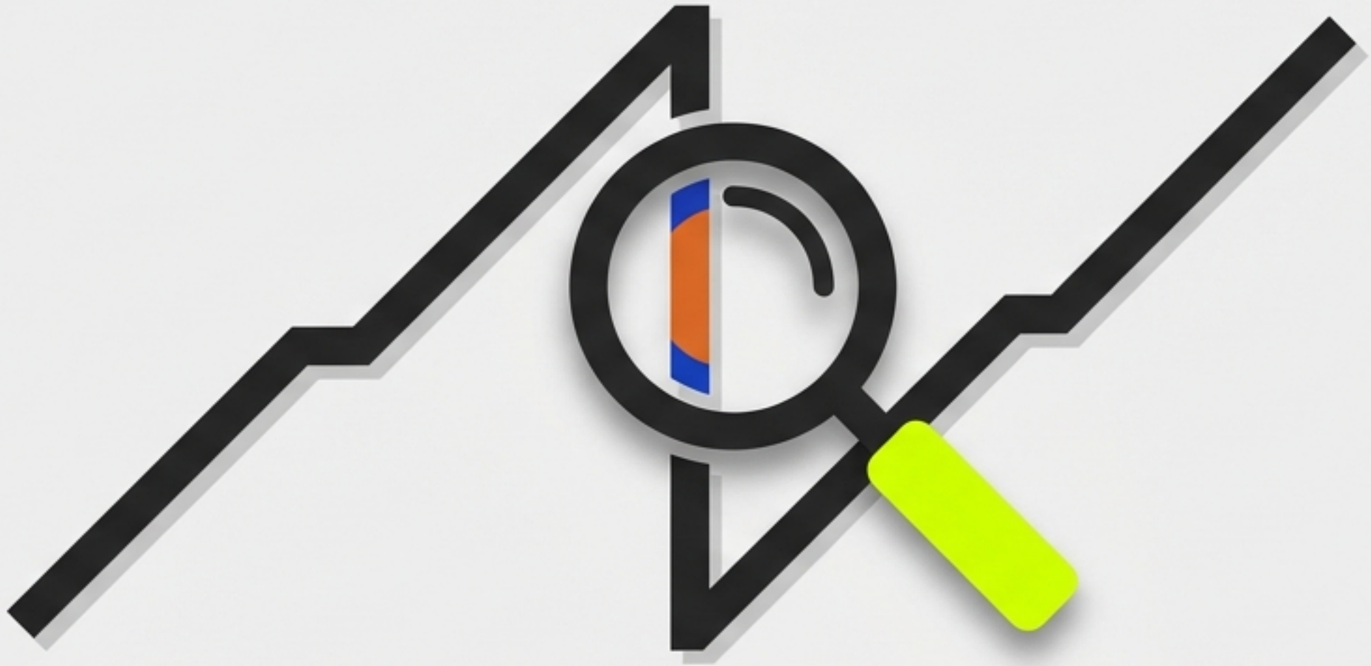
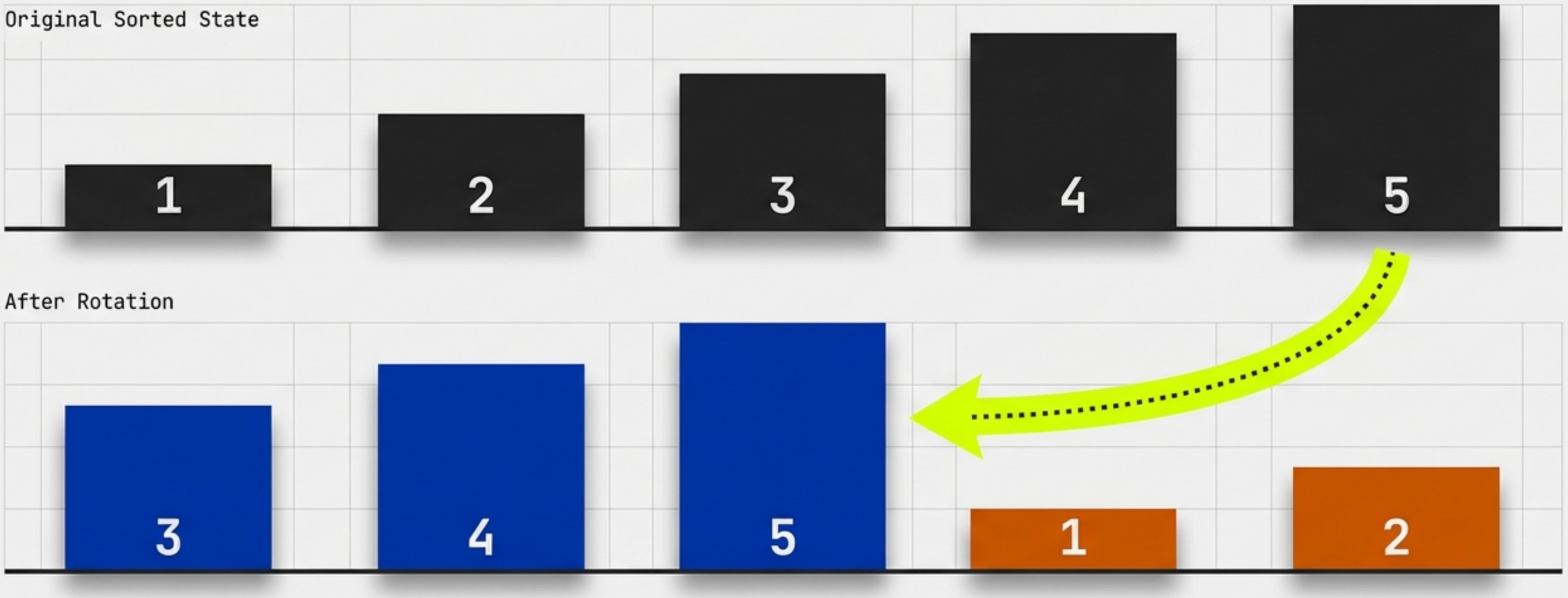
Find Minimum in Rotated Sorted Array



