GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

- Home
- Algorithms
- DS
- GATE
- Interview Corner
- Q&A
- C
- C++
- Java
- Books
- Contribute
- Ask a Q
- About

Array
Bit Magic
C/C++
Articles
GFacts
Linked List
MCQ
Misc
Output
String
Tree
Graph

Find the minimum element in a sorted and rotated array

A sorted array is rotated at some unknown point, find the minimum element in it.

Following solution assumes that all elements are distinct.

Examples

```
Input: {5, 6, 1, 2, 3, 4}
Output: 1

Input: {1, 2, 3, 4}
Output: 1

Input: {2, 1}
```

```
Output: 1
```

A simple solution is to traverse the complete array and find minimum. This solution requires [Tex]\Theta(n)[/Tex] time.
We can do it in O(Logn) using Binary Search. If we take a closer look at above examples, we can easily figure out following pattern: The minimum element is the only element whose previous element is greater than it. If there is no such element, then there is no rotation and first element is the minimum element. Therefore, we do binary search for an element which is smaller than the previous element. We strongly recommend you to try it yourself before seeing the following C implementation.

```
// C program to find minimum element in a sorted and rotated array
#include <stdio.h>

int findMin(int arr[], int low, int high)
{
    // This condition is needed to handle the case when array is not
    // rotated at all
    if (high < low)  return arr[0];

    // If there is only one element left
    if (high == low) return arr[low];

    // Find mid
    int mid = low + (high - low)/2; /*(low + high)/2;*/

    // Check if element (mid+1) is minimum element. Consider
    // the cases like {3, 4, 5, 1, 2}
    if (mid < high && arr[mid+1] < arr[mid])
       return arr[mid+1];

    // Check if mid itself is minimum element
    if (mid > low && arr[mid] < arr[mid - 1])
       return arr[mid];

    // Decide whether we need to go to left half or right half
    if (arr[high] > arr[mid])
       return findMin(arr, low, mid-1);
    return findMin(arr, mid+1, high);
}

// Driver program to test above functions
int main()
{
    int arr1[] =  {5, 6, 1, 2, 3, 4};
    int n1 = sizeof(arr1)/sizeof(arr1[0]);
    printf("The minimum element is %d\n", findMin(arr1, 0, n1-1));

    int arr2[] =  {1, 2, 3, 4};
    int n2 = sizeof(arr2)/sizeof(arr2[0]);
    printf("The minimum element is %d\n", findMin(arr2, 0, n2-1));

    int arr3[] =  {1};
    int n3 = sizeof(arr3)/sizeof(arr3[0]);
```

```
    printf("The minimum element is %d\n", findMin(arr3, 0, n3-1));

    int arr4[] =  {1, 2};
    int n4 = sizeof(arr4)/sizeof(arr4[0]);
    printf("The minimum element is %d\n", findMin(arr4, 0, n4-1));

    int arr5[] =  {2, 1};
    int n5 = sizeof(arr5)/sizeof(arr5[0]);
    printf("The minimum element is %d\n", findMin(arr5, 0, n5-1));

    int arr6[] =  {5, 6, 7, 1, 2, 3, 4};
    int n6 = sizeof(arr6)/sizeof(arr6[0]);
    printf("The minimum element is %d\n", findMin(arr6, 0, n6-1));

    int arr7[] =  {1, 2, 3, 4, 5, 6, 7};
    int n7 = sizeof(arr7)/sizeof(arr7[0]);
    printf("The minimum element is %d\n", findMin(arr7, 0, n7-1));

    int arr8[] =  {2, 3, 4, 5, 6, 7, 8, 1};
    int n8 = sizeof(arr8)/sizeof(arr8[0]);
    printf("The minimum element is %d\n", findMin(arr8, 0, n8-1));

    int arr9[] =  {3, 4, 5, 1, 2};
    int n9 = sizeof(arr9)/sizeof(arr9[0]);
    printf("The minimum element is %d\n", findMin(arr9, 0, n9-1));

    return 0;
}
```

Output:

```
The minimum element is 1
The minimum element is 1
The minimum element is 1
The minimum element is 1
The minimum element is 1
The minimum element is 1
The minimum element is 1
The minimum element is 1
The minimum element is 1
```

