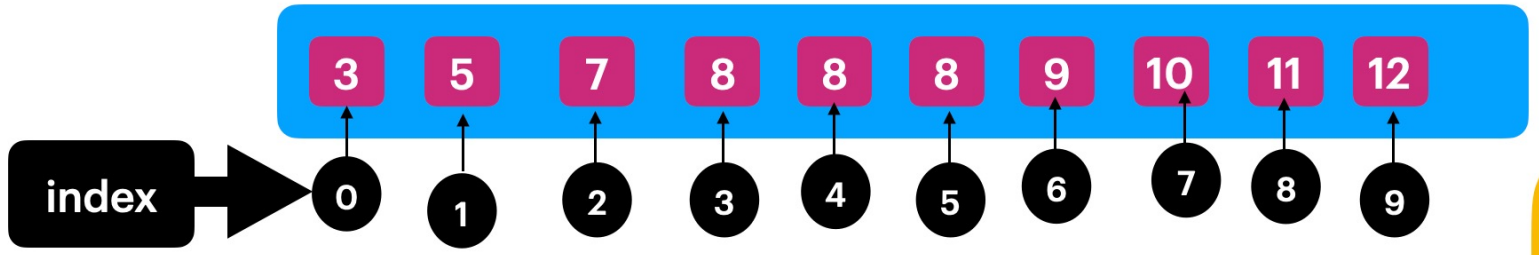
Find Last Occurrence of 8



Algorithm :
Same as Binary Search Algorithm (earlier one)
The only difference is , take a helper variable(i.e result), when we find the element, first update the helper variable with midIndex , then move to right side of the array to reach last occurrence.
when arr[mid] == element
result = mid
start = mid +1

arr
Step1
start = 0
end = 9
mid = 4
result = -1

arr[mid] = element so start = mid+1,
result = mid (i.e result = index 4)

arr
Step2
start = 5
end = 9
mid = 7
result = 4

arr[mid] > element so end = mid-1

arr
Step3
start = 5
end = 6
mid = 5
result = 4

arr[mid] = element so start = mid+1,
result = mid (i.e result = index 5)

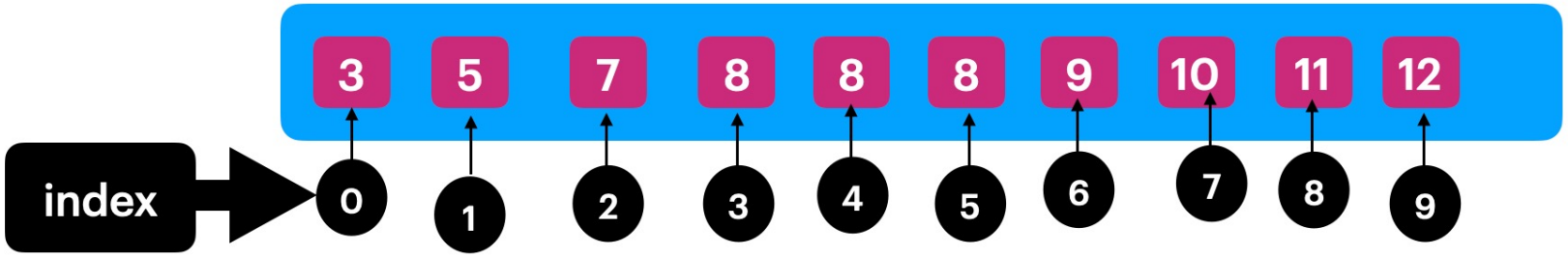
arr
Step4
start = 6
end = 5
result = 5

as start > end
return result i.e index 5

Using iterative
Time Complexity : O(n)
Space Complexity : O(1)

Using Recursion with Binary Search
Time Complexity : O(logn)
Space Complexity : O(logn)
As logn stack frames were active

Using Iterative with Binary Search
Time Complexity : O(logn)
Space Complexity : O(1)



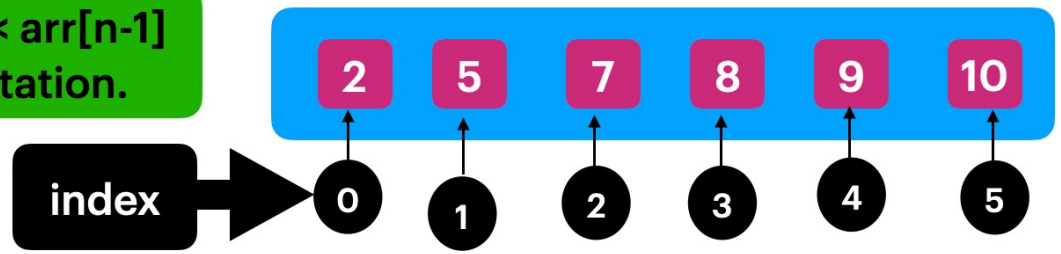
Problem Statement : Find Count of element in a sorted array.

