Previous Next

153. Find Minimum in Rotated Sorted Array

Sept. 4, 2018 | 111.3K views

Average Rating: 4.40 (88 votes)

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand.

(i.e., `[0,1,2,4,5,6,7]` might become `[4,5,6,7,0,1,2]`).

Find the minimum element.

You may assume no duplicate exists in the array.

Example 1:

```
Input: [3,4,5,1,2]
Output: 1
```

Example 2:

```
Input: [4,5,6,7,0,1,2]
Output: 0
```

Solution

Approach 1: Binary Search

Intuition

A very brute way of solving this question is to search the entire array and find the minimum element. The time complexity for that would be $O(N)$ given that `N` is the size of the array.

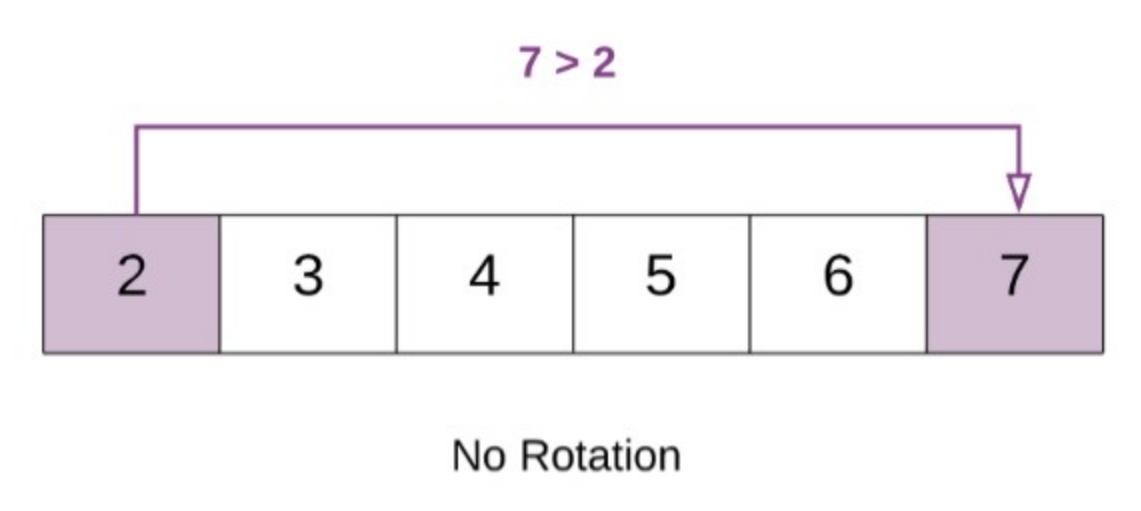
A very cool way of solving this problem is using the `Binary Search` algorithm. In binary search we find out the mid point and decide to either search on the left or right depending on some condition.

Since the given array is sorted, we can make use of binary search. However, the array is rotated. So simply applying the binary search won't work here.

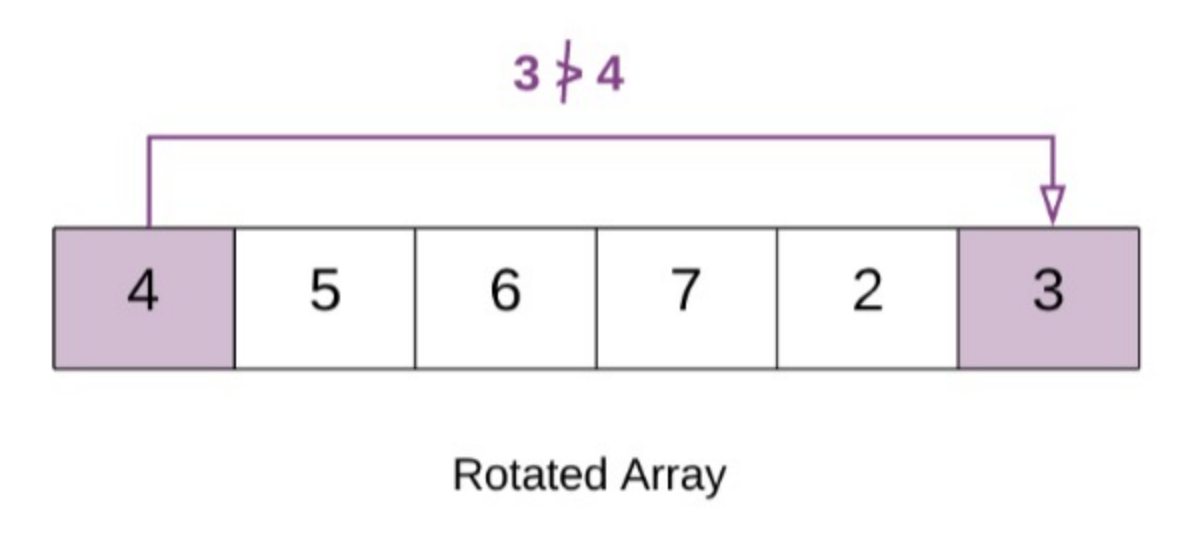
In this question we would essentially apply a modified version of binary search where the `condition` that decides the search direction would be different than in a standard binary search.

