Find the minimum element in a sorted and rotated array

Difficulty Level : Medium • Last Updated : 05 Jul, 2022

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

Recommended Practice

Minimum element in a sorted and rotated array

Try It!

A simple solution is to traverse the complete array and find a minimum. This solution requires O(n) time.
We can do it in O(Logn) using Binary Search. If we take a closer look at the above examples, we can easily figure out the following pattern:

- The minimum element is the only element whose previous is greater than it. If there is no previous element, then there is no rotation (the first element is minimum). We check this condition for the middle element by comparing it with (mid-1)'th and (mid+1)'th elements.
- If the minimum element is not at the middle (neither mid nor mid + 1), then the minimum element lies in either the left half or right half.
  1. If the middle element is smaller than the last element, then the minimum element lies in the left half
  2. Else minimum element lies in the right half.

We strongly recommend you to try it yourself before seeing the following implementation.


Complete Interview Preparation - Self Paced
By Sandeep Jain
Beginner to Advance Level

Find 360 solution to all of your interview woes. Learn 4 years' worth of programming knowledge in just 6 months and ace coding interviews at top tech companies.
Explore Now

C++

```
// C++ program to find minimum
// element in a sorted and rotated array
#include <bits/stdc++.h>
using namespace std;

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]);
    cout << "The minimum element is " << findMin(arr1, 0, n1-1) << endl;

    int arr2[] = {1, 2, 3, 4};
    int n2 = sizeof(arr2)/sizeof(arr2[0]);
    cout << "The minimum element is " << findMin(arr2, 0, n2-1) << endl;

    int arr3[] = {1};
    int n3 = sizeof(arr3)/sizeof(arr3[0]);
    cout<<"The minimum element is "<<findMin(arr3, 0, n3-1)<<endl;

    int arr4[] = {1, 2};
    int n4 = sizeof(arr4)/sizeof(arr4[0]);
    cout<<"The minimum element is "<<findMin(arr4, 0, n4-1)<<endl;

    int arr5[] = {2, 1};
    int n5 = sizeof(arr5)/sizeof(arr5[0]);
    cout<<"The minimum element is "<<findMin(arr5, 0, n5-1)<<endl;

    int arr6[] = {5, 6, 7, 1, 2, 3, 4};
    int n6 = sizeof(arr6)/sizeof(arr6[0]);
    cout<<"The minimum element is "<<findMin(arr6, 0, n6-1)<<endl;

    int arr7[] = {1, 2, 3, 4, 5, 6, 7};
    int n7 = sizeof(arr7)/sizeof(arr7[0]);
    cout << "The minimum element is " << findMin(arr7, 0, n7-1) << endl;

    int arr8[] = {2, 3, 4, 5, 6, 7, 8, 1};
    int n8 = sizeof(arr8)/sizeof(arr8[0]);
    cout << "The minimum element is " << findMin(arr8, 0, n8-1) << endl;

    int arr9[] = {3, 4, 5, 1, 2};
    int n9 = sizeof(arr9)/sizeof(arr9[0]);
    cout << "The minimum element is " << findMin(arr9, 0, n9-1) << endl;

    return 0;
}
```

```
// This is code is contributed by rathbhupendra
```

C

```c
// 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;
}
```

Java

```
// Java program to find minimum element in a sorted and rotated array
import java.util.*;
import java.lang.*;
import java.io.*;

class Minimum
{
    static 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
    public static void main (String[] args)
    {
        int arr1[] =  {5, 6, 1, 2, 3, 4};
        int n1 = arr1.length;
        System.out.println("The minimum element is "+ findMin(arr1, 0, n1

        int arr2[] =  {1, 2, 3, 4};
        int n2 = arr2.length;
        System.out.println("The minimum element is "+ findMin(arr2, 0, n2

        int arr3[] =  {1};
        int n3 = arr3.length;
        System.out.println("The minimum element is "+ findMin(arr3, 0, n3

        int arr4[] =  {1, 2};
        int n4 = arr4.length;
        System.out.println("The minimum element is "+ findMin(arr4, 0, n4

        int arr5[] =  {2, 1};
        int n5 = arr5.length;
        System.out.println("The minimum element is "+ findMin(arr5, 0, n5
```

```
        int arr6[] =  {5, 6, 7, 1, 2, 3, 4};
        int n6 = arr6.length;
        System.out.println("The minimum element is "+ findMin(arr6, 0, n6

        int arr7[] =  {1, 2, 3, 4, 5, 6, 7};
        int n7 = arr7.length;
        System.out.println("The minimum element is "+ findMin(arr7, 0, n7

        int arr8[] =  {2, 3, 4, 5, 6, 7, 8, 1};
        int n8 = arr8.length;
        System.out.println("The minimum element is "+ findMin(arr8, 0, n8

        int arr9[] =  {3, 4, 5, 1, 2};
        int n9 = arr9.length;
        System.out.println("The minimum element is "+ findMin(arr9, 0, n9
    }
}
```