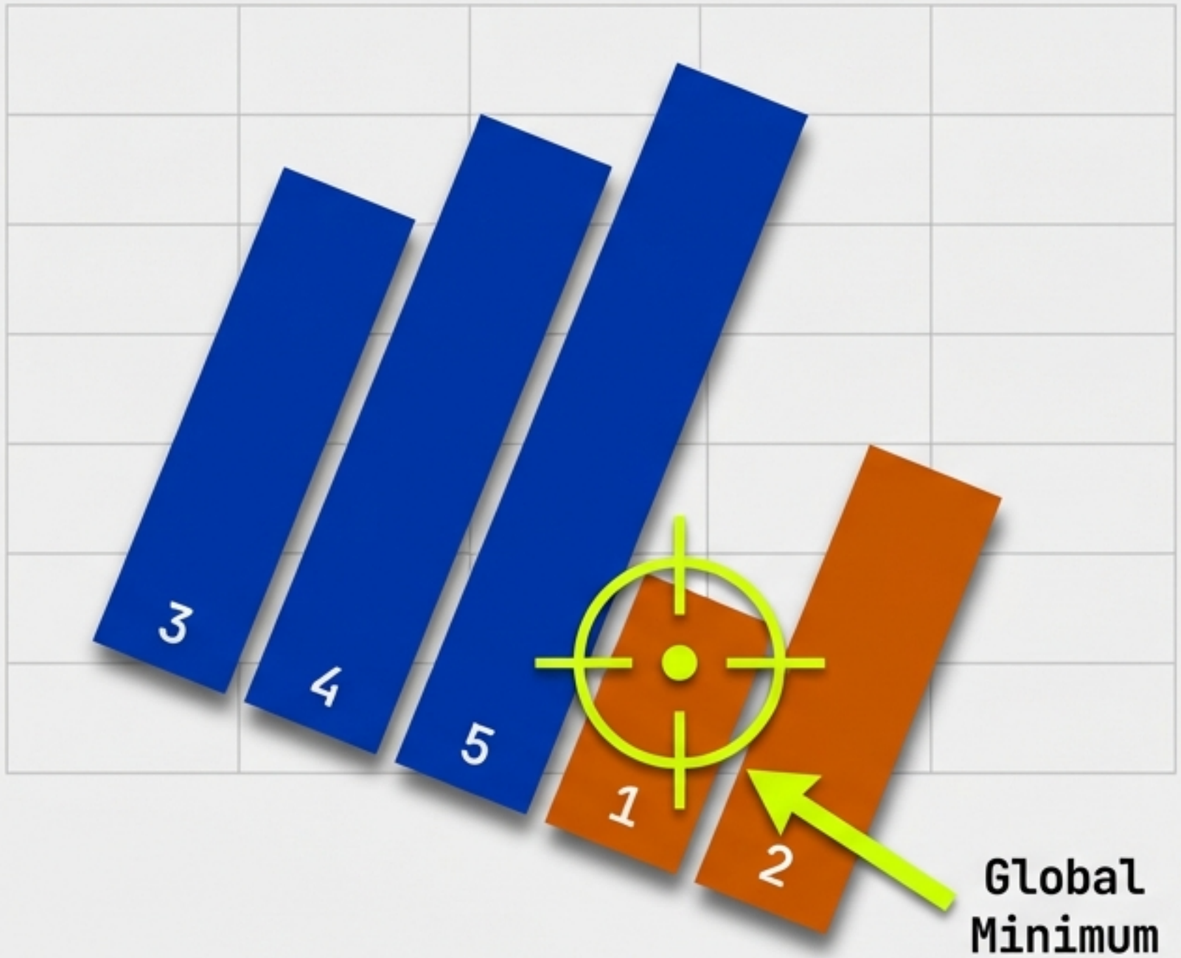
A Visual Detective Approach to Binary Search

