Approach : Binary Search

1. Left = 0, right = n-1
2. While left <= right:
   1. Mid = left + (right-left) //2
   2. To know if mid is in right sorted half or left sorted half and make decision based on that:
   3. Left sorted half: **nums[left] <= nums[mid]**
      1. **Condition to move left**:

We know that left-mid is sorted here.

So if **nums[left] <= target and target < nums[mid], move left**

* + 1. Otherwise move right, left = mid+1
  1. Right sorted half:
     1. **Condition to move right:**

We know that mid-right is sorted here

So if **nums[mid] < target and target <= nums[right],** move right

* + 1. Otherwise move left, right = mid-1

1. Return -1

CODE:

left = 0

right = len(nums)-1

while left <= right:

mid = left + (right-left)//2

if nums[mid] == target:

return mid

# mid in left sorted half

if nums[left] <= nums[mid]:

# condtion to move left: left-mid is sorted so if left < target and mid > target , go left

if nums[left] <= target and target < nums[mid]:

right = mid-1

else:

left = mid+1

# mid in right sorted half

else:

#condition to move right: mid-right sorted(target > mid and target < right)

if nums[mid] < target and nums[right] >= target:

left = mid + 1

else:

right = mid - 1

return -1