Find Pivot/Smallest Element in a Sorted Rotated Array

LeetCode

The task is to find the pivot element in a sorted rotated array. The pivot element is the smallest element in the array and marks the point where the ascending order is disrupted.

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

Output: 1

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

Output: 0

Input: nums = [11,13,15,17]

Output: 11

Approach 1: Using Linear Search to find the pivot/Smallest element.

The approach uses linear search to iterate through the array and find the element that is smaller than its subsequent element. It initializes ans with the first element of the array and compares each element with its next element. If an element is greater, it updates ans with the subsequent element. Finally, it returns ans as the pivot element.

This approach has a time complexity of O(n) since it iterates through the array once.

This approach has a space complexity of O(1) since it does not use any additional data structures that scale with the input size.

Approach 2: Using Binary Search to find the pivot/smallest element.

This approach uses a binary search approach to find the pivot element. It initializes the low and high pointers to the start and end of the array, respectively. Within a while loop that continues until low becomes equal to high, it calculates the mid index as the average of low and high.

If the element at the mid index is greater than the element at the high index, it means the pivot lies in the right part of the array. Thus, low is updated to mid+1.

Otherwise, if the element at the mid index is smaller than the element at the low index or equal to it, it means the pivot lies in the left part of the array, or the mid element itself is the pivot. In this case, high is updated to mid.

After the while loop ends, the low index points to the pivot element, so arr[low] is returned as the pivot.

This approach has a time complexity of O(log n) as it uses binary search to narrow down the search space.

This approach has a space complexity of O(1) since it does not use any additional data structures that scale with the input size.