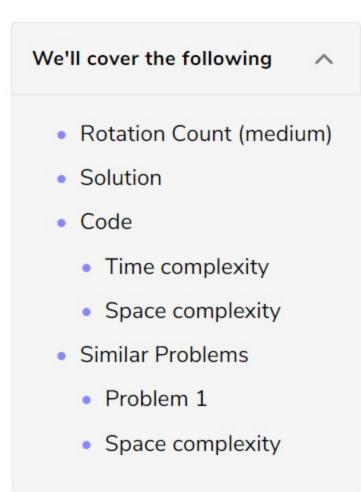


13% completed

Q Search Course

= educative

Solution Review: Problem Challenge 3



Rotation Count (medium)

Given an array of numbers which is sorted in ascending order and is rotated 'k' times around a pivot, find 'k'.

₿

You can assume that the array does not have any duplicates.

Example 1:

```
Input: [10, 15, 1, 3, 8]
Output: 2
Explanation: The array has been rotated 2 times.
                               Original array:
                                                               10
                                                 10 15
                         Array after 2 rotations:
```

Example 2:

```
Input: [4, 5, 7, 9, 10, -1, 2]
Output: 5
Explanation: The array has been rotated 5 times.
                                Original array:
                         Array after 5 rotations:
                                                               9
                                                                    10
```

Example 3:

```
Input: [1, 3, 8, 10]
Output: 0
Explanation: The array has not been rotated.
```

Solution

This problem follows the Binary Search pattern. We can use a similar strategy as discussed in Search in Rotated Array.

In this problem, actually, we are asked to find the index of the minimum element. The number of times the minimum element is moved to the right will be equal to the number of rotations. An interesting fact about the minimum element is that it is the only element in the given array which is smaller than its previous element. Since the array is sorted in ascending order, all other elements are bigger than their previous element.

After calculating the middle, we can compare the number at index middle with its previous and next number. This will give us two options:

- 1. If arr[middle] > arr[middle + 1], then the element at middle + 1 is the smallest.
- 2. If arr[middle 1] > arr[middle], then the element at middle is the smallest.

To adjust the ranges we can follow the same approach as discussed in Search in Rotated Array. Comparing the numbers at indices start and middle will give us two options:

- 1. If arr[start] < arr[middle], the numbers from start to middle are sorted.
- 2. Else, the numbers from middle + 1 to end are sorted.

Code

Here is what our algorithm will look like:

```
Python3
                            G C++
                                        JS JS
  Java
   1 def count_rotations(arr):
        start, end = 0, len(arr) - 1
         while start < end:
          mid = start + (end - start) // 2
          # if mid is greater than the next element
          if mid < end and arr[mid] > arr[mid + 1]:
            return mid + 1
          # if mid is smaller than the previous element
          if mid > start and arr[mid - 1] > arr[mid]:
  11
            return mid
  12
   13
          if arr[start] < arr[mid]: # left side is sorted, so the pivot is on right side</pre>
   14
  15
          else: # right side is sorted, so the pivot is on the left side
   16
            end = mid - 1
  17
        return 0 # the array has not been rotated
  21
  22 def main():
        print(count_rotations([10, 15, 1, 3, 8]))
        print(count_rotations([4, 5, 7, 9, 10, -1, 2]))
        print(count_rotations([1, 3, 8, 10]))
  25
  27
   28 main()
   29
   Run
                                                                                                         Reset
                                                                                               Save
Time complexity
```

Since we are reducing the search range by half at every step, this means that the time complexity of our

algorithm will be O(log N) where 'N' is the total elements in the given array. Space complexity

The algorithm runs in constant space O(1).

Similar Problems

Problem 1 How do we find the rotation count of a sorted and rotated array that has duplicates too?

The above code will fail on the following example!

