Search in Rotated Sorted Array - Notes

What is a Rotated Sorted Array?

- A sorted array is rotated by shifting some elements from the beginning to the end.
- Example:

  Sorted: [0, 1, 2, 3, 4, 5]

  Rotated: [3, 4, 5, 0, 1, 2]
- We must search for a target element in this rotated array using binary search logic.

Goal

Find the index of the target value in the array in O(log n) time.

Concept and Logic Behind the Algorithm

1. Binary Search Basics

- Normally, binary search works on a sorted array.
- We keep checking the middle element and reduce the search range accordingly.

2. New Challenge: Rotation

- In a rotated array, the whole array isn't sorted, but one half is always sorted.
- So, instead of checking if the whole array is sorted, we check which half is sorted.

Step-by-Step Logic

Step 1: Start Binary Search

- Set two pointers: start at the beginning and end at the last index.

Step 2: Find Middle

- Use the formula: mid = start + (end - start) / 2

Step 3: Compare with Target

- If arr[mid] == target: You found the element return mid

Main Strategy: Decide Which Half to Search

A. Check if Left Half is Sorted

Search in Rotated Sorted Array - Notes

If arr[start] <= arr[mid]:

    Left half is sorted

Then check:

If target lies between arr[start] and arr[mid]:

    Search in left half end = mid - 1

Else:

    Search in right half start = mid + 1

B. Otherwise, Right Half is Sorted

If arr[mid] <= arr[end]:

    Right half is sorted

Then check:

If target lies between arr[mid] and arr[end]:

    Search in right half start = mid + 1

Else:

    Search in left half end = mid - 1

Step 4: Repeat

- Continue the process until the element is found or start > end.

Important Conditions Summary

Condition	Meaning	Action
arr[start] <= arr[mid]	Left half is sorted	Check if target is in it
arr[mid] <= arr[end]	Right half is sorted	Check if target is in it
arr[mid] == target	Target found	Return mid

Final Notes

- Always identify which half is sorted before deciding where to search.
- In every iteration of the loop: one half is guaranteed to be sorted.
- Hence, the search works in O(log n) time.