SEARCH IN ROTATED SORTED ARRAY

☆ In this problem we are given an array which was initially sorted array but then rotated. Our job is to verify existence of target element. In this variant of the problem, the array may have duplicates. Brute force solution is to use Linear Search but we need to use Binary Search. Very similar to our previous variant, we will check if the middle element is equal to our target element. If it is we return yes, otherwise we perform the same sequence of checking which half of the array is sorted and contains the target.

This will only fail in one edge case, when arr[low] = arr[mid] = arr[high] which can be fixed by doing a low ++ and high --

Pseudocode :

```
searchInRotatedSortedII(arr, N, a) {
    low = 0
    high = N - 1
    while(low <= high) {
        mid = (low + high) / 2
```

```
            if (arr[mid] == a) {
                return Yes
            } else if (arr[mid] == arr[low] == arr[high]) {
                low++
                high--
            } else if (arr[mid] < arr[high]) {
                if (a <= arr[high] && a >= arr[mid]) {
                    low = mid
                } else {
                    high = mid
                }
            } else {
                if (a <= arr[mid] && a >= arr[low]) {
                    high = mid
                } else {
                    low = mid
                }
            }
        }
    return No
}
```