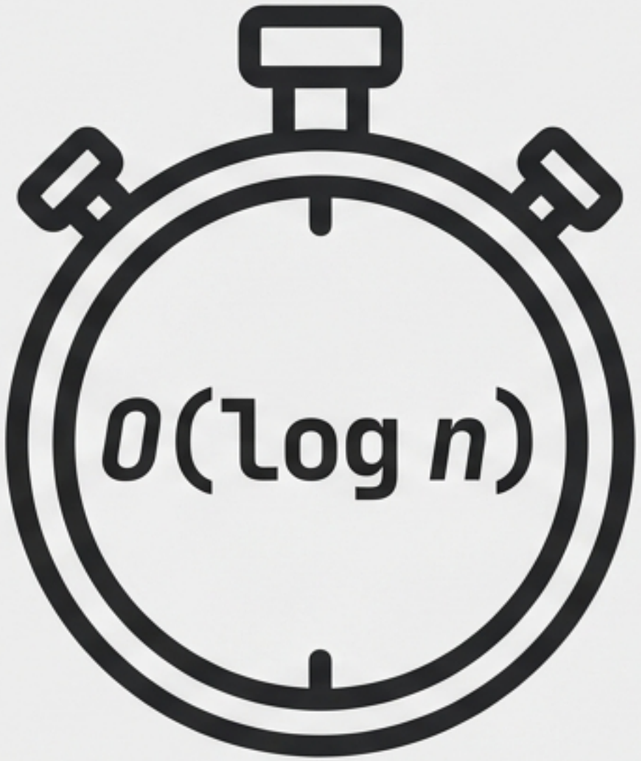
The Anatomy of Rotation



Rotation moves the largest elements behind the smallest. This creates two distinct slopes: a higher ‘Left Sorted Portion’ and a lower ‘Right Sorted Portion’.