We want to find the smallest element in a rotated sorted array. What if the array is not rotated? How do we check that?

If the array is not rotated and the array is in ascending order, then `last element > first element`.

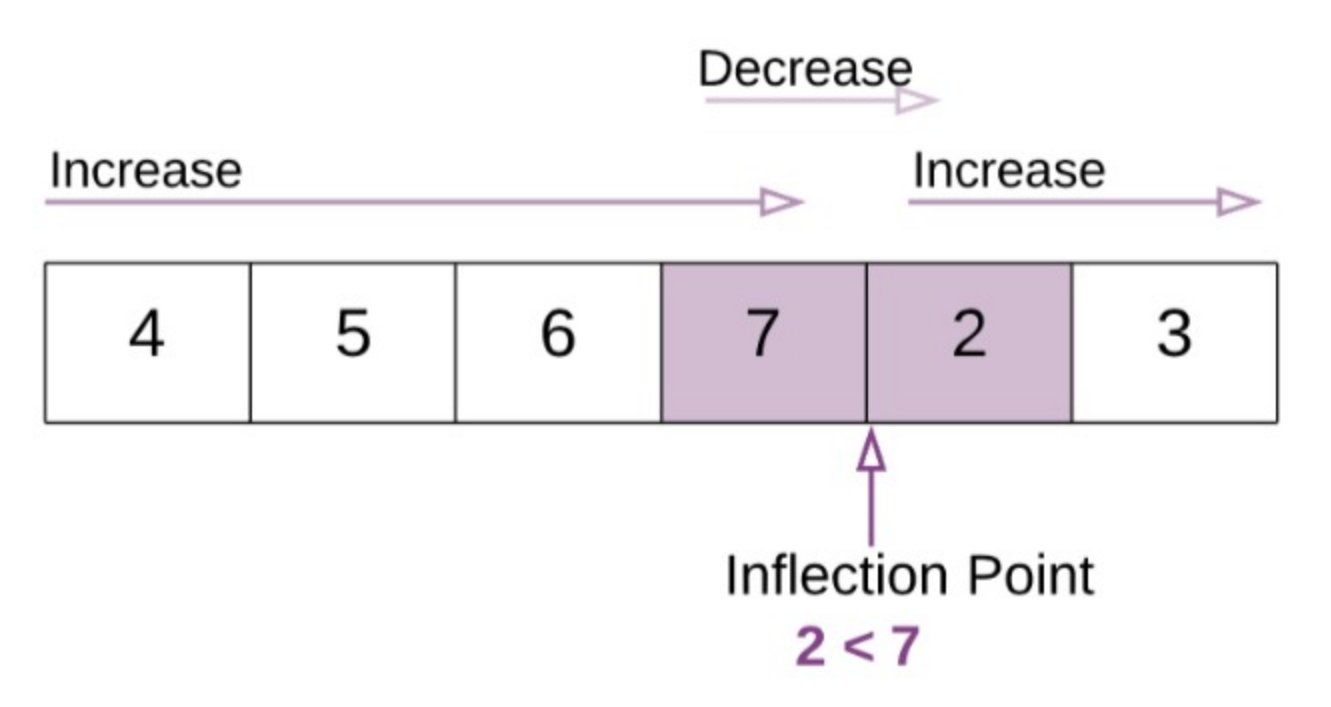


In the above example `7 > 2`. This means that the array is still sorted and has no rotation.



In the above example `3 < 4`. Hence the array is rotated. This happens because the array was initially `[2, 3 ,4 ,5 ,6 ,7]`. But after the rotation the smaller elements `[2,3]` go at the back. i.e. [4, 5, 6, 7, `2, 3]`. Because of this the first element `[4]` in the rotated array becomes greater than the last element.

This means there is a point in the array at which you would notice a change. This is the point which would help us in this question. We call this the `Inflection Point`.