How to handle duplicates?
It turned out that duplicates can't be handled in O(Logn) time in all cases. Thanks to Amit Jain for inputs. The special cases that cause problems are like {2, 2, 2, 2, 2, 2, 2, 2, 0, 1, 1, 2} and {2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2}. It doesn't look possible to go to left half or right half by doing constant number of comparisons at the middle. Following is an implementation that handles duplicates. It may become O(n) in worst case though.

```
// C program to find minimum element in a sorted and rotated array
#include <stdio.h>

int min(int x, int y) { return (x < y)? x :y; }

// The function that handles duplicates.  It can be O(n) in worst case.
```

```
int findMin(int arr[], int low, int high)
{
    // This condition is needed to handle the case when array is not
    // rotated at all
    if (high < low)  return arr[0];

    // If there is only one element left
    if (high == low) return arr[low];

    // Find mid
    int mid = low + (high - low)/2; /*(low + high)/2;*/

    // Check if element (mid+1) is minimum element. Consider
    // the cases like {1, 1, 0, 1}
    if (mid < high && arr[mid+1] < arr[mid])
       return arr[mid+1];

    // This case causes O(n) time
    if (arr[low] == arr[mid] && arr[high] == arr[mid])
        return min(findMin(arr, low, mid-1), findMin(arr, mid+1, high));

    // Check if mid itself is minimum element
    if (mid > low && arr[mid] < arr[mid - 1])
       return arr[mid];

    // Decide whether we need to go to left half or right half
    if (arr[high] > arr[mid])
       return findMin(arr, low, mid-1);
    return findMin(arr, mid+1, high);
}
```

```
// Driver program to test above functions
int main()
{
    int arr1[] =  {5, 6, 1, 2, 3, 4};
    int n1 = sizeof(arr1)/sizeof(arr1[0]);
    printf("The minimum element is %d\n", findMin(arr1, 0, n1-1));

    int arr2[] =  {1, 1, 0, 1};
    int n2 = sizeof(arr2)/sizeof(arr2[0]);
    printf("The minimum element is %d\n", findMin(arr2, 0, n2-1));

    int arr3[] =  {1, 1, 2, 2, 3};
    int n3 = sizeof(arr3)/sizeof(arr3[0]);
    printf("The minimum element is %d\n", findMin(arr3, 0, n3-1));

    int arr4[] =  {3, 3, 3, 4, 4, 4, 4, 5, 3, 3};
    int n4 = sizeof(arr4)/sizeof(arr4[0]);
    printf("The minimum element is %d\n", findMin(arr4, 0, n4-1));

    int arr5[] =  {2, 2, 2, 2, 2, 2, 2, 2, 0, 1, 1, 2};
    int n5 = sizeof(arr5)/sizeof(arr5[0]);
    printf("The minimum element is %d\n", findMin(arr5, 0, n5-1));
```

```
    int arr6[] =  {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1};
    int n6 = sizeof(arr6)/sizeof(arr6[0]);
    printf("The minimum element is %d\n", findMin(arr6, 0, n6-1));

    int arr7[] =  {2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2};
    int n7 = sizeof(arr7)/sizeof(arr7[0]);
    printf("The minimum element is %d\n", findMin(arr7, 0, n7-1));

    return 0;
}
```

Output:

```
The minimum element is 1
The minimum element is 0
The minimum element is 1
The minimum element is 3
The minimum element is 0
The minimum element is 1
The minimum element is 0
```

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

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

Tags: Divide and Conquer

recursive solution for both (idea is that half of the array is always sorted so we can skip that part and go for other half).
http://ideone.com/2ZNALa

• Reply • Share ›

Ravi Kumar • 2 months ago

solution of handling duplicates will give wrong answer at array [1,3,3]. Line which may give O(n) time should be written as

if (arr[low] == arr[mid] || arr[high] == arr[mid])
return min(findMin(arr,low,mid-1),findMin(arr,mid+high));

instead of

if (arr[low] == arr[mid] && arr[high] == arr[mid])
return min(findMin(arr, low, mid-1), findMin(arr, mid+1, high));

Note: OR is used instead of AND.