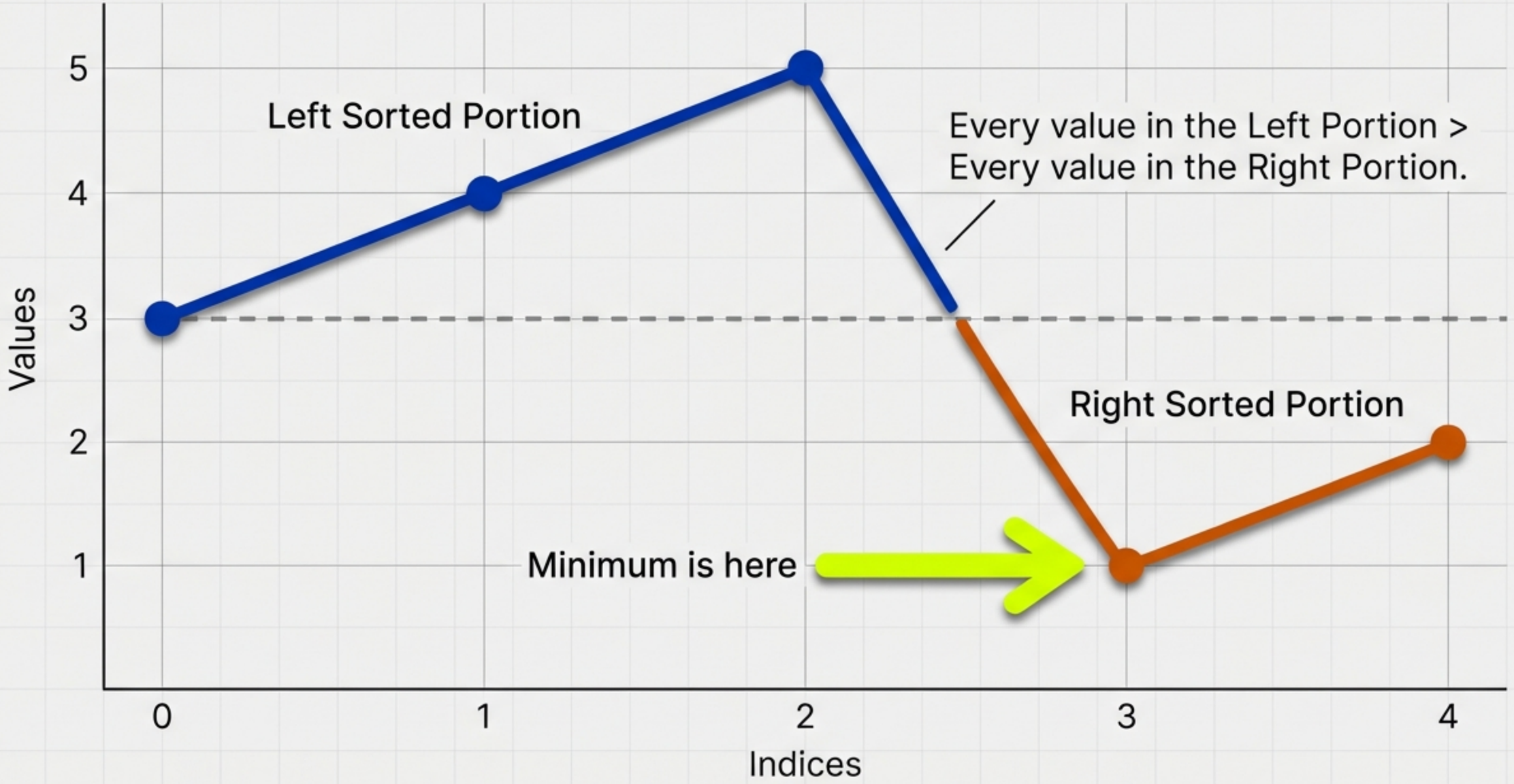
The Objective & The Constraint



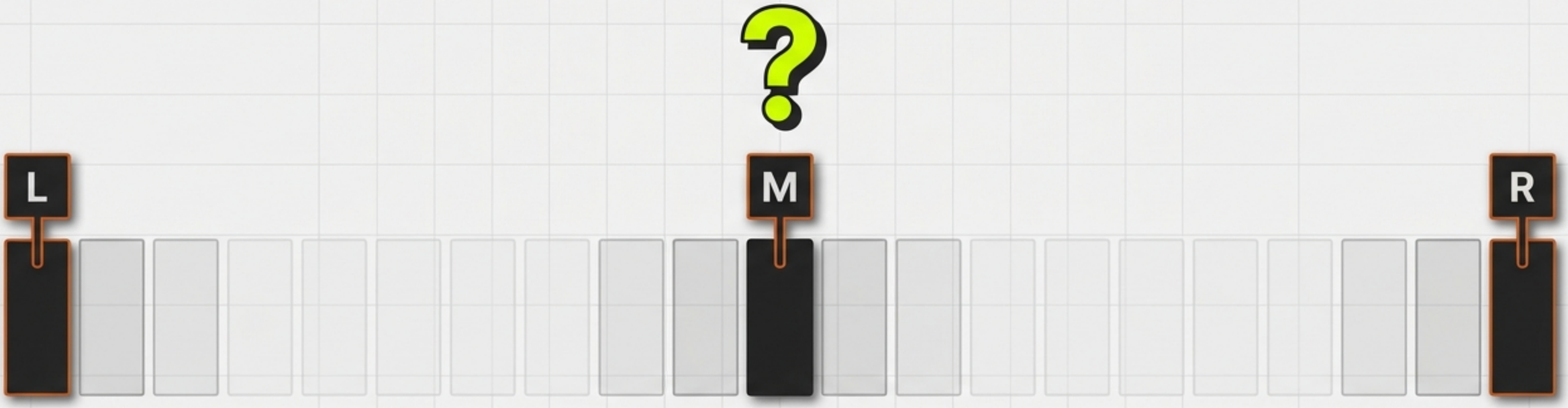


Goal: Find the minimum element.
Constraint: Logarithmic time complexity.
Implication: A linear scan ($O(n)$) is too slow. We must leverage the geometry of the rotation.