Python3

```
# Python program to find minimum element
# in a sorted and rotated array

def findMin(arr, low, 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
    mid = int((low + high)/2)

    # Check if element (mid+1) is minimum element. Consider
    # the cases like [3, 4, 5, 1, 2]
    if mid < high and arr[mid+1] < arr[mid]:
        return arr[mid+1]

    # Check if mid itself is minimum element
    if mid > low and 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
arr1 = [5, 6, 1, 2, 3, 4]
n1 = len(arr1)
print("The minimum element is " + str(findMin(arr1, 0, n1-1)))

arr2 = [1, 2, 3, 4]
n2 = len(arr2)
print("The minimum element is " + str(findMin(arr2, 0, n2-1)))

arr3 = [1]
n3 = len(arr3)
print("The minimum element is " + str(findMin(arr3, 0, n3-1)))

arr4 = [1, 2]
n4 = len(arr4)
print("The minimum element is " + str(findMin(arr4, 0, n4-1)))

arr5 = [2, 1]
n5 = len(arr5)
print("The minimum element is " + str(findMin(arr5, 0, n5-1)))

arr6 = [5, 6, 7, 1, 2, 3, 4]
n6 = len(arr6)
print("The minimum element is " + str(findMin(arr6, 0, n6-1)))

arr7 = [1, 2, 3, 4, 5, 6, 7]
n7 = len(arr7)
print("The minimum element is " + str(findMin(arr7, 0, n7-1)))

arr8 = [2, 3, 4, 5, 6, 7, 8, 1]
n8 = len(arr8)
print("The minimum element is " + str(findMin(arr8, 0, n8-1)))

arr9 = [3, 4, 5, 1, 2]
n9 = len(arr9)
print("The minimum element is " + str(findMin(arr9, 0, n9-1)))

# This code is contributed by Pratik Chhajer
```

C#

```
// C# program to find minimum element
// in a sorted and rotated array
using System;

class Minimum {

    static 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
        // (low + high)/2
        int mid = low + (high - low) / 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
    public static void Main()
    {
        int[] arr1 = { 5, 6, 1, 2, 3, 4 };
```

```
            int n1 = arr1.Length;
            Console.WriteLine("The minimum element is " +
                              findMin(arr1, 0, n1 - 1));

            int[] arr2 = { 1, 2, 3, 4 };
            int n2 = arr2.Length;
            Console.WriteLine("The minimum element is " +
                              findMin(arr2, 0, n2 - 1));

            int[] arr3 = { 1 };
            int n3 = arr3.Length;
            Console.WriteLine("The minimum element is " +
                              findMin(arr3, 0, n3 - 1));

            int[] arr4 = { 1, 2 };
            int n4 = arr4.Length;
            Console.WriteLine("The minimum element is " +
                              findMin(arr4, 0, n4 - 1));

            int[] arr5 = { 2, 1 };
            int n5 = arr5.Length;
            Console.WriteLine("The minimum element is " +
                              findMin(arr5, 0, n5 - 1));

            int[] arr6 = { 5, 6, 7, 1, 2, 3, 4 };
            int n6 = arr6.Length;
            Console.WriteLine("The minimum element is " +
                              findMin(arr6, 0, n1 - 1));

            int[] arr7 = { 1, 2, 3, 4, 5, 6, 7 };
            int n7 = arr7.Length;
            Console.WriteLine("The minimum element is " +
                              findMin(arr7, 0, n7 - 1));

            int[] arr8 = { 2, 3, 4, 5, 6, 7, 8, 1 };
            int n8 = arr8.Length;
            Console.WriteLine("The minimum element is " +
                              findMin(arr8, 0, n8 - 1));

            int[] arr9 = { 3, 4, 5, 1, 2 };
            int n9 = arr9.Length;
            Console.WriteLine("The minimum element is " +
                              findMin(arr9, 0, n9 - 1));
        }
    }
```

```
// This code is contributed by vt_m.
```

