.
if(arr[t]==max_so_far){if(j==-1)j=t;}.
else j=-1;.
min_so_far = MIN(min_so_far, arr[n-1-t]);.
if(arr[n-1-t]==min_so_far){if(i==-1)i=n-1-t;}.
else i=-1;.
}if(i==-1)i=0;else i++;.
```

if(j==-1)j=n-1;else j--;.
if(i<j)cout<<i<<" "<<j<<endl;.
else cout<<"array already sorted"<<endl;.
return 0;.
}

hunter • 11 months ago
//my code is simple
left=0;
right=n-1;
while(left<right)
{
while(a[left]a[right-1])
right--;
if(left<right)
{
//now left and right positions indicates min unsorted subarray
sort(a,left,right);
break;
}
}
//if any thing wrong correct me.............

hunter hunter • 11 months ago
sorry it won't work............

Priyanshu • 11 months ago
I have written a solution using 2 scans in a simplified way.
Please throw some test cases if it gets failed

```
 /* Paste your code here (You may delete these lines if not writing c
#include<iostream>
using namespace std;
int main()
{
    int a[]={0,5,3,2,9,6,7,3};
    int n=sizeof(a)/sizeof(a[0]);
    int i,start=0,end=n-1;
    int min=a[n-1],max=a[0];
    for(i=n-2;i>=0;i--)
    {
        if(a[i]<=min)
        min=a[i];
        else
        start=i;
```

see more

me.abhinav • 11 months ago

A different O(n) solution:

Let answer = (start, end), where 'start' = first index of the subarray to be sorte
to be sorted.

1) Scan the array a[0..n-1] from left-to-right. Let current index be 'i'. How can w
(i.e. 'start') of the subarray to be sorted??
If a[i] > minimum(a[i..n-1]) then this a[i] must be present at some other index >
'start' = i and we need to sort the array from 'start' upto some index 'end' (whic
'start' and break from loop.

2) Scan the array a[0..n-1] from right-to-left. Let current index be 'i'. Following t
then this a[i] must be present at some other index < i in the completely sorted

sort the array from "start" (calculated in step-(1)) upto "end". Print "end" and bre

With some pre-processing (which takes O(n) time), we can determine min(a[i
(see code).

me.abhinav ➦ me.abhinav • 11 months ago

I suppose the code could not be pasted correctly. So pasting it again:

```
 #include <iostream>
#define SIZ 100
#define MAX(a, b) (a>b)?a:b
#define MIN(a, b) (a>b)?b:a

using namespace std;

int main()
{
        int a[SIZ], n, min[SIZ], max[SIZ];
        int i;

        while(1){
                cout<<"Enter array size: ";
                cin>>n;
                cout<<"Enter "<<n<<" elements.\n";
```

see more

Ganesh • a year ago

You can find java code here:

[sourcecode language="JAVA"]

```
/**
* Given an unsorted array arr[0..n-1] of size n, find the minimum length subarr
* such that sorting this subarray makes the whole array sorted.
* Example:
* If the input array is [10, 12, 20, 30, 25, 40, 32, 31, 35, 50, 60], your program s
* find that the subarray lies between the indexes 3 and 8.
*
* @author GAPIITD
*
*/
public class MinimumLengthUnsortedSubarray {

public static void main(String[] args) {
int arr[] = {10, 12, 20, 30, 25, 40, 32, 31, 35, 50, 60};
findMinimumLengthUnsortedSubarray(arr);
```

see more

Vamsi Subhash • 2 years ago

```
 max=arr[0];
first=-1;
second=-1
for(I=0;i<n;i++) {
if (arr[I]>max) {
max=arr[I];
maxIdx=I;
}
else {
if(first==-1) first=I;
else second=I;
}
}
output arr[first] to arr[second] => Result
```

Tulasi · 2 years ago

It can be done using stacks also. But requires auxiliary space in form of stack

```
 #include<stdio.h>
#include "stack/stack.c"

int main(){
        int a[]={0, 1, 15, 25, 6, 7, 30, 40, 50};
        int n=sizeof(a)/sizeof(a[0]),i,j;

        Stack minS=getStackInstance();
        Stack maxS=getStackInstance();

        push(minS,a[n-1]);
        push(maxS,a[0]);

        for(i=1,j=n-2;i<n && j>=0;i++,j--){
                if(a[i] > peek(maxS))
                        push(maxS,a[i]);
```

see more

Palash ➦ Tulasi · 2 years ago

You don't need stacks to do what you are doing.

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

spandan • 2 years ago

we can sort the copy of original array first(Nlogn),then compare the mismatch

prakhar spandan • 2 years ago

if you are already sorting, then what's the significance of this informatic

abhimanu • 3 years ago

To optimize the part where we are finding the max and min between indices s than if followed by another if. Reason being if arr[i] is more than max then if ca iteration.

