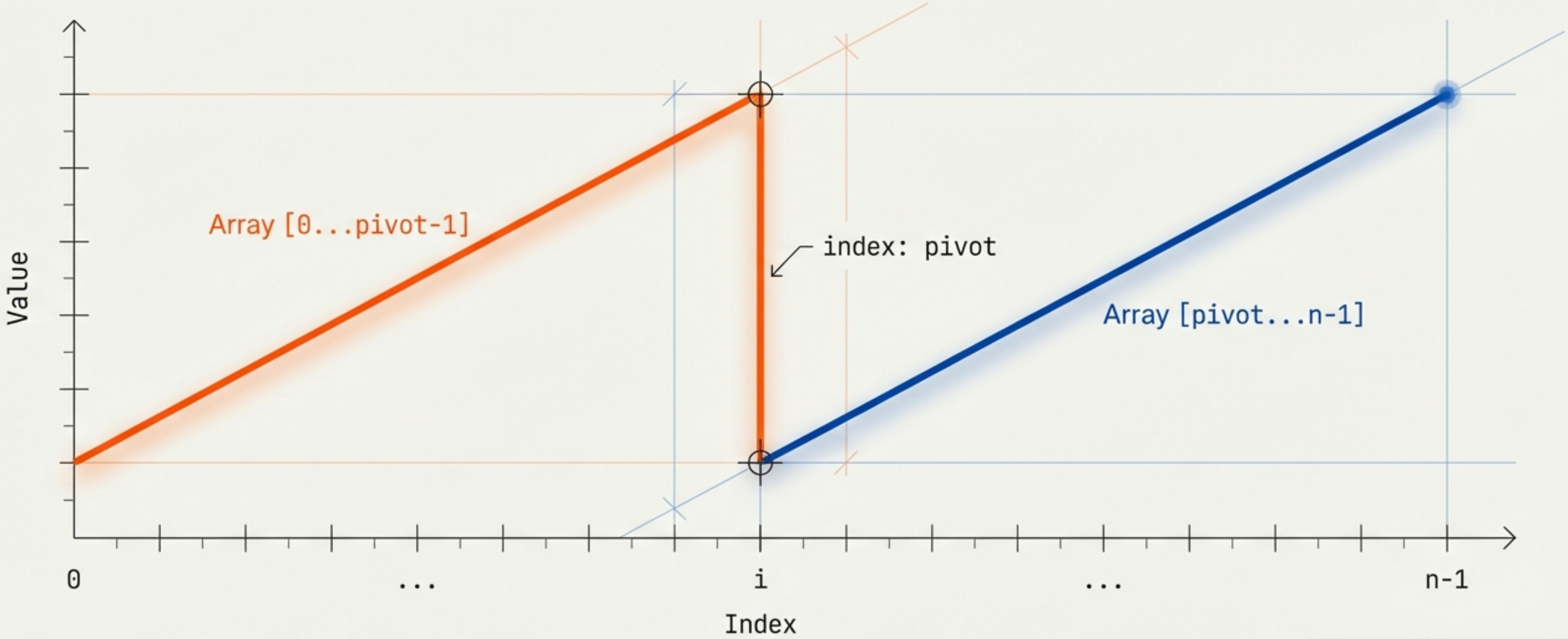
Search in Rotated Sorted Array