The “Two Portions” Insight



The Mid Pointer Identity Crisis

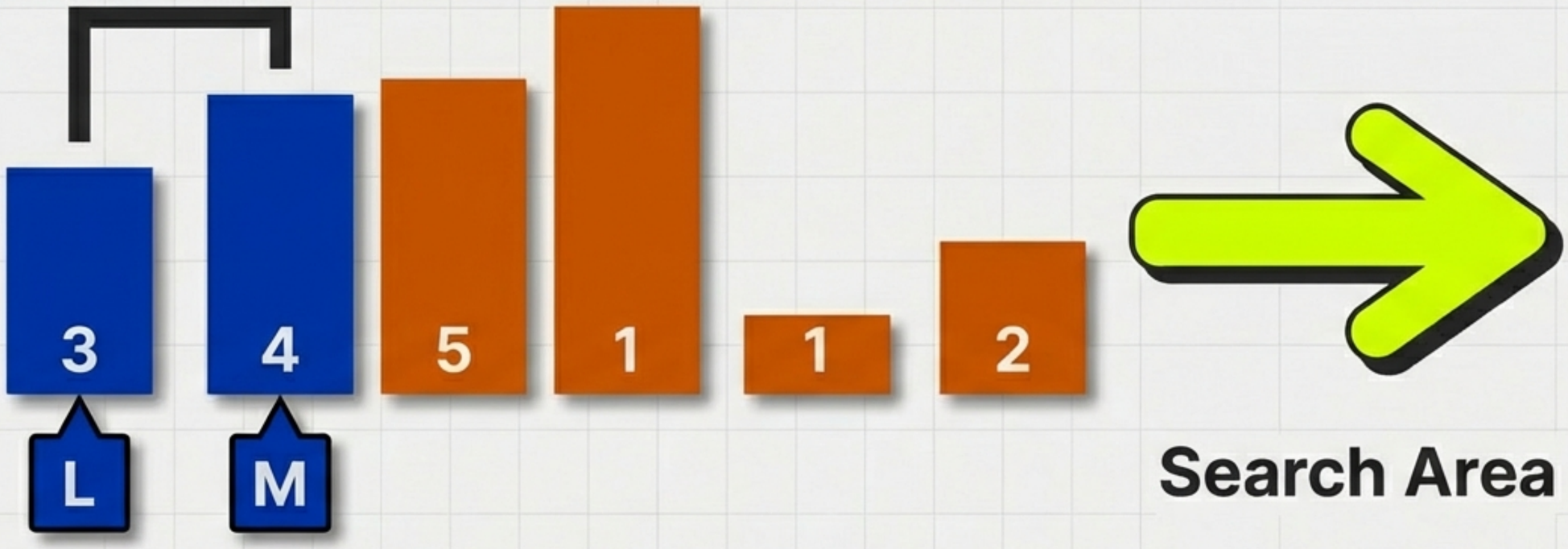


Standard Binary Search:
Compare Mid vs Target

Rotated Binary Search:
Compare Mid vs Left Pointer

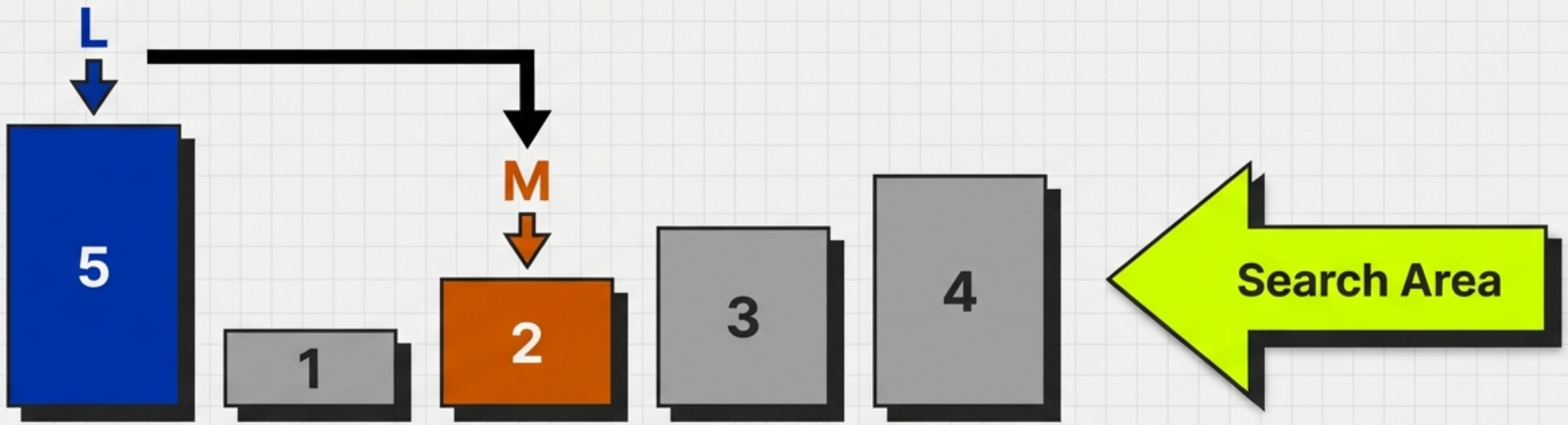
The Core Question: Am I standing on the Left slope or the Right slope?

Identifying the Left Sorted Portion




```
if nums[mid] >= nums[left]:
    # We are on the high ground (Left Portion)
    # Minimum is to the right
    left = mid + 1
```