Ex : {3,5,7,8,8,8,9,10,11,12}

count(8) = 3
count(12) = 1
count(15) = -1

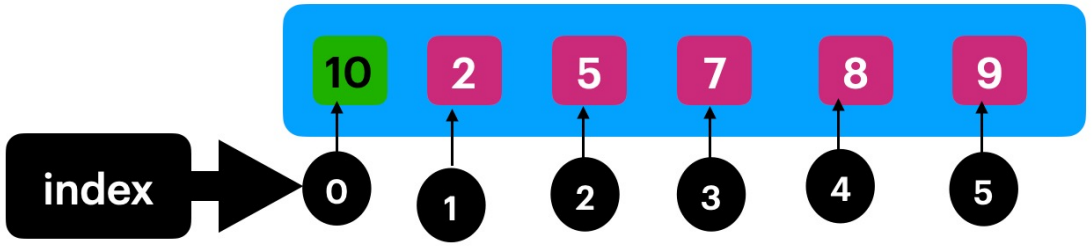
Algorithm : Using Binary Search
Time Complexity O(n)
Space Complexity O(1)
1. Find out Left Occurrence Index of given element
2. Find out Right Occurrence Index of given element
3. If left and right index's are -1 then return -1
otherwise return rightOccurIndex - leftOccrIndex + 1

As arr[0] < arr[n-1] so no rotation.



Rotated Sorted Array Rotation 0

Problem Statement : Find a RotationCount in a Rotated Sorted Array.

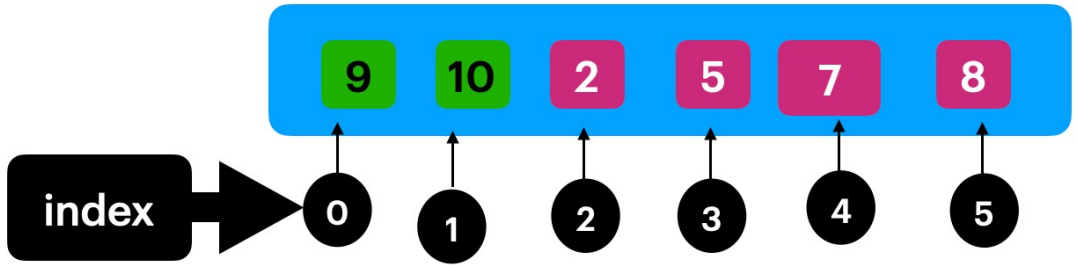


Rotation Count 1

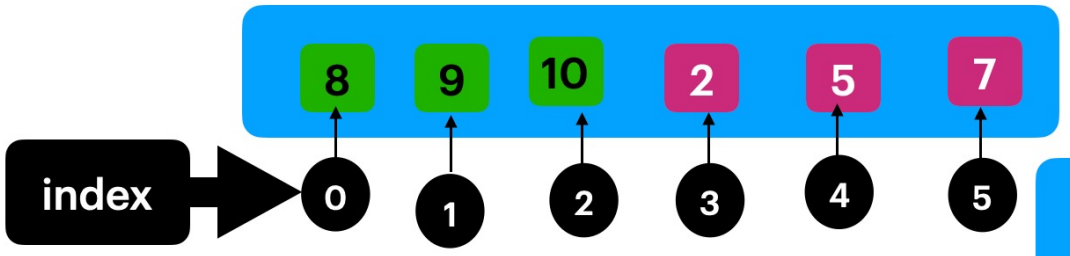
BruteForce Approach.

Find the minimum element within the array , then return the minimum element index.

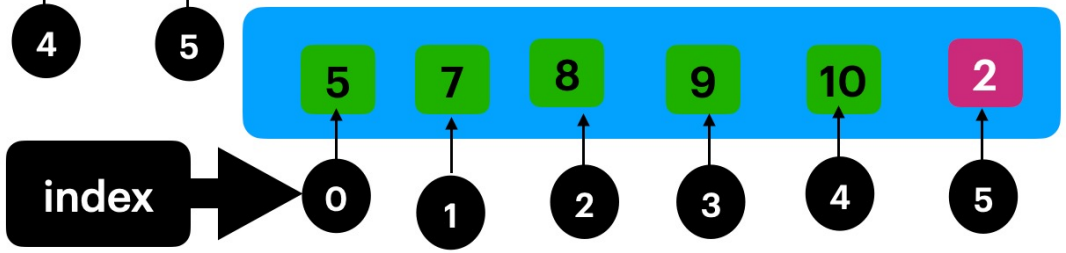
TimeComplexity : O(n)
Space Complexity: O(1)



Rotation Count 2



Rotation Count 3



Rotation Count 5