PHP

```
<?php
// PHP program to find minimum
// element in a sorted and
// rotated array

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

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

    // Find mid
    $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 Code
$arr1 = array(5, 6, 1, 2, 3, 4);
$n1 = sizeof($arr1);
echo "The minimum element is " .
    findMin($arr1, 0, $n1 - 1) . "\n";

$arr2 = array(1, 2, 3, 4);
$n2 = sizeof($arr2);
echo "The minimum element is " .
    findMin($arr2, 0, $n2 - 1) . "\n";

$arr3 = array(1);
$n3 = sizeof($arr3);
echo "The minimum element is " .
    findMin($arr3, 0, $n3 - 1) . "\n";

$arr4 = array(1, 2);
$n4 = sizeof($arr4);
echo "The minimum element is " .
    findMin($arr4, 0, $n4 - 1) . "\n";

$arr5 = array(2, 1);
$n5 = sizeof($arr5);
echo "The minimum element is " .
    findMin($arr5, 0, $n5 - 1) . "\n";

$arr6 = array(5, 6, 7, 1, 2, 3, 4);
$n6 = sizeof($arr6);
echo "The minimum element is " .
    findMin($arr6, 0, $n6 - 1) . "\n";

$arr7 = array(1, 2, 3, 4, 5, 6, 7);
$n7 = sizeof($arr7);
echo "The minimum element is " .
    findMin($arr7, 0, $n7 - 1) . "\n";

$arr8 = array(2, 3, 4, 5, 6, 7, 8, 1);
$n8 = sizeof($arr8);
echo "The minimum element is " .
    findMin($arr8, 0, $n8 - 1) . "\n";
```

```
$arr9 = array(3, 4, 5, 1, 2);
$n9 = sizeof($arr9);
echo "The minimum element is " .
    findMin($arr9, 0, $n9 - 1) . "\n";

// This code is contributed by ChitraNayal
?>
```

Javascript

```
<script>

// Javascript program to find minimum element in a sorted and rotated arr


    function findMin(arr,low,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
        let mid =low + Math.floor((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
let arr1=[5, 6, 1, 2, 3, 4];
let n1 = arr1.length;
document.write("The minimum element is "+ findMin(arr1, 0, n1-1)+"<br

let arr2=[1, 2, 3, 4];
let n2 = arr2.length;
document.write("The minimum element is "+ findMin(arr2, 0, n2-1)+"<br

let arr3=[1];
let n3 = arr3.length;
document.write("The minimum element is "+ findMin(arr3, 0, n3-1)+"<br

let arr4=[1,2];
let n4 = arr4.length;
document.write("The minimum element is "+ findMin(arr4, 0, n4-1)+"<br

let arr5=[2,1];
let n5 = arr5.length;
document.write("The minimum element is "+ findMin(arr5, 0, n5-1)+"<br

let arr6=[5, 6, 7, 1, 2, 3, 4];
let n6 = arr6.length;
document.write("The minimum element is "+ findMin(arr6, 0, n6-1)+"<br

let arr7=[1, 2, 3, 4, 5, 6, 7];
let n7 = arr7.length;
document.write("The minimum element is "+ findMin(arr7, 0, n7-1)+"<br

let arr8=[2, 3, 4, 5, 6, 7, 8, 1];
let n8 = arr8.length;
document.write("The minimum element is "+ findMin(arr8, 0, n8-1)+"<br

let arr9=[3, 4, 5, 1, 2];
let n9 = arr9.length;
document.write("The minimum element is "+ findMin(arr9, 0, n9-1)+"<br

// This code is contributed by avanitrachhadiya2155
```

```
</script>
```

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
```

Time Complexity: O(n)

Auxiliary Space: O(1)

How to handle duplicates?

The above approach in the worst case(If all the elements are the same) takes O(N).

Below is the code to handle duplicates in O(log n) time.

C++

```
// C++ program to find minimum element in a sorted
// and rotated array containing duplicate elements.
#include <bits/stdc++.h>
using namespace std;

// Function to find minimum element
int findMin(int arr[], int low, int high)
{
    while (low < high) {
        int mid = low + (high - low) / 2;
        if (arr[mid] == arr[high])
            high--;
        else if (arr[mid] > arr[high])
            low = mid + 1;
        else
            high = mid;
```