1 • Reply • Share ›

legalroot • 2 months ago

insead of this " return min(findMin(arr, low, mid-1), findMin(arr, mid+1, high)"
i think its better to use return findmin(arr,low+1,high-1); correct me if i am wrong

• Reply • Share ›

Vall • 3 months ago

Why not this?

package interview;

import java.io.*;
import java.util.*;

class RotatedArray
{

static void printmin(int a[])
{
HashSet s = new HashSet();
for(int i=0;i<a.length;i++) s.add(a[i]);="" system.out.println(collections.min(s));="" }="" public="" static="" void="" main(string="" args[])="" {="" int="" a[]="{3,4,1,2};" int="" len="a.length;" int="" mid="len/2;" int="" flag="0;" printmin(a);="" easy="" implementation="" to="" find="" lowest="" element,="" but="" not="" the="" right="" solution="" for="" this="" problem="" if(a[0]<a[len-1])="" {="" system.out.println("array="" is="" not="" rotated.="" so="" lowest="" element="" is="" :="" "="" +="" a[0]);="" flag="1;" }="" if(flag!="1)" {="" if(a[0]<a[mid])="" {="" for(int=""

see more

• Reply • Share ›

neer1304 • 3 months ago

Simpler version for both the cases :-
http://ideone.com/LaJnN5

2 • Reply • Share ›

Jerry Goyal ➚ neer1304 • 10 days ago

fails for 3,2,1,4,5,6,7

• Reply • Share ›

neer1304 ➚ Jerry Goyal • 10 days ago

Its not a sorted and rotated array. The correct example can be 8,9,1,4,5,6,7

• Reply • Share ›

Jerry Goyal ➚ neer1304 • 10 days ago

oh yes,you are right.

• Reply • Share ›

jaswant • 5 months ago

Algorithm mentioned above gives wrong output for this input { 6,9,15,12,1,3 }. correct output is 1 but this gives 12.

• Reply • Share ›

Sandeep ➚ jaswant • 5 months ago

That isn't a sorted array.

4 • Reply • Share ›

imps • 5 months ago

To be honest I don't get why all the complications with checking the neighboring element and stuff. It's not very neat.

Here's something that's straightforward:

```
int getMinimum(vector<int> input){
if(!input.size())
return -1;
if(input.size()==1){
return input[0];
}

int start = 0;
int end = input.size()-1;
int mid;
```

```
while(start <= end){
mid = (start+end)/2;
if(start==end){
return input[start];
```

see more

• Reply • Share ›

Random • 6 months ago

Try this one it will work for all the cases when input contains no duplicate

```
int min_in_sorted_rotated(int *a, int l, int h )
{

if( l == h )
return a[l];
// calculate mid element.

int mid = ( l + h) / 2;

if((a[mid-1]< a[mid] && a[mid] > a[mid +1]) && (l < mid < h))
return a[mid + 1];
else if(( a[mid] < a[mid -1] && a[mid] < a[mid + 1] )&& (l < mid <h)) return="" a[mid="" ];="" 
else="" if(="" a[mid]="">= a[l])
{
if( a[mid] > a[l] && a[mid] > a[h])
return min_in_sorted_rotated(a,mid+1,h);
else
return a[l];
```

see more

• Reply • Share ›

Random • 6 months ago

Above algorithm is fails when input array in reverse order.
Example input : 10 9 8 7 6 5 4 3 2 1
Output : 5

3 • Reply • Share ›

Salman Ahmed ➦ Random • 5 months ago

I don't understand how that is rotated array

3 • Reply • Share ›

Vamshikrishna • 6 months ago

Ravi is right the condition should be if(arr[high]>=arr[mid]) instead of if(arr[high]>arr[mid])

• Reply • Share ›

Ravi Kasha • 6 months ago

The solution that handles duplicates doesn't work for the case {1,3,3}.

The correct solution can be obtained by changing

if (arr[high] > arr[mid])

return findMin(arr, low, mid-1);

To

if (arr[high] >= arr[mid])
return findMin(arr, low, mid-1);

• Reply • Share ›

Han Wang • 7 months ago

if (high < low) return arr[0];

I think this part is wrong.

It should be

if(num[high] > num[low])
return num[0];

Since the array is not sorted.

