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

3) Print *s* and *e*.

Implementation:

```c
#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)
    {
      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.

Related Topics:

- Find Union and Intersection of two unsorted arrays
- Pythagorean Triplet in an array
- Maximum profit by buying and selling a share at most twice
- Design a data structure that supports insert, delete, search and getRandom in constant time

- Print missing elements that lie in range 0 – 99
- Iterative Merge Sort
- Group multiple occurrence of array elements ordered by first occurrence
- Given a sorted and rotated array, find if there is a pair with a given sum

Tweet 0 g+1 1

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

79 Comments GeeksforGeeks 1 Login

Recommend 2 Share Sort by Newest

Join the discussion…

Sushmita Das • 2 months ago
binary search for finding elts smaller than min and greater than max...doesnt help in an increasing the efficiency coz asymptotically it is 0(n)..
• Reply • Share ›

mandrake → Sushmita Das • a month ago
Not exactly. Suppose you have two algorithms to solve a particular array problem. First one solves the problem with a 5 passes where as the second one solves with a single pass. Asymptotically both are O(n). Does that mean the second algorithm is not more efficient than first algorithm?
• Reply • Share ›

Saurabh • 2 months ago
why not just simply sort the array and check first index of sorted array mismatching with the original array the only problem is of O(n) extra space
• Reply • Share ›

mudit → Saurabh • 4 days ago
Complexity would be O(nlogn)
• Reply • Share ›

walter • 3 months ago
There's no need to "check if it work then include more element".
Find "s" and "e" as discussed, and do a binary search for "just_smaller_than_A[s+1]" and "just_larger_than_A[e-1]" in A[0 ~ s] and A[e ~ n-1], respectively.
• Reply • Share ›

Kenneth • 6 months ago
http://ideone.com/VeWsyk
1 • Reply • Share ›

payal • 6 months ago

Consider the array: [10, 12, 25, 20, 30, 31, 32, 40, 35, 50, 60]
there exist 2 unsorted subarray : {25,20} and {40,35}
Now the minimum length of unsorted subarray should be : 4
Output of the given and other discussed solutions are : subarray of size 7(index 2-8).

• Reply • Share ›

Ashish Dwivedi payal • 4 months ago

payal u just need one subarray, you are considering two subarrays, u r not supposed to find the separate sub array and sort them individually, rather take one chunk, it could be partially sorted internally

1 • Reply • Share ›

amine • 6 months ago

// simpler O(n) solution in one scan from left to right and another scan from //right to left
int minLengthUnsorted(int A[], int n)
{

int max = A[0];
int right = 0;
for(int i = 1; i<n; ++i="")="" {="" if(a[i]<max)="" right="i;" else="" max="A[i];" }="" int="" min="A[n-1];" int="" left="n-1;" for(="" int="" i="n-2;" i="">=0; --i)
{
if(A[i]>min)
left = i;
else
min = A[i];
}
std::cout<<"from: "<<left<<" to:="" "<<right<<std::endl;="" return="" (right-left+1);="" }="">

• Reply • Share ›

Sachin Hosmani • 6 months ago

Here's a one pass implementation with one extra binary search: http://codepad.org/BRFOIhD4

Hope I didn't miss anything.

2 • Reply • Share ›

Wayne Sachin Hosmani • 2 months ago

I think it's not right for the array {3, 4, 8, 9, 5, 7, 10, 1, 2, 3, 5, 6, 11}, you should also have a minVal to keep track of the min value of the candidate unsorted array, once a[i] < min, need to use bin_search again to get new left position

• Reply • Share ›

Nikhat • 6 months ago

The question says: find *minimum* length subarray, sorting which makes the whole array sorted (sorted array can be an increasing sequence or decreasing one as its not specified).

That means if the array is:

15, 14, 12, 20, 10, 9, 8, 7, 6 then the minimum length subarray should be: 15,14,12,20 (length=4) because sorting them makes the whole array sorted:

20, 15, 14, 12, 10, 9, 8, 7, 6.