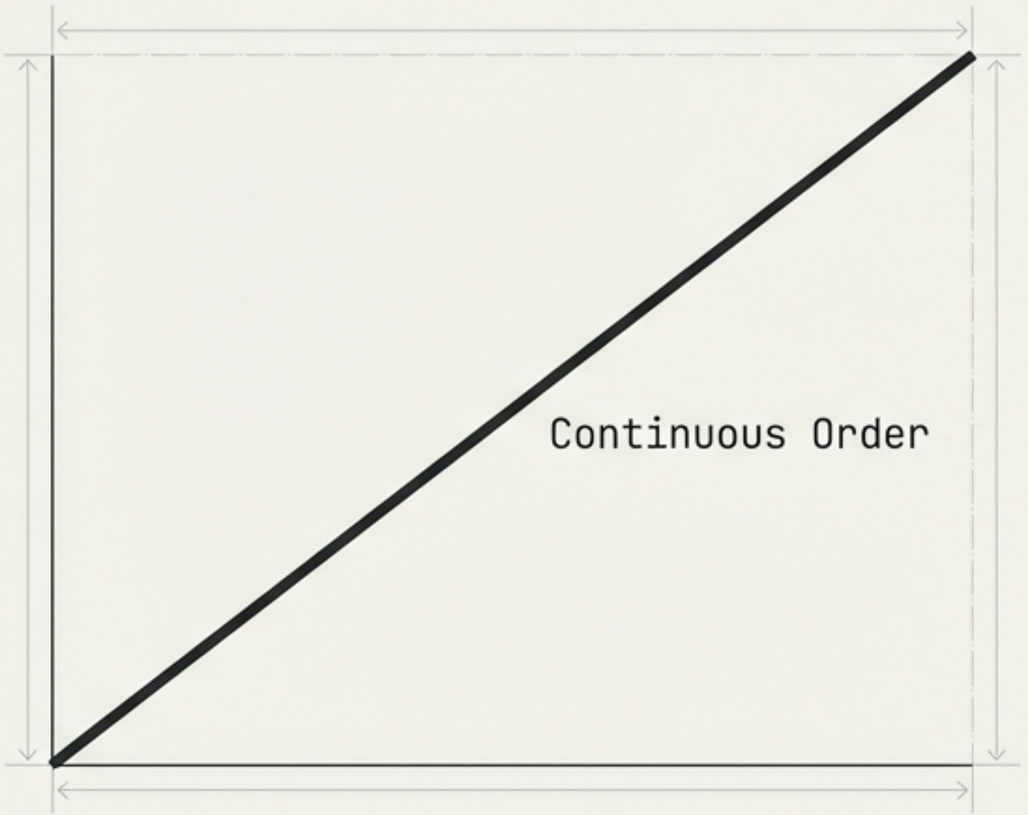
Deriving O(log n) Efficiency through Visual Intuition



