



AfterAcademy



OFFER

Android Online Course by
MindOrks

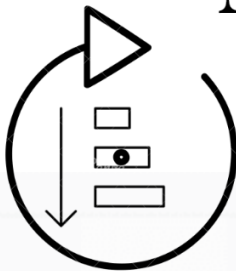
Start your career in Android
Development. Learn by doing real
projects.

ENROLL
NOW



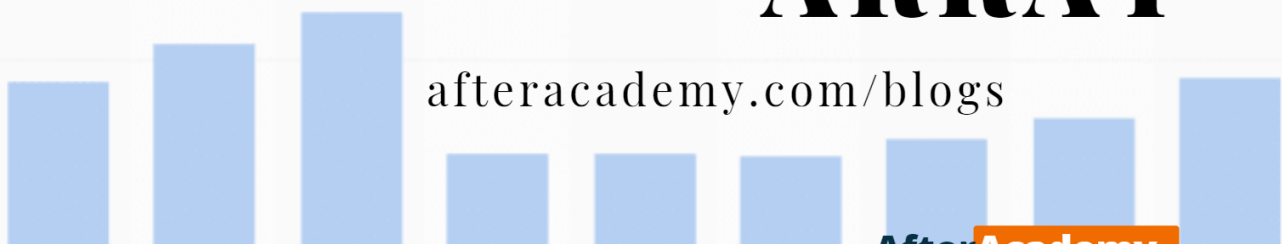
Admin AfterAcademy
16 Jan 2020

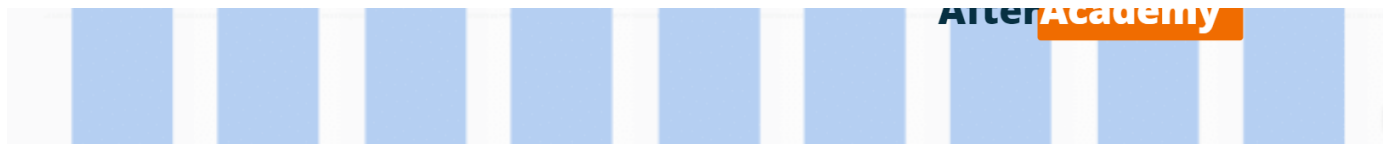
Find Minimum in a Sorted and Rotated array



MINIMUM IN A SORTED AND ROTATED ARRAY

afteracademy.com/blogs





Understanding The Problem

Problem Description: You have a **sorted and rotated** array where elements are sorted and rotated circularly. Write a program to find the minimum element in the array.

- Suppose the elements in array are [1, 2, 3, 4, 5, 6, 7], then rotating by three times array becomes [4, 5, 6, 7, 1, 2, 3].
- You won't be provided with number times array is rotated.
- The array will not contain duplicates.

Examples

Input: array[] = [4, 5, 6, 7, 1, 2, 3]

Output: 1

Input: array[] = [8, 9, 4, 5, 6, 7]

Output: 4

Input: array[] = [3, 4, 5, 6, 7]

Output: 3

Input: array[] = [3, 2]

Output: 2

Solutions

We will be discussing two solutions to this problem:

1. **Brute force approach (Linear Search):** Traverse the complete array and find the minimum.
2. **An approach similar to Binary Search:** Compare mid with first and last element of the subarray and move accordingly.

1. Brute force approach(Linear Search)

The idea is to traverse the array from the first element to the last element of the array and maintain a minimum.

Solution steps

- Create a min variable and initialize it with INT_MAX
- Iterate over the array and update the min variable with the minimum of min and each element of the array.

Pseudo Code

```
int findMin(int array[], int size)
{
    int minimum = INT_MAX
    for(int i=0 to size)
        minimum = min(minimum, array[i])
    return minimum
}
```

Complexity Analysis

Time Complexity: $O(n)$

Space Complexity: $O(1)$

Critical Ideas to Think

- Why did we initialize the minimum with INT_MAX?
- Can we use the property of sorted and rotated array to think of a better solution to reduce the time complexity?

2. An approach similar to Binary Search

If we look closely then you will find that there are at most three cases that could be encountered.

- **Case 1:** The leftmost value is less than the rightmost value in the array. This means that the array is not rotated.
Example: [1 2 3 4 5 6 7 8 9]
- **Case 2:** The value in the middle of the array is greater than the leftmost and rightmost values in the array.
Example: [4 5 6 7 8 9 1 2 3]
- **Case 3:** The value in the middle of the array is less than the leftmost and rightmost values in the array.
Example: [7 8 9 1 2 3 4 5 6]

if we have case 1 then we just return the leftmost value in the array. If we have case 2, we will try to find the minimum on the right side of the array. If we have case 3, we will try to find the minimum to the left side of the array.

