33. Search in Rotated Sorted Array

Medium 13819 880 Add to List Share

There is an integer array `nums` sorted in ascending order (with **distinct** values).

Prior to being passed to your function, `nums` is **possibly rotated** at an unknown pivot index `k` (`1 <= k < nums.length`) such that the resulting array is `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]` (**0-indexed**). For example, `[0,1,2,4,5,6,7]` might be rotated at pivot index `3` and become `[4,5,6,7,0,1,2]`.

Given the array `nums` **after** the possible rotation and an integer `target`, return *the index of* `target` *if it is in* `nums`, *or* `-1` *if it is not in* `nums`.

You must write an algorithm with `O(log n)` runtime complexity.

Example 1:

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

Example 2:

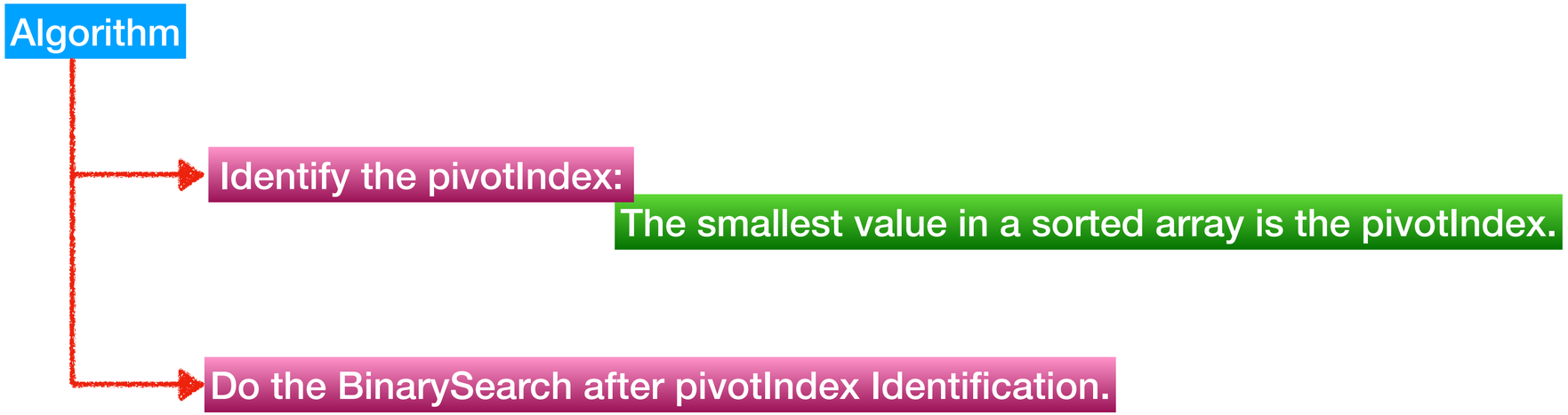
```
Input: nums = [4,5,6,7,0,1,2], target = 3
Output: -1
```

Example 3:

```
Input: nums = [1], target = 0
Output: -1
```

Constraints:

- `1 <= nums.length <= 5000`
- `-10^4 <= nums[i] <= 10^4`
- All values of `nums` are **unique**.
- `nums` is an ascending array that is possibly rotated.
- `-10^4 <= target <= 10^4`


Algorithm
Identify the pivotIndex:
The smallest value in a sorted array is the pivotIndex.
Do the BinarySearch after pivotIndex Identification.