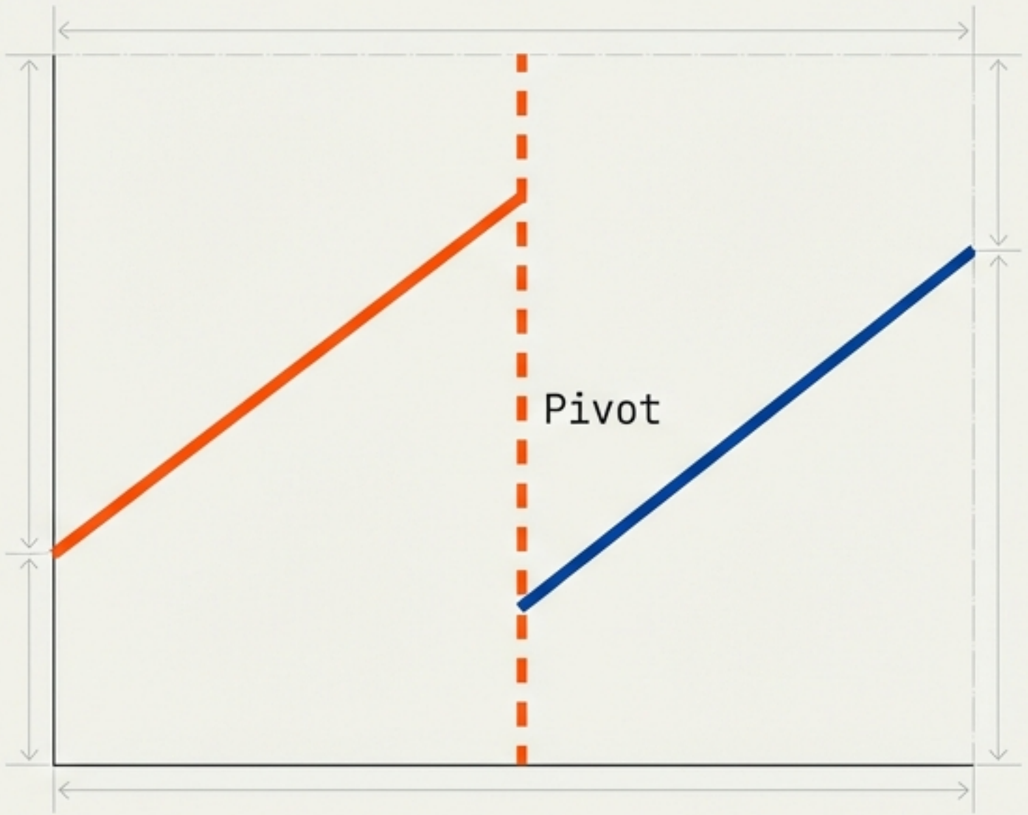
LeetCode 33 / Blind 75

The Geometry of Rotation

Standard Sorted Array

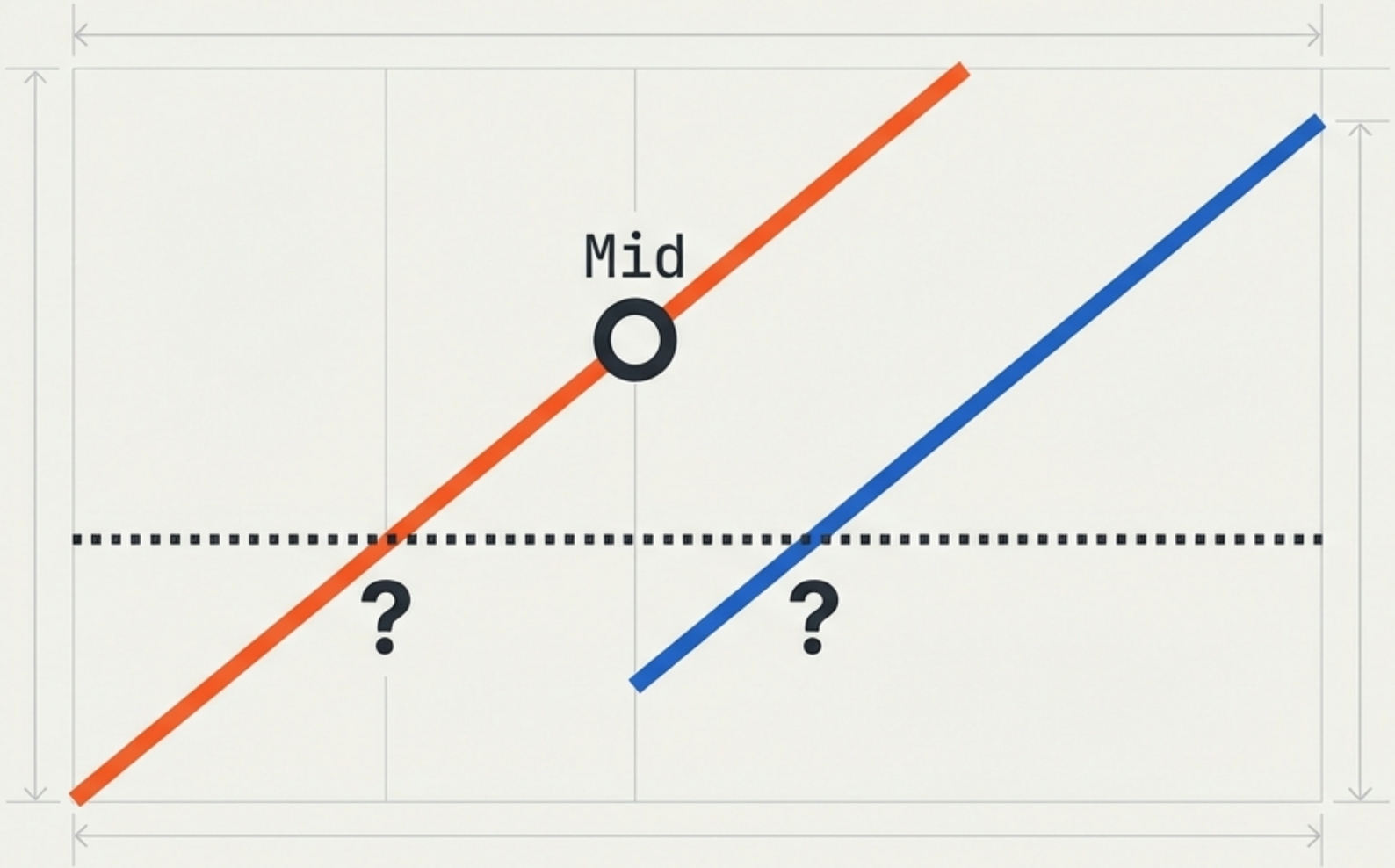


Rotated Sorted Array