1 • Reply • Share ›

changhaz • 7 months ago

My compact solution that passed Leetcode OJ:

details can be found at
http://changhaz.wordpress.com/...

int findMin(vector<int> &num) {
int start=0,end=num.size()-1;

while (start<end) {="" if="" (num[start]<num[end])="" return="" num[start];="" int="" mid="" (start+end)/2;" if="" (num[mid]="">=num[start]) {
start = mid+1;
} else {
end = mid;
}
}

return num[start];
}

}

1 • Reply • Share ›

amateur • 7 months ago

in "return min(findMin(arr, low, mid-1), findMin(arr, mid+1, high));" isn't the first call to findMin redundant since arr[low] & arr[mid] are same all elements within low and min are same since array is sorted and also, since arr[mid] & arr[high] are same the min element would lie between mid and high since array is sorted?

• Reply • Share ›

ashish ➦ amateur • 5 months ago

array is sorted And rotated , not sorted

1 • Reply • Share ›

paramvir • 8 months ago

less complicated implementation of above problem. Note: It do not handles duplicates.

```
int findMin(int arr[],int s,int e)
{
int md = (s+e) >> 1;
if ( arr[s] <= arr[e] )
// normal case
{
return arr[s];
//return findMin(arr, md+1, e);
}
else if ( arr[s] > arr[md] )
// to the left
{
return findMin(arr, s,md);
}
else if ( arr [s] <= arr[md] )
{
return findMin(arr,md+1,e);
}
}
```

• Reply • Share ›

sumeetmi2 • 8 months ago

My implementation in java.. O(log n) fixed for all cases decreasing/increasing order sorted or not.. The drawback is even if its well sorted it will go through logn split

http://ideone.com/fork/G3zMIb

• Reply • Share ›

abhishekgupta76 • 9 months ago

The Function for non-duplicate answer gives wrong result for the following input.
{8,7,6,5,4,3,2,1,10,9}

Reply • Share ›

krishna • 10 months ago

if we are doing this in O(n) time what is point to use recursion which uses more stack space , why cant we use linear scan in O(1) space

Reply • Share ›

Pankaj Pal ➚ krishna • 9 months ago

It will be O(n) in worst case .. specially the cases like 2,2,2,0,1,2
For all other cases, it will be O(log n)

Reply • Share ›

ANA • 10 months ago

http://ideone.com/Q97Bgd

Reply • Share ›

Guest • 10 months ago

is there any need to have this statement in first algorithm

if (high < low) return arr[0];

because i think this condition will never satisfy because we have high==low so at that point itself it will written a[low]

plz help..am i missing something???

Reply • Share ›

parijat ➚ Guest • 9 months ago

it should be..if(high<=low)return arr[low];

Reply • Share ›

np ➚ Guest • 10 months ago

yes this condition is required consider array 1,2 in this case

mid becomes 0 and

findMin(arr, low, mid-1); will get executed

so low=0 and high =-1
so runtime error will ocuur

Reply • Share ›

Lovlean Arora • 10 months ago

int arr7[] = {3,3,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2};

duplicate handling logic still fails for this case... plz check

1 • Reply • Share ›

parijat ➚ Lovlean Arora • 9 months ago

yeah correct....it wont do even for {5,4,3,0,2,2,2,2,2}

• Reply • Share ›

Ravi Kasha ➚ parijat • 6 months ago

It is not the rotation of a sorted array. Check Out.

• Reply • Share ›

Saurabh • a year ago

Simple Implementation for sorted rotated array having no duplicates: (O(log n))

http://ideone.com/1uciN0

Please comment if it is wrong.

• Reply • Share ›

Arvind kumar • a year ago

it will give wrong output for the case whwn all element of the array are same
like {2,2,2,2,2,2,2,2};

• Reply • Share ›

suku • a year ago

{2,1,0,8,7,6,5,4,3}; gives wrong output

• Reply • Share ›

asdaa ➚ suku • 10 months ago

coz ur input is wrong

4 • Reply • Share ›

himanshu dagar • a year ago

This code can not handle the elements in descending order(that is in the category of sorted)

1 • Reply • Share ›

reshma • a year ago

why (mid<high)..checking ..it="" is="" always="" true="" na..="" tell="" me="" if="" i="" am="" incorrect..="">

1 • Reply • Share ›

himanshu dagar ➚ reshma • a year ago

suppose our mid is last element at some time .So to prevent the checking of that element with the next element which doesn't exist ,we have to check that first . Another wise Ur code can give unexpected result. We should remain in array bounds na.