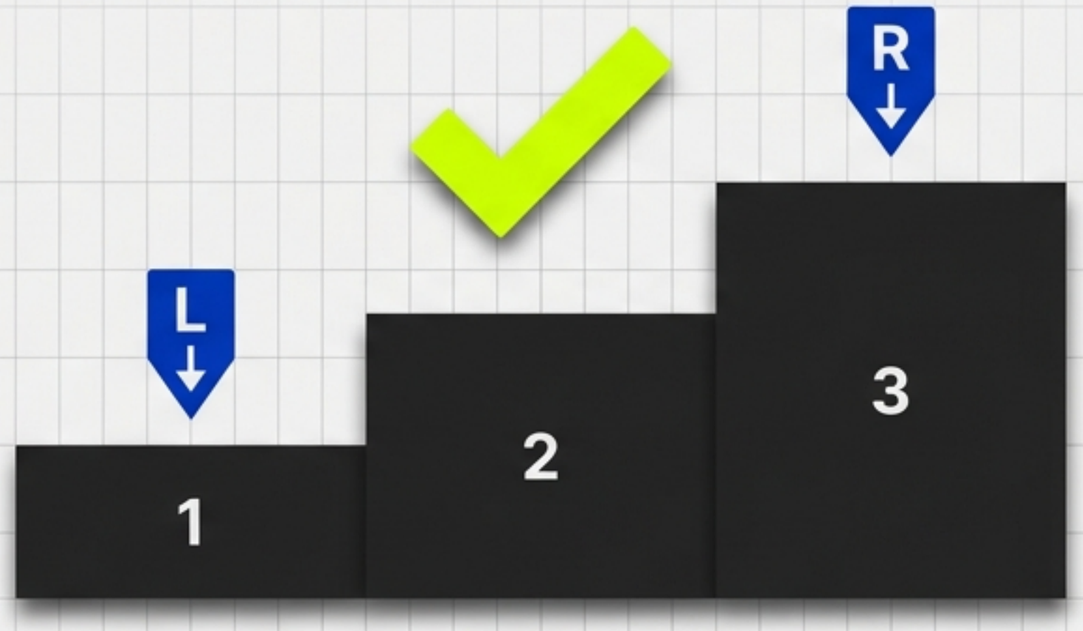
Identifying the Right Sorted Portion




```
else: # nums[mid] < nums[left]
      # We are on the low ground (Right Portion)
      # Minimum is here or to the left
      right = mid - 1
```

```
Track result:
res = min(res, nums[mid])
```