We treat the array as a graph. Order exists, but it is discontinuous.

The Binary Search Dilemma



Ambiguity:

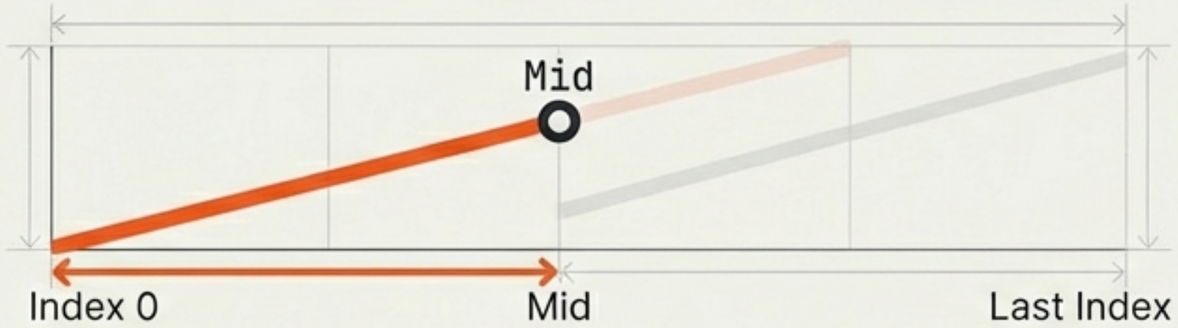
If `Target < Mid`...

Is it here? (**Left**)
Or here? (**Right**)