Algorithm to identify the pivotIndex

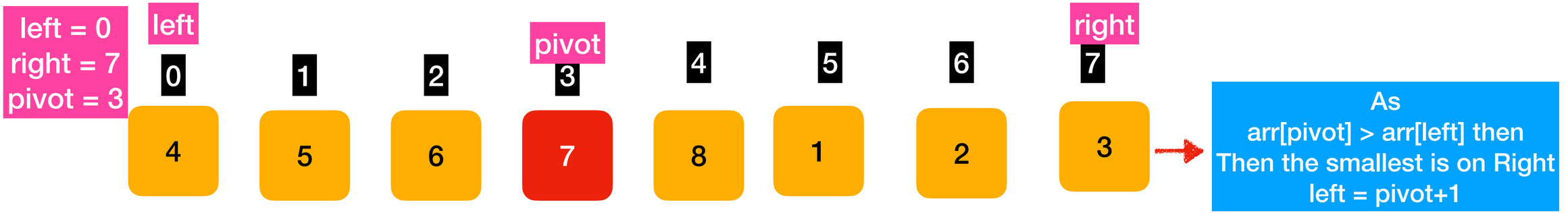
The Smallest value in a sorted array is the pivotIndex:

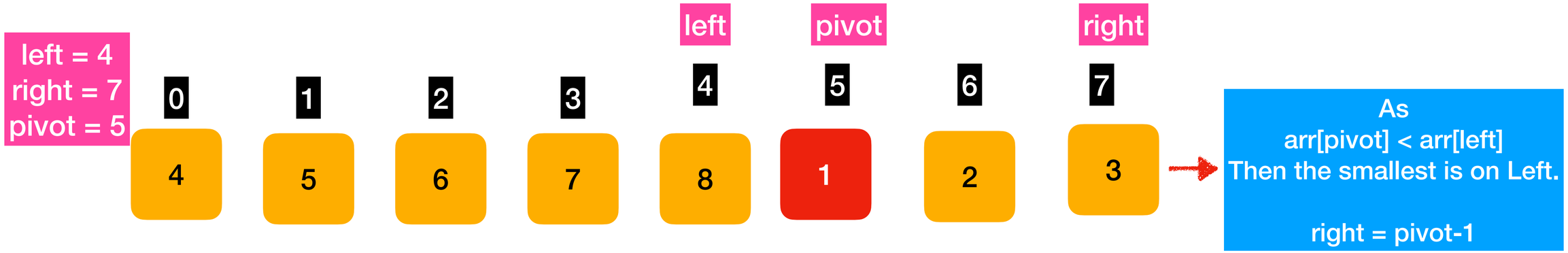
Array is not rotated.

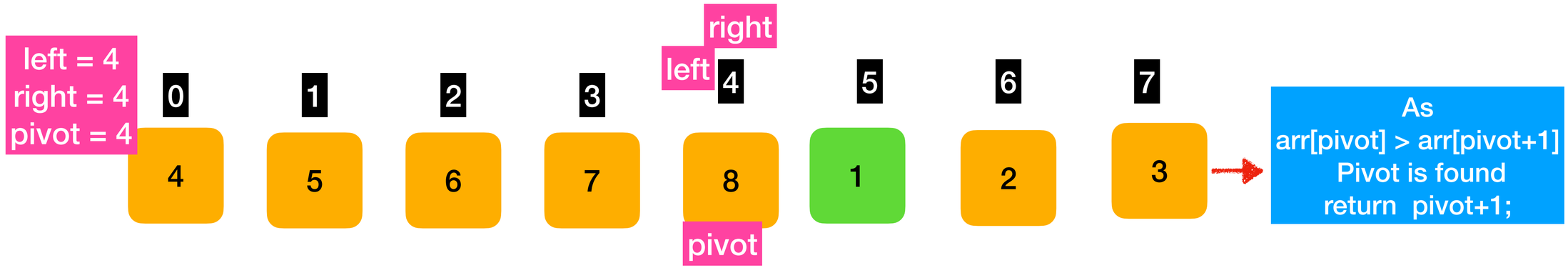
Base Check :
if(arr[0] <= arr[n-1])
{
return 0;
}

Algorithm to identify the pivotIndex

The Smallest value in a sorted array is the pivotIndex:







Algorithm to do BinarySearch After pivotIndex.

target == nums[pivotIndex] —> return pivotIndex.

pivotIndex == 0 then do the BinarySearch from (0, n-1).

target > arr[0] then do the BinarySearch from (0, pivot-1).

taget < arr[0] then do the BinarySearch from (pivotIndex+1, n-1).

Time Complexity : O(logn)
Space Complexity : O(1)