```
    }
    return arr[high];
}

// Driver code
int main()
{
    int arr1[] = { 5, 6, 1, 2, 3, 4 };
    int n1 = sizeof(arr1) / sizeof(arr1[0]);
    cout << "The minimum element is "
         << findMin(arr1, 0, n1 - 1) << endl;

    int arr2[] = { 1, 2, 3, 4 };
    int n2 = sizeof(arr2) / sizeof(arr2[0]);
    cout << "The minimum element is "
         << findMin(arr2, 0, n2 - 1) << endl;

    int arr3[] = { 1 };
    int n3 = sizeof(arr3) / sizeof(arr3[0]);
    cout << "The minimum element is "
         << findMin(arr3, 0, n3 - 1) << endl;

    int arr4[] = { 1, 2 };
    int n4 = sizeof(arr4) / sizeof(arr4[0]);
    cout << "The minimum element is "
         << findMin(arr4, 0, n4 - 1) << endl;

    int arr5[] = { 2, 1 };
    int n5 = sizeof(arr5) / sizeof(arr5[0]);
    cout << "The minimum element is "
         << findMin(arr5, 0, n5 - 1) << endl;

    int arr6[] = { 5, 6, 7, 1, 2, 3, 4 };
    int n6 = sizeof(arr6) / sizeof(arr6[0]);
    cout << "The minimum element is "
         << findMin(arr6, 0, n6 - 1) << endl;

    int arr7[] = { 1, 2, 3, 4, 5, 6, 7 };
    int n7 = sizeof(arr7) / sizeof(arr7[0]);
    cout << "The minimum element is "
         << findMin(arr7, 0, n7 - 1) << endl;

    int arr8[] = { 2, 3, 4, 5, 6, 7, 8, 1 };
    int n8 = sizeof(arr8) / sizeof(arr8[0]);
    cout << "The minimum element is "
```

```
        << findMin(arr8, 0, n8 - 1) << endl;

    int arr9[] = { 3, 4, 5, 1, 2 };
    int n9 = sizeof(arr9) / sizeof(arr9[0]);
    cout << "The minimum element is "
        << findMin(arr9, 0, n9 - 1) << endl;

    return 0;
}

// This code is contributed by Sania Kumari Gupta
// (kriSania804)
```

C

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

// Function to find minimum element
int findMin(int arr[], int low, int high)
{
    while (low < high) {
        int mid = low + (high - low) / 2;
        if (arr[mid] == arr[high])
            high--;
        else if (arr[mid] > arr[high])
            low = mid + 1;
        else
            high = mid;
    }
    return arr[high];
}

// Driver code
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;
    }

    // This code is contributed by Sania Kumari Gupta
    // (kriSania804)
```

Java

```java
// Java program to find minimum element
// in a sorted and rotated array containing
// duplicate elements.
import java.util.*;
import java.lang.*;

class GFG{

// Function to find minimum element
public static int findMin(int arr[],
                          int low, int high)
{
    while(low < high)
    {
        int mid = low + (high - low) / 2;
        if (arr[mid] == arr[high])
            high--;

        else if(arr[mid] > arr[high])
            low = mid + 1;
        else
            high = mid;
    }
    return arr[high];
}

// Driver code
public static void main(String args[])
{
    int arr1[] = { 5, 6, 1, 2, 3, 4 };
    int n1 = arr1.length;
    System.out.println("The minimum element is " +
                       findMin(arr1, 0, n1 - 1));

    int arr2[] = { 1, 2, 3, 4 };
    int n2 = arr2.length;
    System.out.println("The minimum element is " +
                       findMin(arr2, 0, n2 - 1));

    int arr3[] = {1};
    int n3 = arr3.length;
    System.out.println("The minimum element is " +
```

```
                        findMin(arr3, 0, n3 - 1));

        int arr4[] = { 1, 2 };
        int n4 = arr4.length;
        System.out.println("The minimum element is " +
                        findMin(arr4, 0, n4 - 1));

        int arr5[] = { 2, 1 };
        int n5 = arr5.length;
        System.out.println("The minimum element is " +
                        findMin(arr5, 0, n5 - 1));

        int arr6[] = { 5, 6, 7, 1, 2, 3, 4 };
        int n6 = arr6.length;
        System.out.println("The minimum element is " +
                        findMin(arr6, 0, n6 - 1));

        int arr7[] = { 1, 2, 3, 4, 5, 6, 7 };
        int n7 = arr7.length;
        System.out.println("The minimum element is " +
                        findMin(arr7, 0, n7 - 1));

        int arr8[] = { 2, 3, 4, 5, 6, 7, 8, 1 };
        int n8 = arr8.length;
        System.out.println("The minimum element is " +
                        findMin(arr8, 0, n8 - 1));

        int arr9[] = { 3, 4, 5, 1, 2 };
        int n9 = arr9.length;
        System.out.println("The minimum element is " +
                        findMin(arr9, 0, n9 - 1));
    }
    }

    // This is code is contributed by SoumikMondal
```