1 ^ | v • Reply • Share ›

Brave Heart • a year ago

The recursive function presented is overly complicated.
Here's a simple solution I coded: http://hastebin.com/hocuranebo...

^ | v • Reply • Share ›

Naveed • a year ago

public class minimumElementInTheSortedRotatedArray {

static int a[] = {8,9,10,1,1,2,2,3,7};
static boolean sorted = true;
public static void main(String[] args) {
for(int i=0;i<a.length-1;i++) {="" if(a[i]="">a[i+1]) {
System.out.println(a[i+1]);
sorted = false;
break;
}
}
if(sorted) {
System.out.println(a[0]);
}
}

}

^ | v • Reply • Share ›

vrg • a year ago

This Question is asked many times in the comment section but not answered
// This condition is needed to handle the case when array is not rotated at //all

if (high < low) return arr[0];

When will the above statement get executed?

5 ^ | v • Reply • Share ›

pulkit mehra • a year ago

In the 2nd program handling repetitive case, you should change the following line:-

if (arr[high] > arr[mid])

to

if (arr[high] >= arr[mid])

else it won't work for i/p such as {3,4,1,2,2,2,2,2,2,2,2}

Musa Paljoš • a year ago

This is the iterative solution for this problem. Please correct me If I'm wrong? Could we optimise this?

```
public static int minInRotadedArray(int[] arr) { // Find the minimum element in a sorted and rotated array
int l = 0;
int r = arr.length - 1;
int i;
while (r >= l) {
i = l + (r - l) / 2;
if (arr[l] < arr[i] && arr[r] > arr[i]) {
return l;
} else if (arr[l] > arr[i] && arr[r] < arr[i]) {
return r;
} else if (arr[l] > arr[i] && arr[r] > arr[i]) {
r = i;
} else if (arr[l] < arr[i] && arr[r] < arr[i]) {
l = i;
} else {
return arr[i] > arr[r] ? r : i;
}
}
return -1;
}
```


Pankkaj Yadav • a year ago

why u r trying to follow a long way.... I have another idea....I am giving my logic ....please check the syntax accordingly....

```
function(int a[])

{
static int i=0;
if (a[i]>a[i+1])
{
i++;
function(a[]);
```

```
function(a[]);
}
else
printf("%d minimum number in present in array is ", a[i]);
}
```

• Reply • Share ›

sijayaraman ➚ Pankkaj Yadav • a year ago

its is not working for the input {5,6,1,2,3,4}

• Reply • Share ›

bani • a year ago

when the array is reverse sorted , method 1 gives wrong answer...
test case where code is flawed :-
arr[]={6,5,4,3,2,1}
kindly look into this

• Reply • Share ›

sijayaraman ➚ bani • a year ago

the question is for sorted and rotated array. i don't think your input is correct.
suppose let say input = { 1,2,3,4,5,6} , if you rotate the array in clock or anticlockwise 5 times then the output will be like this {2,3,4,5,6,1} and {6,1,2,3,4,5}. i dont see you input is match with clock or anticlockwise rotation.

@GeeksForGeeks Could you please confirm?.

1 • Reply • Share ›

Shreyas • a year ago

@GeeksforGeeks method where duplicates are handled.... gives wrong answer for the array 9,10,1,2,2,2,2,2,2,2... kindly check....

• Reply • Share ›

iyer ➚ Shreyas • 4 months ago

yes.. you are right
i think the condition should be (arr[low]==arr[mid] || arr[high]==arr[mid])
i mean instead of && we should use ||
@GeeeksforGeeks please correct this..

• Reply • Share ›

Load more comments

- 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

- Nikhil kumar

  public class...

  Print missing elements that lie in range 0 – 99 · 5 minutes ago

- Ashish Aggarwal

  Try Data Structures and Algorithms Made Easy -...

  Algorithms · 27 minutes ago

- Vlad

  Thanks. Very interesting lectures.

  Expected Number of Trials until Success · 1 hour ago

- cfh

  My implementation which prints the index of the...

  Longest Even Length Substring such that Sum of First and Second Half is same · 1 hour ago

- Gaurav pruthi

  forgot to see that part ;)

  Bloomberg Interview | Set 1 (Phone Interview) · 2 hours ago

- saeid aslami

  thanks

  Greedy Algorithms | Set 7 (Dijkstra's shortest path algorithm) · 2 hours ago