假设按照升序排序的数组在预先未知的某个点上进行了旋转。

(例如，数组 `[0,0,1,2,2,5,6]` 可能变为 `[2,5,6,0,0,1,2]`)。

编写一个函数来判断给定的目标值是否存在于数组中。若存在返回 `true`，否则返回 `false`。

示例 1:

```
输入：nums = [2,5,6,0,0,1,2], target = 0
输出：true
```

示例 2:

```
输入：nums = [2,5,6,0,0,1,2], target = 3
输出：false
```

进阶:

- 这是 搜索旋转排序数组 的延伸题目，本题中的 `nums` 可能包含重复元素。
- 这会影响到程序的时间复杂度吗？会有怎样的影响，为什么？

思路，该题的解题思路，个人认为2分查找区分状况能进行解决，首先判断左右头的状况，等于需要找的值，返回，不等于，则进行二分查找，若是左边大于右边，则改值存在旋转，则中心点，无论大于，还是小于，都需要到两边进行查找。若是等于同理（无法判断是否为旋转数组）大于，若是小于，则数组为有序的数，2分查找即可。（摩了半天的用例，终于把代码摩出来了）

```
class Solution {
public:
    bool dfs(vector<int>& nums,int left,int right,int &target){
        if(left>right||right>=nums.size()){
            return false;
        }else if(left==right){
            if(nums[left]==target){
                return true;
            }else{
                return false;
            }
        }
        if(nums[left]==target||nums[right]==target){
            return true;
        }
```

```
    if(nums[left]<nums[right]){
        int i=(left+right)/2;
        if(nums[i]==target){
            return true;
        }
        if(nums[i]<target){
            if(dfs(nums,i+1,right,target)){
                return true;
            }
        }else{
            if(dfs(nums,left,i-1,target)){
                return true;
            }
        }
    }else if(nums[left]==nums[right]){
            int i=(left+right)/2;
            if(nums[i]==target){
            return true;
            }
            if(dfs(nums,i+1,right,target)||dfs(nums,left,i,target)){
                return true;
            }
    }else{
        int i=(left+right)/2;
        if(nums[i]==target){
            return true;
        }else if(dfs(nums,i+1,right,target)||dfs(nums,left,i,target)){
                return true;
            }
    }
    return false;
}
bool search(vector<int>& nums, int target) {
    if(nums.size()==0){
        return false;
```

```
        }else{
            return dfs(nums,0,nums.size()-1,target);
        }
    }
};
```