

# Problem 153: Find Minimum in Rotated Sorted Array

## Problem Information

Difficulty: **Medium**

Acceptance Rate: 0.00%

Paid Only: No

## Problem Description

Suppose an array of length

$n$

sorted in ascending order is

rotated

between

1

and

$n$

times. For example, the array

`nums = [0,1,2,4,5,6,7]`

might become:

`[4,5,6,7,0,1,2]`

if it was rotated

4

times.

[0,1,2,4,5,6,7]

if it was rotated

7

times.

Notice that

rotating

an array

[a[0], a[1], a[2], ..., a[n-1]]

1 time results in the array

[a[n-1], a[0], a[1], a[2], ..., a[n-2]]

.

Given the sorted rotated array

nums

of

unique

elements, return

the minimum element of this array

.

You must write an algorithm that runs in

$O(\log n)$  time

.

Example 1:

Input:

nums = [3,4,5,1,2]

Output:

1

Explanation:

The original array was [1,2,3,4,5] rotated 3 times.

Example 2:

Input:

nums = [4,5,6,7,0,1,2]

Output:

0

Explanation:

The original array was [0,1,2,4,5,6,7] and it was rotated 4 times.

Example 3:

Input:

```
nums = [11,13,15,17]
```

Output:

11

Explanation:

The original array was [11,13,15,17] and it was rotated 4 times.

Constraints:

```
n == nums.length
```

```
1 <= n <= 5000
```

```
-5000 <= nums[i] <= 5000
```

All the integers of

nums

are

unique

.

nums

is sorted and rotated between

1

and

n

times.

## Code Snippets

### C++:

```
class Solution {
public:
    int findMin(vector<int>& nums) {

    }
};
```

### Java:

```
class Solution {
    public int findMin(int[] nums) {

    }
}
```

### Python3:

```
class Solution:
    def findMin(self, nums: List[int]) -> int:
```

### Python:

```
class Solution(object):
    def findMin(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
```

### JavaScript:

```
/**
 * @param {number[]} nums
 * @return {number}
 */
```

```
var findMin = function(nums) {  
  
};
```

### TypeScript:

```
function findMin(nums: number[]): number {  
  
};
```

### C#:

```
public class Solution {  
    public int FindMin(int[] nums) {  
  
    }  
}
```

### C:

```
int findMin(int* nums, int numsSize) {  
  
}
```

### Go:

```
func findMin(nums []int) int {  
  
}
```

### Kotlin:

```
class Solution {  
    fun findMin(nums: IntArray): Int {  
  
    }  
}
```

### Swift:

```
class Solution {  
    func findMin(_ nums: [Int]) -> Int {
```

```
}  
}
```

### Rust:

```
impl Solution {  
    pub fn find_min(nums: Vec<i32>) -> i32 {  
  
    }  
}
```

### Ruby:

```
# @param {Integer[]} nums  
# @return {Integer}  
def find_min(nums)  
  
end
```

### PHP:

```
class Solution {  
  
    /**  
     * @param Integer[] $nums  
     * @return Integer  
     */  
    function findMin($nums) {  
  
    }  
}
```

### Dart:

```
class Solution {  
    int findMin(List<int> nums) {  
  
    }  
}
```

### Scala:

```

object Solution {
  def findMin(nums: Array[Int]): Int = {

  }
}

```

### Elixir:

```

defmodule Solution do
  @spec find_min(nums :: [integer]) :: integer
  def find_min(nums) do

  end
end

```

### Erlang:

```

-spec find_min(Nums :: [integer()]) -> integer().
find_min(Nums) ->
.

```

### Racket:

```

(define/contract (find-min nums)
  (-> (listof exact-integer?) exact-integer?)
  )

```

## Solutions

### C++ Solution:

```

/*
 * Problem: Find Minimum in Rotated Sorted Array
 * Difficulty: Medium
 * Tags: array, sort, search
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

```



```

class Solution {
public:
    int findMin(vector<int>& nums) {

    }

};

```

### Java Solution:

```

/**
 * Problem: Find Minimum in Rotated Sorted Array
 * Difficulty: Medium
 * Tags: array, sort, search
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
    public int findMin(int[] nums) {

    }

}

```

### Python3 Solution:

```

"""
Problem: Find Minimum in Rotated Sorted Array
Difficulty: Medium
Tags: array, sort, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
    def findMin(self, nums: List[int]) -> int:
        # TODO: Implement optimized solution
        pass

```

## Python Solution:

```
class Solution(object):
    def findMin(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
```

## JavaScript Solution:

```
/**
 * Problem: Find Minimum in Rotated Sorted Array
 * Difficulty: Medium
 * Tags: array, sort, search
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * @param {number[]} nums
 * @return {number}
 */
var findMin = function(nums) {

};
```

## TypeScript Solution:

```
/**
 * Problem: Find Minimum in Rotated Sorted Array
 * Difficulty: Medium
 * Tags: array, sort, search
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

function findMin(nums: number[]): number {
```

```
};
```

### C# Solution:

```
/*
 * Problem: Find Minimum in Rotated Sorted Array
 * Difficulty: Medium
 * Tags: array, sort, search
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

public class Solution {
    public int FindMin(int[] nums) {

    }
}
```

### C Solution:

```
/*
 * Problem: Find Minimum in Rotated Sorted Array
 * Difficulty: Medium
 * Tags: array, sort, search
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

int findMin(int* nums, int numsSize) {

}
```

### Go Solution:

```
// Problem: Find Minimum in Rotated Sorted Array
// Difficulty: Medium
```

```
// Tags: array, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func findMin(nums []int) int {

}
```

### Kotlin Solution:

```
class Solution {
    fun findMin(nums: IntArray): Int {

    }
}
```

### Swift Solution:

```
class Solution {
    func findMin(_ nums: [Int]) -> Int {

    }
}
```

### Rust Solution:

```
// Problem: Find Minimum in Rotated Sorted Array
// Difficulty: Medium
// Tags: array, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
    pub fn find_min(nums: Vec<i32>) -> i32 {

    }
}
```

### Ruby Solution:

```
# @param {Integer[]} nums
# @return {Integer}
def find_min(nums)

end
```

### PHP Solution:

```
class Solution {

    /**
     * @param Integer[] $nums
     * @return Integer
     */
    function findMin($nums) {

    }

}
```

### Dart Solution:

```
class Solution {
  int findMin(List<int> nums) {

  }
}
```

### Scala Solution:

```
object Solution {
  def findMin(nums: Array[Int]): Int = {

  }
}
```

### Elixir Solution:

```
defmodule Solution do
  @spec find_min(nums :: [integer]) :: integer
  def find_min(nums) do
```

```
end  
end
```

### **Erlang Solution:**

```
-spec find_min(Nums :: [integer()]) -> integer().  
find_min(Nums) ->  
.
```

### **Racket Solution:**

```
(define/contract (find-min nums)  
  (-> (listof exact-integer?) exact-integer?)  
  )
```