Python3

```
# Python3 program to find
# minimum element in a sorted
# and rotated array containing
# duplicate elements.
```

```
# Function to find minimum element
def findMin(arr, low, high):

    while (low < high):
        mid = low + (high - low) // 2;

        if (arr[mid] == arr[high]):
            high -= 1;
        elif (arr[mid] > arr[high]):
            low = mid + 1;
        else:
            high = mid;
    return arr[high];

# Driver code
if __name__ == '__main__':

    arr1 = [5, 6, 1, 2, 3, 4];
    n1 = len(arr1);
    print("The minimum element is ",
          findMin(arr1, 0, n1 - 1));

    arr2 = [1, 2, 3, 4];
    n2 = len(arr2);
    print("The minimum element is ",
          findMin(arr2, 0, n2 - 1));

    arr3 = [1];
    n3 = len(arr3);
    print("The minimum element is ",
          findMin(arr3, 0, n3 - 1));

    arr4 = [1, 2];
    n4 = len(arr4);
    print("The minimum element is ",
          findMin(arr4, 0, n4 - 1));

    arr5 = [2, 1];
    n5 = len(arr5);
    print("The minimum element is ",
          findMin(arr5, 0, n5 - 1));

    arr6 = [5, 6, 7, 1, 2, 3, 4];
    n6 = len(arr6);
    print("The minimum element is ",
```

```
            findMin(arr6, 0, n6 - 1));

        arr7 = [1, 2, 3, 4, 5, 6, 7];
        n7 = len(arr7);
        print("The minimum element is ",
            findMin(arr7, 0, n7 - 1));

        arr8 = [2, 3, 4, 5, 6, 7, 8, 1];
        n8 = len(arr8);
        print("The minimum element is ",
            findMin(arr8, 0, n8 - 1));

        arr9 = [3, 4, 5, 1, 2];
        n9 = len(arr9);
        print("The minimum element is ",
            findMin(arr9, 0, n9 - 1));

# This code is contributed by Princi Singh
```

C#

```
// C# program to find minimum element
// in a sorted and rotated array
// containing duplicate elements.
using System;

class GFG{

// Function to find minimum element
public static int findMin(int []arr, int low,
                          int high)
{
    while (low < high)
    {
        int mid = low + (high - low) / 2;
        if (arr[mid] == arr[high])
            high--;

        else if (arr[mid] > arr[high])
            low = mid + 1;
        else
            high = mid;
    }
    return arr[high];
```

```
}

// Driver code
public static void Main(String []args)
{
    int []arr1 = { 5, 6, 1, 2, 3, 4 };
    int n1 = arr1.Length;
    Console.WriteLine("The minimum element is " +
                      findMin(arr1, 0, n1 - 1));

    int []arr2 = { 1, 2, 3, 4 };
    int n2 = arr2.Length;
    Console.WriteLine("The minimum element is " +
                      findMin(arr2, 0, n2 - 1));

    int []arr3 = {1};
    int n3 = arr3.Length;
    Console.WriteLine("The minimum element is " +
                      findMin(arr3, 0, n3 - 1));

    int []arr4 = { 1, 2 };
    int n4 = arr4.Length;
    Console.WriteLine("The minimum element is " +
                      findMin(arr4, 0, n4 - 1));

    int []arr5 = { 2, 1 };
    int n5 = arr5.Length;
    Console.WriteLine("The minimum element is " +
                      findMin(arr5, 0, n5 - 1));

    int []arr6 = { 5, 6, 7, 1, 2, 3, 4 };
    int n6 = arr6.Length;
    Console.WriteLine("The minimum element is " +
                      findMin(arr6, 0, n6 - 1));

    int []arr7 = { 1, 2, 3, 4, 5, 6, 7 };
    int n7 = arr7.Length;
    Console.WriteLine("The minimum element is " +
                      findMin(arr7, 0, n7 - 1));

    int []arr8 = { 2, 3, 4, 5, 6, 7, 8, 1 };
    int n8 = arr8.Length;
    Console.WriteLine("The minimum element is " +
                      findMin(arr8, 0, n8 - 1));
```

```
    int []arr9 = { 3, 4, 5, 1, 2 };
    int n9 = arr9.Length;
    Console.WriteLine("The minimum element is " +
                      findMin(arr9, 0, n9 - 1));
}
}

// This code is contributed by Amit Katiyar
```

