81. Search in Rotated Sorted Array II

已知存在一个按非降序排列的整数数组 nums ，数组中的值不必互不相同。
给你 旋转后 的数组 nums 和一个整数 target ，请你编写一个函数来判断给定的目标值是否存在于数组中。如果 nums 中存在这个目标值 target ，则返回true ，否则返回 false
你必须尽可能减少整个操作步骤。
输入：nums = [2,5,6,0,0,1,2], target = 0
输出 :true

第一类
1011110111 和 1110111101 这种。此种情况下 nums[start] == nums[mid]，分不清到底是前面有序还是后面有序，此时 start++ 即可。相当于去掉一个重复的干扰项。

第二类
2 3 4 5 6 7 1 这种，也就是 **nums[start] < nums[mid]**。此例子中就是 **2 < 5**;
这种情况下，前半部分有序。因此如果 nums[start] <= target <nums[mid]，则在前半部分找，否则去后半部分找。

第三类
6 7 1 **2** 3 4 5 这种，也就是 **nums[start] > nums[mid]**。此例子中就是 **6 > 2**;
这种情况下，后半部分有序。因此如果 nums[mid] <target<=nums[end]。则在后半部分找，否则去前半部分找。
//https://leetcode-cn.com/problems/search-in-rotated-sorted-array-ii/solution/zai-javazhong-ji-bai-liao-100de-yong-hu-by-reedfan/

```
var search = function(nums, target) {
    let left = 0;
    let right = nums.length - 1;

    if (nums[left] === target || nums[right] === target) return true;

    while (nums[left] === nums[left + 1] && left <= right) left++;
    while (nums[right] === nums[right - 1] && left <= right) right++;

    while (left <= right) {
        let mid = left + parseInt((right - left) / 2);
        if (nums[mid] === target) return true;
        //前半部分有序
        if (nums[mid] >= nums[left]) {
            if (target > nums[left] && target < nums[mid]) {
                right = mid - 1;
```

```
            } else {
                left = mid + 1
            }

        } else {
            //后半部分有序
           //target在后半部分
            if (target > nums[mid] && target < nums[right]) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
    }
    return false;
};
```