Example 1:

Explanation: The array has been rotated 3 times

1 def count_rotations_with_duplicates(arr):

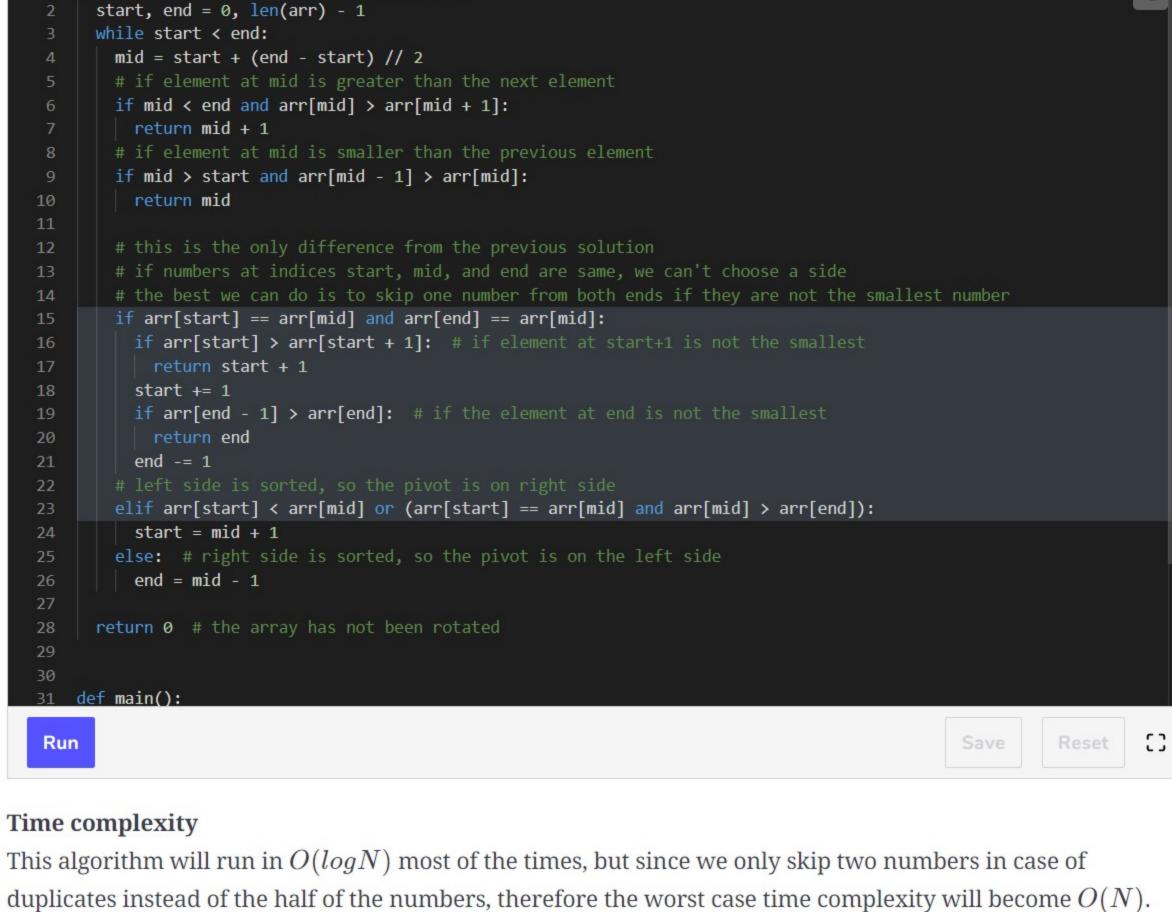
Input: [3, 3, 7, 3] Output: 3

```
Original array:
                               Array after 3 rotations:
Solution
```

incrementing start or decrementing end, we will check if either of them is the smallest number.

G C++ JS JS Python3 Java

We can follow the same approach as discussed in Search in Rotated Array. The only difference is that before

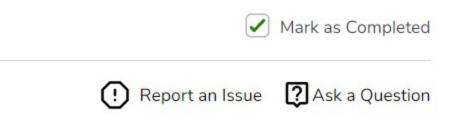


← Back

Problem Challenge 3

Space complexity

The algorithm runs in constant space O(1).



Next ->

Introduction