Javascript

```
<script>
// JavaScript program to find minimum element in a sorted
// and rotated array containing duplicate elements.

// Function to find minimum element
function findMin(arr, low, high)
{
    while(low < high)
    {
        let mid = Math.floor(low + (high - low)/2);
        if (arr[mid] == arr[high])
            high--;
        else if(arr[mid] > arr[high])
            low = mid + 1;
        else
            high = mid;
    }
    return arr[high];
}

var arr1 = [5, 6, 1, 2, 3, 4];
var n1 = arr1.length;
document.write("The minimum element is " + findMin(arr1, 0, n1-1) + "<br>

var arr2 = [1, 2, 3, 4];
var n2 = arr2.length;
document.write("The minimum element is " + findMin(arr2, 0, n2-1) + "<br>

var arr3 = [1];
var n3 = arr3.length;
document.write("The minimum element is " + findMin(arr3, 0, n3-1) + "<br>

var arr4 = [1, 2];
var n4 = arr4.length;
```

```
document.write("The minimum element is " + findMin(arr4, 0, n4-1) + "<br>

var arr5 = [2, 1];
var n5 = arr5.length;
document.write("The minimum element is " + findMin(arr5, 0, n5-1) + "<br>

var arr6 = [5, 6, 7, 1, 2, 3, 4];
var n6 = arr6.length;
document.write("The minimum element is " + findMin(arr6, 0, n6-1) + "<br>

var arr7 = [1, 2, 3, 4, 5, 6, 7];
var n7 = arr7.length;
document.write("The minimum element is " + findMin(arr7, 0, n7-1) + "<br>

var arr8 = [2, 3, 4, 5, 6, 7, 8, 1];
var n8 = arr8.length;
document.write("The minimum element is " + findMin(arr8, 0, n8-1) + "<br>

var arr9 = [3, 4, 5, 1, 2];
var n9 = arr9.length;
document.write("The minimum element is " + findMin(arr9, 0, n9-1) + "<br>

// This code is contributed by probinsah
</script>
```

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
```

Time Complexity: O(Log n)

Auxiliary Space: O(1)

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.

Like 137

Previous

Find the Rotation Count in Rotated Sorted array

Next

Print left rotation of array in O(n) time and O(1) space

RECOMMENDED ARTICLES

Page : 1 2 3

array
15, Nov 21

03 **Javascript Program for Search an element in a sorted and rotated array**
15, Nov 21

04 **C++ Program for Search an element in a sorted and rotated array**
15, Nov 21

element in a sorted and rotated array
15, Nov 21

07 **Php Program for Search an element in a sorted and rotated array**
15, Nov 21

08 **C Program for Search an element in a sorted and rotated array**
15, Nov 21

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy | Normal | Medium | Hard | Expert

Improved By : ukasp, Sar_123, rathbhupendra, shreyashagrawal, SaptakathaAdak, SoumikMondal, anjali_1102, amit143katiyar, princi singh, avanitrachhadiya2155, probinsah, amartyaghoshgfg, simmytarika5, _shinchancode, krisania804

Article Tags : Adobe, Amazon, Binary Search, Microsoft, Morgan Stanley, Samsung, Snapdeal, Times Internet, Arrays, Divide and Conquer, Searching

Practice Tags : Morgan Stanley, Amazon, Microsoft, Samsung, Snapdeal, Adobe, Times Internet, Arrays, Searching, Divide and Conquer, Binary Search

Improve Article | Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks
A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company

About Us

Careers

In Media

Contact Us

Privacy Policy

Copyright Policy

Learn

Algorithms

Data Structures

SDE Cheat Sheet

Machine learning

CS Subjects

Video Tutorials

Courses

News

Top News

Technology

Work & Career

Business

Finance

Lifestyle

Knowledge

Languages

Python

Java

CPP

Golang

C#

SQL

Kotlin

Web Development

Web Tutorials

Django Tutorial

HTML

JavaScript

Contribute

Write an Article

Improve an Article

Pick Topics to Write

Write Interview Experience

NodeJS

@geeksforgeeks , Some rights reserved

Do Not Sell My Personal Information