The Anchor Strategy

Identifying the sorted half

State A: Left Side Sorted



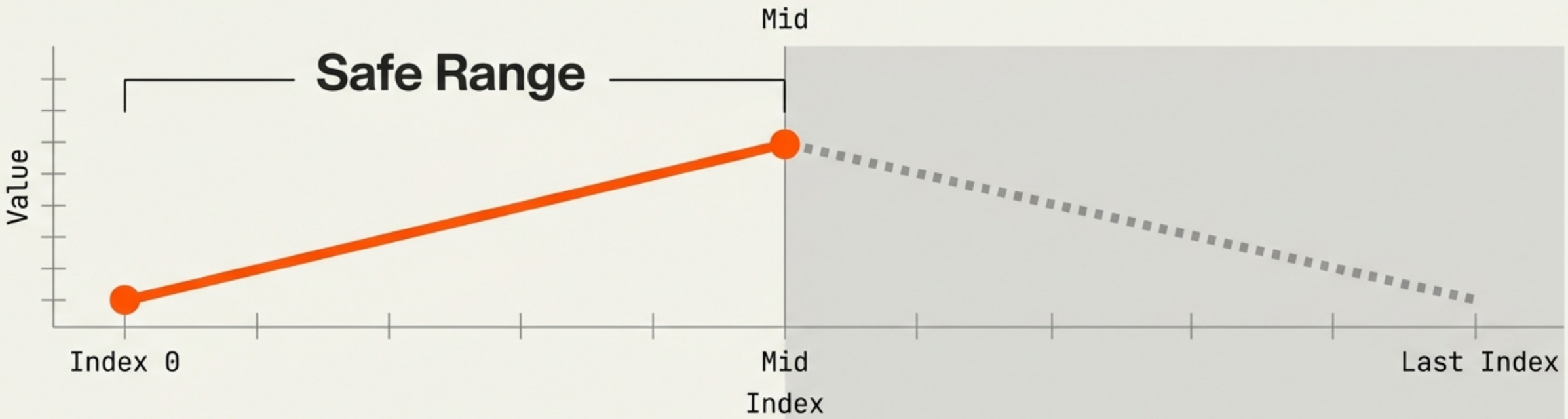

```
if nums[mid] >= nums[left]
```

State B: Right Side Sorted


```
else
```

Case Study 1: Left Sorted Portion

`nums[mid] >= nums[left]`



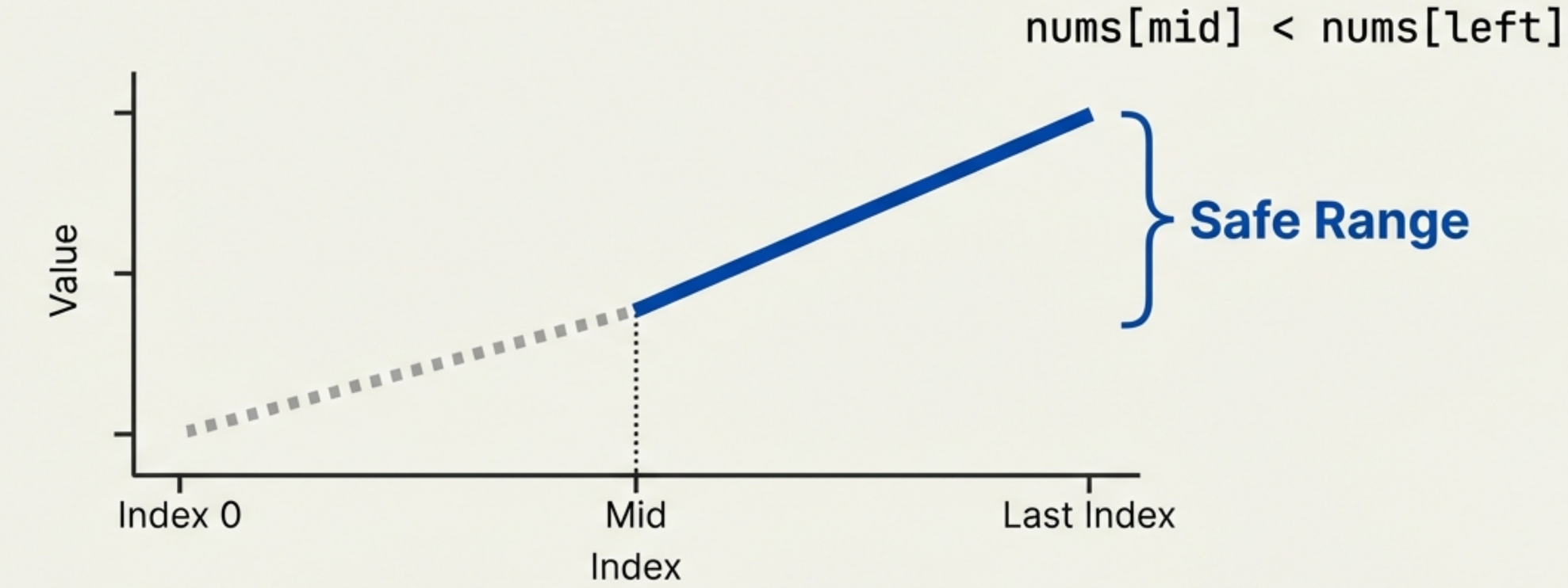
YES ← **Is Target in this range?** → **NO**