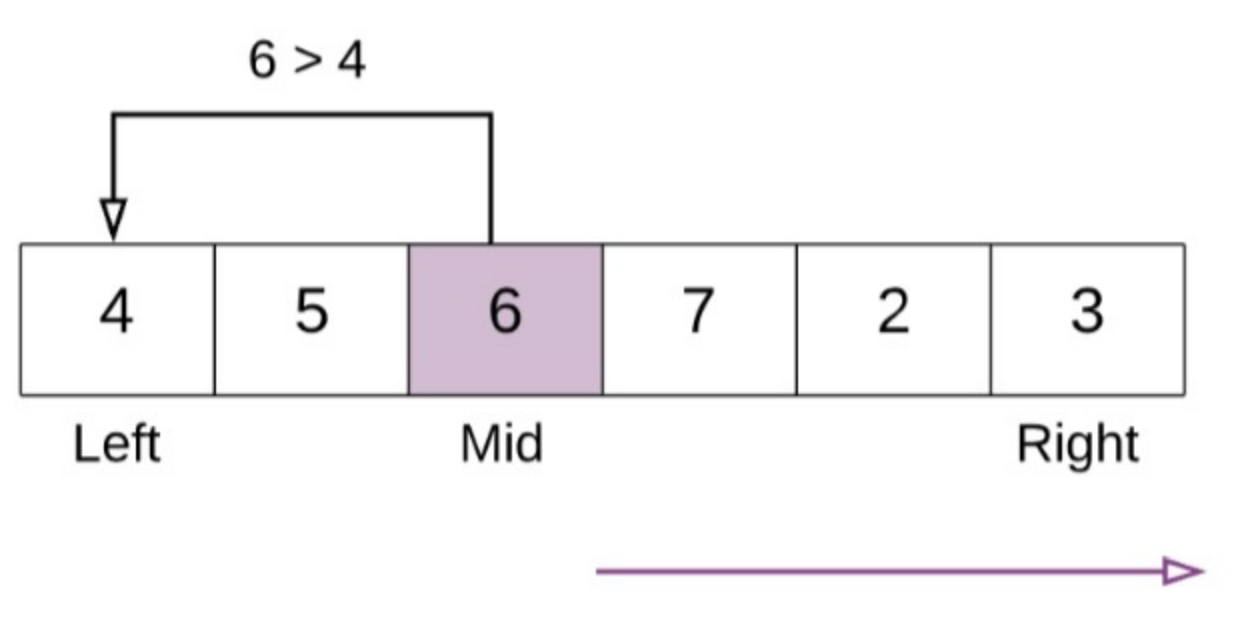


In this modified version of binary search algorithm, we are looking for this point. In the above example notice the `Inflection Point` .

> All the elements to the left of inflection point > first element of the array.
> All the elements to the right of inflection point < first element of the array.

Algorithm

1. Find the `mid` element of the array.
2. If `mid element > first element of array` this means that we need to look for the inflection point on the right of `mid`.
3. If `mid element < first element of array` this that we need to look for the inflection point on the left of `mid`.

