Technique: Binary Search with Duplicates

Level: Easy

Given a sorted array that can contain duplicates, find the first occurrence of the target element. For example:

A = [1,3,4,6,6,6,7] and Target = 6, return index 3.

Questions to Clarify:
Q. Can I assume that the input is an integer array?
A. Yes

Q. How do you want the output?
A. Return the index of the first occurrence.

Q. What to return if the target is not found?
A. Return -1

Solution:
If the mid is not the first occurrence of the target, you treat it like a greater element, i.e, continue your search in the lower half.

To check if a mid is the first occurrence, you check if mid-1 is the same as mid.

Pseudocode:

```
(Note: Never write pseudocode in an actual interview. Unless you're writing a
few lines quickly to plan out your solution. Your actual solution should be in
a real language and use good syntax.)

low = 0, high = a.length - 1
while low <= high
     find mid
     if mid is greater or mid is equal but not 1st occurrence:
         high = mid - 1
     else if mid is less than target
         low = mid + 1
     else
         return mid // mid must be first occurrence

return -1 // not found
```

Test Cases:
Edge Cases: empty array, null array
Base Cases: single element, 2 equal elements (target missing/present)
Regular Cases: only equal elements, no duplicate elements, normal case

Time Complexity: O(log(n))

Space Complexity: O(1)

```
public static int firstOccurence(int[] a, int target) {
    if (a == null) {
        return -1;
    }

    int low = 0;
    int high = a.length - 1;
    while (low <= high) {
        int mid = low + (high - low)/2 ;
        if (a[mid] > target
                || (a[mid] == target && mid > 0 && a[mid - 1] == target)) {
            high = mid - 1;
        } else if (a[mid] < target) {
            low = mid + 1;
        } else {
            return mid;
        }
    }

    return -1;
}
```