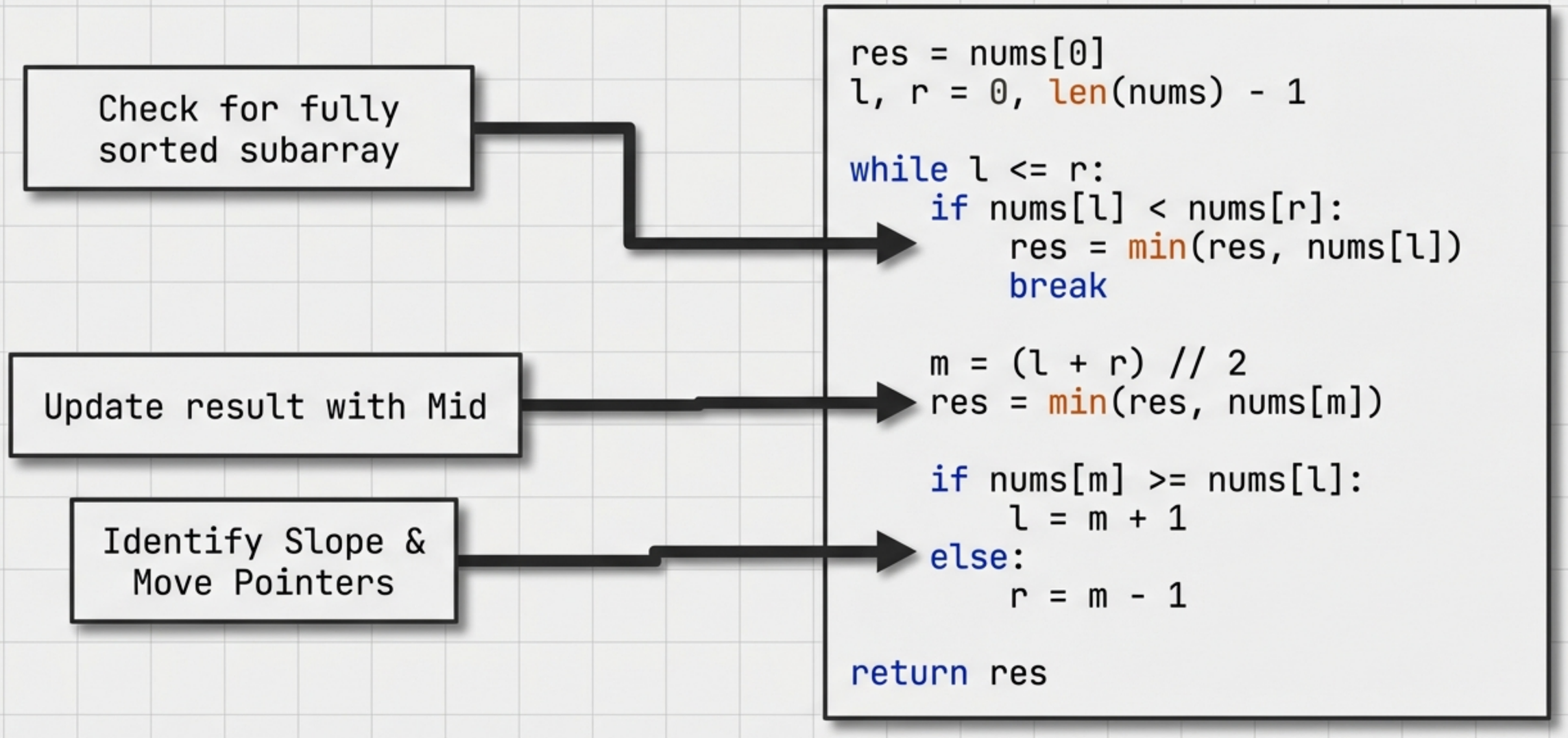
The "Already Sorted" Edge Case



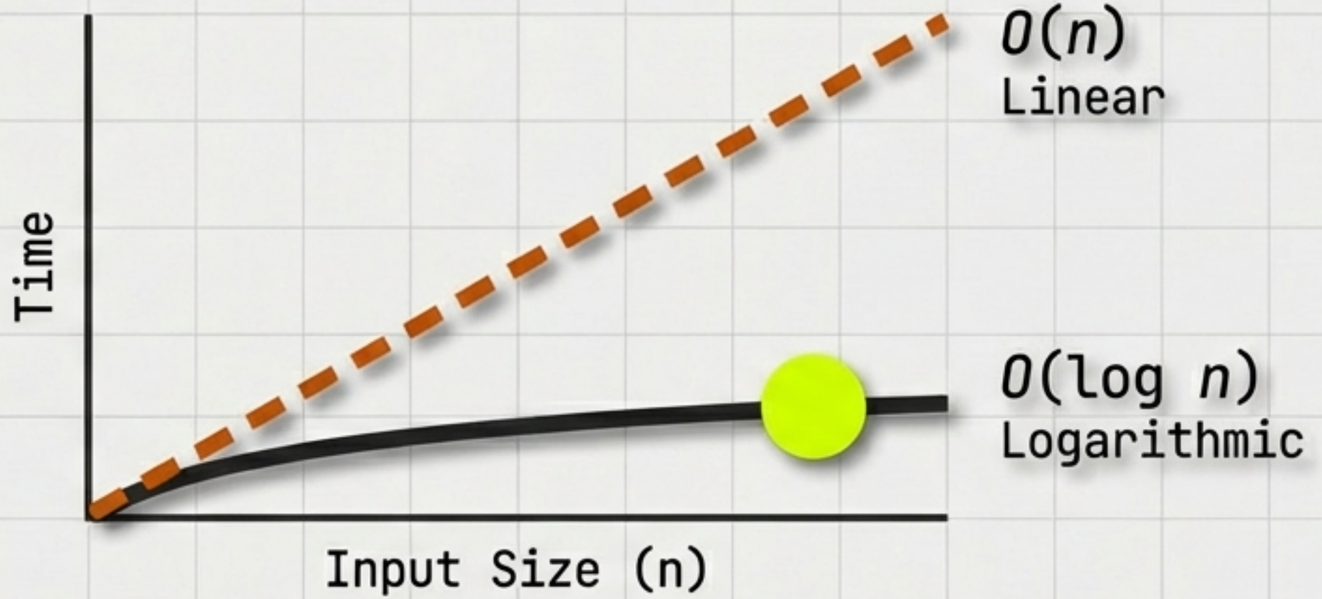

```
if nums[left] < nums[right]:
    res = min(res, nums[left])
    break
```

If the current search range is fully sorted (no rotation pivot), the minimum is simply the first element. No further binary search needed.

Algorithm Assembly



Efficiency & Complexity



Time Complexity	$O(\log n)$ (Binary Search halves the space)
Space Complexity	$O(1)$ (Iterative approach, no recursion stack)

By mapping the array geometry, we solve the mystery without scanning every clue.