Check: `target >= nums[left]` AND `target < nums[mid]`

Search Left
(`right = mid - 1`)

Search Right
(`left = mid + 1`)

Case Study 2: Right Sorted Portion



NO

Search Left
`(right = mid - 1)`

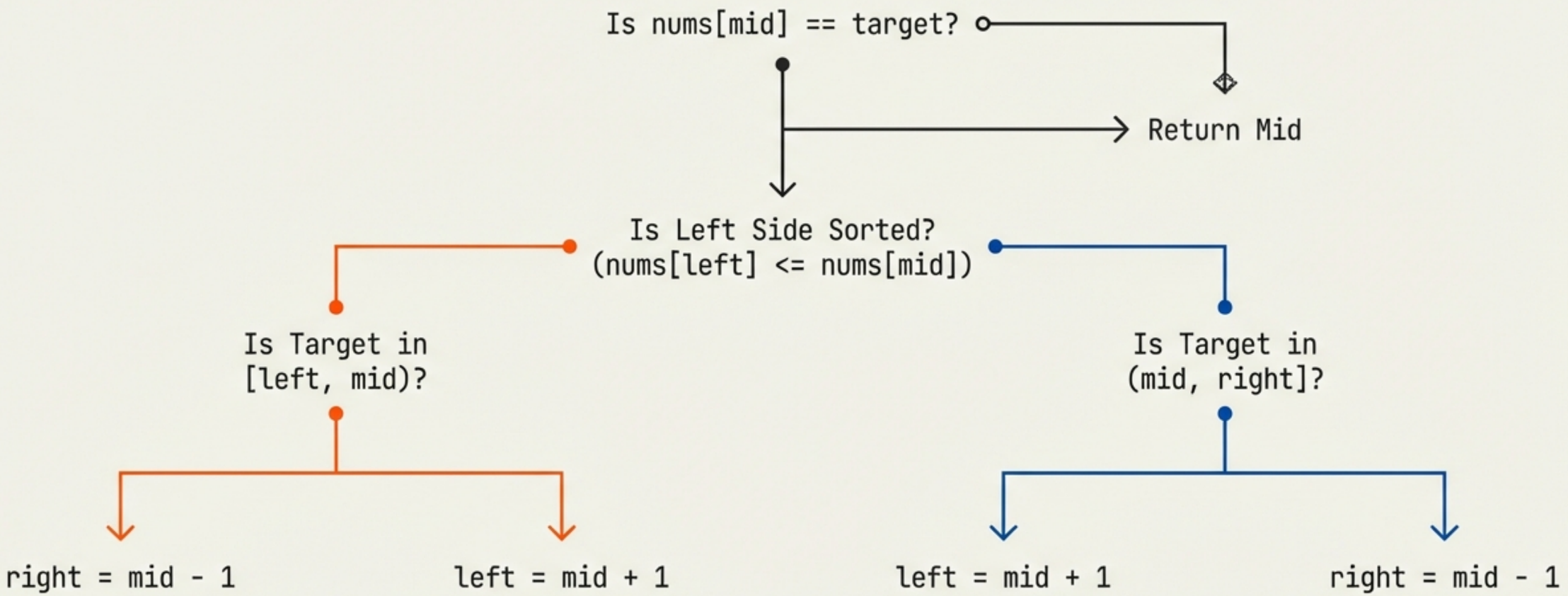
Is Target in this range?