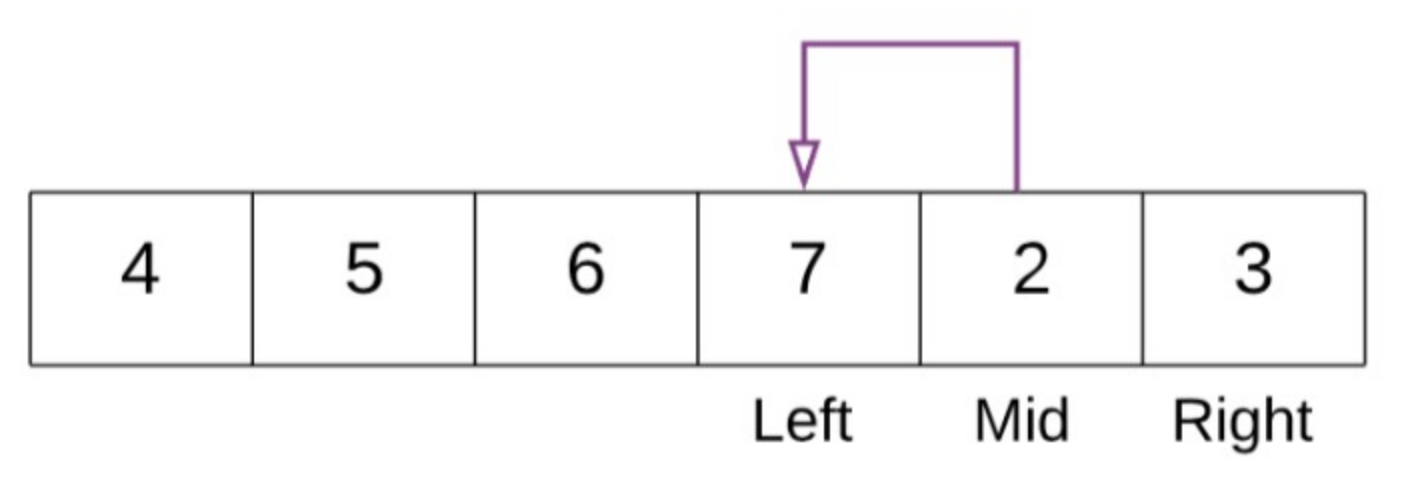


In the above example mid element `6` is greater than first element `4`. Hence we continue our search for the inflection point to the right of mid.

4 . We stop our search when we find the inflection point, when either of the two conditions is satisfied:

`nums[mid] > nums[mid + 1]` Hence, **mid+1** is the smallest.

`nums[mid - 1] > nums[mid]` Hence, **mid** is the smallest.



In the above example. With the marked left and right pointers. The mid element is `2`. The element just before `2` is `7` and `7>2` i.e. `nums[mid - 1] > nums[mid]`. Thus we have found the point of inflection and `2` is the smallest element.

Java | Python | Copy

```python
class Solution(object):
    def findMin(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        # If the list has just one element then return that element.
        if len(nums) == 1:
            return nums[0]

        # left pointer
        left = 0
        # right pointer
        right = len(nums) - 1

        # if the last element is greater than the first element then there is no rotation.
        # e.g. 1 < 2 < 3 < 4 < 5 < 7. Already sorted array.
        # Hence the smallest element is first element. A[0]
        if nums[right] > nums[0]:
            return nums[0]

        # Binary search way
        while right >= left:
            # Find the mid element
            mid = left + (right - left) / 2
            # if the mid element is greater than its next element then mid+1 element is the smallest
            # This point would be the point of change. From higher to lower value.
```

Complexity Analysis

- Time Complexity : Same as Binary Search $O(\log N)$
- Space Complexity : $O(1)$

Rate this article:

Previous Next

Comments: 55

Sort By

Type comment here... (Markdown is supported)

Preview Post

tomba 141 February 17, 2019 2:05 PM

```
    int findMin(vector<int>& nums) {
        int lo = 0, hi = nums.size()-1;
        while (lo < hi) {
```

Read More

50 Share Reply

SHOW 10 REPLIES

DenisGubar 62 September 6, 2018 11:50 AM

Too many comparisons.
The problem's invariant for shifted cases is left element always greater than right. Maintaining it is sufficient for binary search.

Read More

49 Share Reply

SHOW 6 REPLIES

andruet 19 August 3, 2019 8:58 AM

python solution using only 8 lines

```
left, right = 0, len(nums) - 1
while nums[left] > nums[right]:
```

Read More

18 Share Reply

SHOW 4 REPLIES

azimbabu 133 November 19, 2018 10:00 AM

Line 37 of Java solution should be if (nums[mid] > nums[left]) instead of if (nums[mid] > nums[0])

10 Share Reply

SHOW 1 REPLY

codepractice1 6 October 13, 2018 11:58 PM

Great explanation! I had one here but it wasn't as concise: http://code.scottshipp.com/2018/06/27/find-the-minimum-in-a-sorted-rotated-array/

4 Share Reply

SHOW 1 REPLY

jainrishabh 73 September 9, 2018 7:50 PM

A little simpler version:

```
public static void main(String[] args) {
            int arr[] = { 4, 4, 4, 4 };
```

Read More

6 Share Reply

SHOW 1 REPLY

kirj 15 November 25, 2018 10:20 AM

We can break the loop once we see a sorted subarray (break if `nums[from] <= nums[to]`):

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

Read More

2 Share Reply

chef29 2 September 5, 2018 10:08 AM

the solution code seems to have a bug, returns -1 for [2,2,2,2,2,3,3,1,2]

2 Share Reply

SHOW 6 REPLIES

kindernerd 8 September 9, 2018 7:32 PM

```
class Solution {
    public int findMin(int[] nums) {
        if(nums[0]<nums[nums.length-1]){
```

Read More

1 Share Reply

SHOW 1 REPLY

lukskywalkerrr 10 June 11, 2020 1:44 PM

Easiest way to solve this problem:

Apply a filter of < last element to each element, and we get a boolean array

[3,4,5,1,2] => filter(< last element) => [F, F, F, F, T, T]

Read More

0 Share Reply

SHOW 1 REPLY

‹ 1 2 3 4 5 6 ›