```
// step 2(a) of above algo
  max = arr[s]; min = arr[s];
  for(i = s + 1; i <= e; i++)
  {
    if(arr[i] > max)
      max = arr[i];
    else if(arr[i] < min)
      min = arr[i];
  }
```

torabul • 3 years ago

Very nicely done. Thanks.

roadies · 3 years ago

can we make a min and a max heap, and try to remove the elements from left
have smaller elements on their correct positions or not, and similarly for right t

hii · 3 years ago

```
int main()
{
int arr[]={1,2,4,7,6,3,9,4};
int start=end=0,largest=arr[0];
int i;
for(i=1;i<n;i++)
{
if(arr[i]<largest) //if current element is smaller than largest so
{ //so far then it is a part of subarray so set
if(!start) // end=i
start=i-1;
end=i;
}
else
largest=arr[i];
}
}
```

1

shanky · 3 years ago

can it be done using stacks??

Vineel Kumar Reddy · 3 years ago

```
int main()
{
        int a[] = {0, 1, 15, 25, 6, 7, 30, 40, 50,60}; //10, 12, 20, 
        int n;
        int i = 0;
        n = sizeof(a)/sizeof(int);
        int low,high;

        //search for high
        //find the number greater than its right side element i.e., 2
        for(i = 0;i < n-1;i++ ){
                if(a[i] > a[i+1]){
                        high = i;
                        break;
                }
        }
        //now from right find which number is less than 25 i.e., 7
```

see more

Prateek Caire • 3 years ago

Another solution in O(n)

```
 L(i)
        if(i == 0)
                return l[0]
        if(l[i] != I)
                return l[i]
        if(a[i] < l(i-1))
                l[i] = i
        else
```

```
                l[i] = l(i-1)
        return l[i]

        //R(i) can be created in a same way

        US()
                for each i from 0 to n-1
                        if(l[i] != s[i])
```

see more

sg · 3 years ago

// step 2(b) of above algo for(i = 0; i min) s = i; } // step 2(c) of above algo for(i; } has a small bug

it should have been
// step 2(b) of above algo
for(i = 0; i min)
s = i;
break;
}
// step 2(c) of above algo
for(i = n -1 ; i >= e+1; --i)
{
if(arr[i] < max)
e = i;
break;
}

Sandeep ➜ sg · 3 years ago

@sg: Thanks for pointing out the bug. We have corrected the original p

Prateek Caire • 3 years ago

Am i missing any case here in solution given below?

```
US()
        max = a[0]
        for each i from 1 to n-1
                if(a[i] >= max)
                        max = a[i]
                else
                        e = i
        min = a[n-1]
        for each i from n-2 to 0
                if(a[i] <= min)
                        min = a[i]
                else
                        s = i
```

Rohit ➦ Prateek Caire • 10 months ago

I don't think anything is missing. It is a nice/better solution. :)

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

Dzmitry Huba • 3 years ago

Folks,

This algorithm has a small issue. Consider the following input: {10, 12, 20, 55,

3, and e = 6, min = 25, max = 40. But once you'll move left boundary to include
noticed in current algorithm.

I think correct algorithm should be:
[sourcecode language="Csharp"]
static Tuple<int, int> FindMinUnsortedSubarray(int[] a)
{
int i = 0, j = a.Length - 1;

for (; i < j && a[i] <= a[i + 1]; i++) ;

if (i == j)
return new Tuple<int, int>(-1, -1);

for (; a[j - 1] <= a[j]; j--) ;

int min = a[++i], max = a[--j];

see more

Sandeep ➦ Dzmitry Huba • 3 years ago

@Dzmitry Huba:

Please take a closer look at algorithm. In step 2(c), e is updated. Also, correct output for your example.

```
 int main()
{
  int arr[] = {10, 12, 20, 55, 25, 40, 23, 31, 35, 50, 60};
  int arr_size = sizeof(arr)/sizeof(arr[0]);
  printUnsorted(arr, arr_size);
  getchar();
  return 0;
}
```

Let me know if I missed something.

Dzmitry Huba Sandeep • 3 years ago

I'm sorry, I missed that while calculating min and max both bou
to the left and min of increasing sequence to the right) are inclu
consideration. Your solution is correct. In my solution there is n
the only values that influence the result are boundaries. Again, \

anirudh • 3 years ago

Small correction is needed here, s=3 and e=7.So we need to find the min,max
to be 25 and 40 respectively. The array specified in the solution includes 35 as

GeeksforGeeks anirudh • 3 years ago

@anirudh: Thanks for pointing this out. There was a typo. We have fixe

lovocas • 3 years ago

I think something may be wrong

Find first element (if there is any) in arr[e+1..n-1] which is smaller than max, c
above example 1, e is changed to 8 (index of 35)

should be
"Find first element arr[xx](if there is any) in arr[e+1..n-1] which is **larger** than n

sry ,my written english is not pretty good ,do you guys know what I mean, isn't

for example :
in example(2)
[10, 12, 20, 30, 25, 40, 32, 31, 35, 50, 60] the result is 3 , 8;
but
if [10, 12, 20, 30, 25, 40, 32, 31, 35, **36** ,50, 60] the wrong result is 3, 8! It's wrc
3,9

Sandeep lovocas • 3 years ago
@lovocas: Please take a closer look at the algo and implementation. It
implementation strictly follow the algorithm, and it prints correct results
36 ,50, 60].

Let me know if you still feel that something is wrong/missing.

lovocas Sandeep • 3 years ago
2(c) Find first element (if there is any) in arr[e+1..n-1] which is s
this element

```
 // step 2(c) of above algo
for( i = e+1; i < n; i++)
{
  if(arr[i] < max)
    e = i;
}
```

actually I just read the Solution description carefully, not pay mu

2(c) should be Find the **last** element ....it may make some

after all ,code is right.

Sandeep ➜ lovocas • 3 years ago

@lovocas:
yes, putting *last* seems to make more sense than *first*.

spgokul • 3 years ago

really nice program and cool answer ... Great

kp101090 • 3 years ago

Very nice program geeks !!!

Subscribe Add Disqus to your site