But I guess the solution gives the ans: 0 to 8 indices (length=9).

Please let me know if I am missing something.

Thanks!

• Reply • Share ›

Sachin Hosmani Nikhat • 6 months ago

Sorted in *increasing* order.

2 • Reply • Share ›

helper • 6 months ago

very very clean explaination. thanks a lot

• Reply • Share ›

numid • 7 months ago

another solution with two traversals can be :http://ideone.com/2bBF7y

• Reply • Share ›

RB • 8 months ago

Simple solution with two traversals:

http://ideone.com/1KJ0KH

8 • Reply • Share ›

Sriram Ganesh RB • 21 days ago

It doesnt work for example 2..

0, 1, 15, 25, 6, 7, 30, 40, 50..

the correct answer is sort from 2 to 5.
But your code is giving wrong output..

• Reply • Share ›

Itachi Uchiha Sriram Ganesh • 16 days ago [illegible] • 10 days ago

His code gives the same output. 2 to 5. Its correct.

^ | ⌄ • Reply • Share ›

The_Geek ➦ RB • 5 months ago

pretty gud solution !

^ | ⌄ • Reply • Share ›

randomguy202 ➦ The_Geek • 4 months ago

Seems to be working

http://ideone.com/q7XPil

Result: "Sort subarray from 2 to 5"

^ | ⌄ • Reply • Share ›

Aditya Goel ➦ RB • 7 months ago

Amazing!! **@GeeksforGeeks** Please refer to this soln.

2 ^ | ⌄ • Reply • Share ›

don • 9 months ago

#define n 9

main()

{

int a[n]={0, 1, 15, 25, 6, 7, 30, 40, 50};

int s,e,mini,maxi,i,min,max;

min=max=0;

for(i=0;i<n;i++) {="" if(a[i]="">a[i+1])

{

s=i;

break;

}}

see more

1 ^ | ⌄ • Reply • Share ›

viks176 • 9 months ago

i hope this should suffice, plz let me know in case of failure for some test case:

http://ideone.com/6gHRbm

• Reply • Share ›

vipinkaushal • 9 months ago

we can simply trace the whole array twice for left and right index
implementation is here
http://ideone.com/WEX6lz

• Reply • Share ›

Prakhar Gairola • 10 months ago

I think this code is better than above one, only single array scan. If any error pls comment or give suggestions.

http://ideone.com/c8vD2l

• Reply • Share ›

sudheer470 Prakhar Gairola • 9 months ago

How ur Code is better than author's code ?(i'm not arguing)
Even he
using more number of for loops, he scanning the whole array for only
once (for all for loops). for 2b and 2c we can do it by using binary
search instead of linear search.
and in ur code, u are using more number of comparisons.
And scanning the array for 2 times i guess.

• Reply • Share ›

guest Prakhar Gairola • 10 months ago

your program give W.A for

int arr[] = {10, 12, 20, 30, 35, 40, 42, 43, 45, 50, 60,10};

• Reply • Share ›

Prakhar Gairola guest • 10 months ago

now I made this change, I think it should work now for all cases. btw thanx for providing feedback.

• Reply • Share ›

Vãîbhåv Joshî • 10 months ago

java code

http://ideone.com/ODm63l

• Reply • Share ›

Guest • a year ago

how the above algorithm works for
100 200 300 250 260 270 9 10 11
I think it does not work for this. Please help
1 • Reply • Share ›

Vãîbhåv Joshî Guest • 10 months ago
http://ideone.com/ODm63l
• Reply • Share ›

typing.. Guest • 10 months ago
I think It gives correct answer 0 to 8, post your answer if it is anything else.
• Reply • Share ›

Vivek • a year ago
for step 2 we can do a binary search in a[0..s-1] for the ceil value of min.

we can do a binary search in a[e+1....n-1] for the floor value of max.
• Reply • Share ›

arpit1990 Vivek • a year ago
It doesn't matter because in worst case, our step-1 (a) or(b) ill give us O(n) and after you ill apply the binary search that ill not improve the time complexity at all... i hope u get this...
• Reply • Share ›

Vivek arpit1990 • a year ago
i know time complexity won't improve but but by applying binary search u ll surely improve the running time as the constant factors would become less
• Reply • Share ›

wrestler • a year ago
A better solution with no over checking. Find start from right to left of the array. And end from st towards end.

def findSubArray(arr):

st=len(arr)-1

i_min=st

for i in range(len(arr)-1,-1,-1):

if arr[i_min]>arr[i]:

i_min=i