Check: `target > nums[mid]`
AND
`target <= nums[right]`

YES

Search Right
`(left = mid + 1)`

The Decision Algorithm



Implementation (Python)

```python
def search(nums: List[int], target: int) -> int:
    l, r = 0, len(nums) - 1

    while l <= r:
        mid = (l + r) // 2
        if target == nums[mid]:
            return mid

        # Check for Left Sorted Anchor
        if nums[l] <= nums[mid]:
            if nums[l] <= target < nums[mid]:
                r = mid - 1
            else:
                l = mid + 1
        # Otherwise, Right Sorted Anchor
        else:
            if nums[mid] < target <= nums[r]:
                l = mid + 1
            else:
                r = mid - 1
    return -1
```

Case 1: Left Sorted

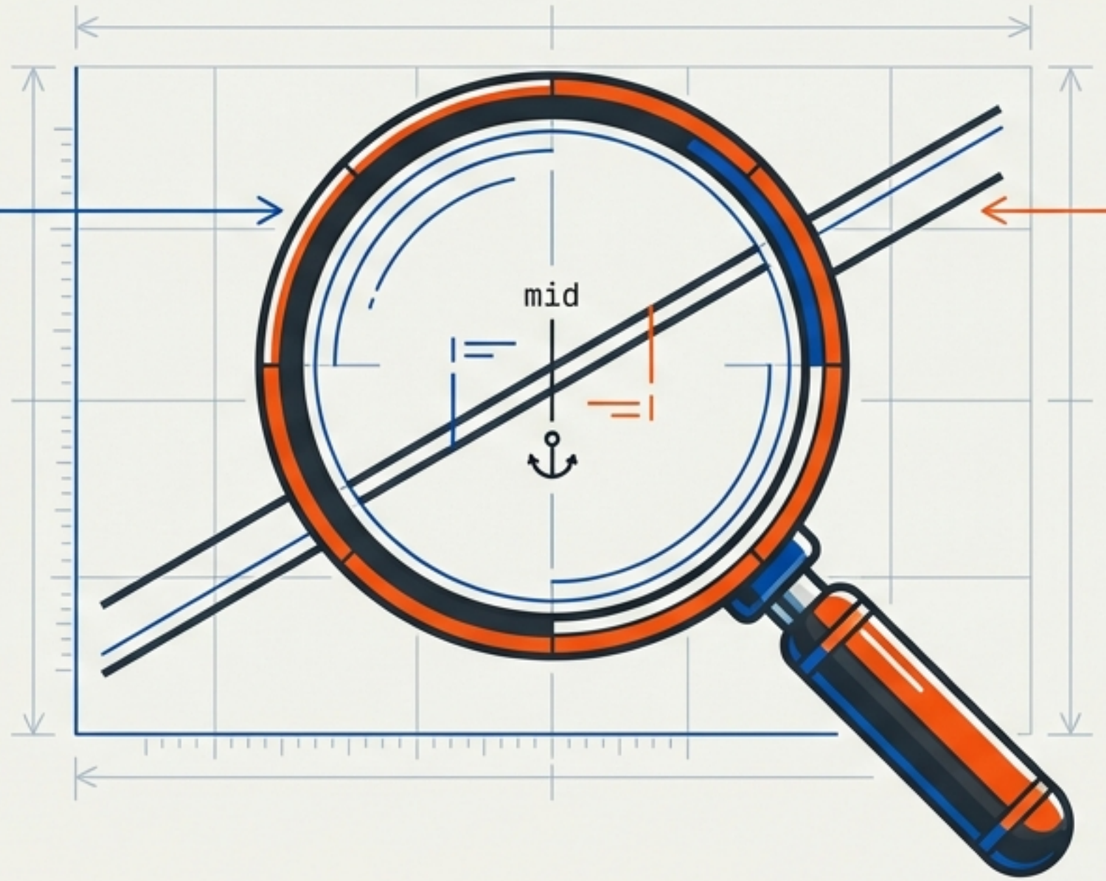
Case 2: Right Sorted

Complexity & Edge Cases

Time Complexity

`O(log n)`

Reason: We eliminate 50% of the search space at every step, regardless of rotation.



Space Complexity

`O(1)`

Reason: Iterative pointers (`l`, `r`, `mid`) only. No recursion stack.

The key is not finding the pivot, but using the sorted half as an anchor.