Solution steps

1. Initialize a **left** and **right** variable with 0 and size-1 of the array.

2. Iterate until $\text{left} < \text{right}$

- If the value at left is less than the value at right then return value at left.
- Initialize **mid** = $(\text{left} + \text{right}) / 2$
- Now compare if the value at **mid** is greater than the value at **right**, then we search the minimum in the left part of the array. Update **left** = **mid** + 1
- Otherwise, we search in the right part of the array. Update **right** = **mid**

Pseudo Code - Iterative

```
int findMin(int array[], int size)
{
    int left = 0
    int right = size - 1
    while(left < right)
    {
        if(array[left] < array[right]) // case 1
            return array[left]
        int mid = left + (right-left)/2
        if(array[mid] > array[right])
            left = mid + 1           // case 2
        else
            right = mid              // case 3
    }
    return array[left]
}
```

Pseudo Code - Recursive

```
// Helper function
int bsearch(int array[], int left, int right)
{
```

```
    if(left < right)
    {
        if(array[left] < array[right])           // case 1
            return array[left]
        int mid = left + (right - left)/2
        if(array[mid] > array[right])
            return bsearch(array, mid + 1, right) // case 2
        else
            return bsearch(array, left, mid)      // case 3
    }
    return array[left]
}
// Driver function
int findMin(int array[], int size)
{
    return bsearch(array, 0, size - 1)
}
```

Complexity Analysis

The above approach looks similar to the idea of the binary search.

Time Complexity = $O(\log n)$

Space Complexity = $O(1)$

Critical Ideas to Think

- Do all the three cases discussed above are handled by the algorithm?
- Is it necessary to return array[left] in the end?
- Why did we initialize mid with $(\text{left} + (\text{right} - \text{left})/2)$ instead of $(\text{left} + \text{right})/2$?
- Will this approach work if we have duplicates in the array?
- Can we solve the problem in $O(\log n)$ if we have duplicates in the input array?

- Can we think of some different approach to solve this problem?
- Prepare a list of problems where we can use the idea similar to a binary search for the solution.



NEW

Android App Development Online Course by MindOrks

Start your career in Android Development. Learn by doing real projects.

[CHECK NOW](#)

Comparison of different solution

Approach	Time Complexity	Space Complexity
Linear Search	$O(n)$	$O(1)$
Binary Search	$O(\log n)$	$O(1)$

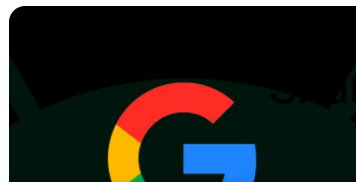
Similar Problems to solve

- Search an element in a circular sorted array
- Find the first or last occurrence of a given number in a sorted array
- Count occurrences of a number in a sorted array with duplicates
- Find smallest missing element from a sorted array
- Find Floor and Ceil of a number in a sorted array
- Search in a nearly sorted array
- Find the peak element in an array

Please comment down below if you have alternative approaches or find an error/bug in the above approaches.

Happy Coding, Enjoy Algorithms!


AfterAcademy Data Structure And Algorithms Online Course—Admissions Open



NEW

Share this blog and spread the knowledge

Google Android Developer

 SHARE ON FACEBOOK SHARE ON TWITTER SHARE ON LINKEDIN SHARE ON TELEGRAM SHARE ON REDDIT SHARE ON WHATSAPP

Recommended for You



Average of levels of Binary tree

afteracademy.com

Average of Levels in Binary Tree

Given a binary tree, write a program to return the average value of the nodes on each level in the form of an array. The range of the node's value is in the range of 32-bit signed integer.



Admin AfterAcademy
4 Nov 2020



Recursive Insertion Sort

afteracademy.com

Recursive Insertion Sort

Write a program for the recursive implementation of Insertion Sort. Insertion Sort is used to sort a given array. This problem will sharpen your recursion skills.



Admin AfterAcademy
22 Sep 2020



Admin AfterAcademy
15 Sep 2020

When to Convert a 2-D DP array to 1-D DP array And How



Admin AfterAcademy
12 Sep 2020

Merge sort in linked list

1-D DP array And How

afteracademy.com

When to Convert a 2-D DP array to 1-D DP array And How?

In which situation 2 dimensional DP can be dropped to 1 dimension? Is there any principle or regular pattern? This is a very important question when it comes to optimization of DP arrays. Let's find out.

Sort List - Merge Sort

Sort a linked list using Merge Sort. This is a very famous interview problem that demonstrates the concept of recursion. This problem is quite similar to Merge Sort in Arrays.

Interview question

Knight on chessboard



Asked in  Goldman Sachs

afteracademy.com

Knight on chessboard

Given a square chessboard of A x B size, the position of Knight (C, D) and the position of a target (E, F) is given. Write a program to find out the minimum steps a Knight will take to reach the target position. This problem is a good example of BFS algorithm.



Admin AfterAcademy
7 Sep 2020



Largest Element in an Array

afteracademy.com

Largest Element In An Array

Given an array `arr[]` of size `n`, write a program to find the largest element in it. To find the largest element we can just traverse the array in one pass and find the largest element by maintaining a max variable.



Admin AfterAcademy
3 Sep 2020

Our Learners Work At



AfterAcademy

Stay up to date. Follow us on



© Copyright 2019

MindOrks Nextgen Private Limited
Gurgaon, Haryana, India
+91-8287460223

About Us

MindOrks
Amit Shekhar
Janishar Ali

Quick Links

[Contact Us](#)
[Privacy Policy](#)
[Terms And Conditions](#)
[Cookie Policy](#)

Free Resources

[Publication](#)
[Medium](#)
[Video Lessons](#)
[Open Source](#)