```
else:

st=i

end=st
```

see more

2 • Reply • Share ›

Venu Gopal • a year 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, binary search, or stack. All the steps are explained in detail with help of comments for better understanding of reader..
http://ideone.com/q7X3ES
• Reply • Share ›

groomnestle • a year ago
A (nlgn) solution is simpler: make a copy of array A into B, sort B using quicksort/mergesort (nlgn), then find the first leftmost and rightmost elements that differ in A and B. The minimum subarray would be the array between these 2 elements.
• Reply • Share ›

Castle Age ➦ groomnestle • a year 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
2 • Reply • Share ›

Shikha Gupta • 2 years ago
can sum1 pls give an eg to explain when part 2b of above solution arises??
2 • Reply • Share ›

Pushkar ➦ Shikha Gupta • 2 years ago
Part 2(b) of the solution arises with the input {10,12,30,11,40,32,31,35,50,60} Here 25 is replaced by 11 in this example. so step 2(a) find min for [30,11,40,32,31] as 11. So step 2(b) changes the value of s to 1 which is the index of 12. Hope you got it!!!
1 • Reply • Share ›

Shikha Gupta ➦ Pushkar • 2 years ago
thanks ..how do v copy paste code from vc++ here..wenever i paste it changes to something else...can u tel sum good material to understand asymptotic

notations and dynamic programming

Reply • Share ›

Shikha Gupta • 2 years 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[i]<0)="" {=""
minind="i;" break;="" }="" }="" 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,maxind);

}
```

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)

Reply • Share ›

exor • 2 years ago

The last part where you scan the insertion point for min and max can be done in logn using binary search (because the starting and ending sequences are sorted)...

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

1 • Reply • Share ›

kuldeepshandilya • 2 years ago

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

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

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

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 sum of sizes of all these sub arrays.

Reply • Share ›

Ankit Gupta • 2 years 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;.
}
```

Reply • Share ›

hunter • 2 years 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.............
```

Reply • Share ›

hunter ➦ hunter • 2 years ago

sorry it won't work............

• Reply • Share ›

Priyanshu • 2 years 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 code) */
#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

• Reply • Share ›

me.abhinav • 2 years ago

A different O(n) solution:
Let answer = (start, end), where 'start' = first index of the subarray to be sorted, and 'end' = last index of the subarray to be sorted.
1) Scan the array a[0..n-1] from left-to-right. Let current index be 'i'. How can we say that 'i' is the required first index (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 > i in the completely sorted array. Thus 'start' = i and we need to sort the array from 'start' upto some index 'end' (which is calculated in next step). Print 'start' and break from loop.

2) Scan the array a[0..n-1] from right-to-left. Let current index be 'i'. Following the above logic, if a[i] < max(a[0..i]) then this a[i] must be present at some other index < i in the completely sorted array. Thus 'end' = i and we need to sort the array from 'start' (calculated in step-(1)) upto 'end'. Print 'end' and break from loop.

With some pre-processing (which takes O(n) time), we can determine min(a[i..n-1]) and max(a[0..i]) in constant time (see code).

• Reply • Share ›

Load more comments

Subscribe   Add Disqus to your site   Privacy

-
-
-
-
    - Interview Experiences
    - Advanced Data Structures
    - Dynamic Programming
    - Greedy Algorithms
    - Backtracking
    - Pattern Searching
    - Divide & Conquer
    - Mathematical Algorithms
    - Recursion
    - 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 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
- Follow @GeeksforGeeks

- ## Recent Comments

    - lt_k

      i need help for coding this function in java...

      Java Programming Language · 1 hour ago
    - Piyush

      What is the purpose of else if (recStack[*i])...

      Detect Cycle in a Directed Graph · 1 hour ago
    - Andy Toh

      My compile-time solution, which agrees with the...

      Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago
    - lucy

      because we first fill zero in first col and...

      Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago
    - lucy

      @GeeksforGeeks i don't n know what is this long...

      Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 2 hours ago
    - manish

Because TAN is not a subsequence of RANT. ANT...

Given two strings, find if first string is a subsequence of second · 2 hours ago

-