33. Search in Rotated Sorted Array

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand.

(i.e., `[0,1,2,4,5,6,7]` might become `[4,5,6,7,0,1,2]`).

You are given a target value to search. If found in the array return its index, otherwise return `-1`.

You may assume no duplicate exists in the array.

Your algorithm's runtime complexity must be in the order of *O*(log *n*).

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
```

We have to solve this question in O(logn) time
hmmmmm......
logn
Binary Search comes in mind and that is what we are going to use
U can look this array as join of two sorted arrays
we will find the pivot indx that is the starting position of second sorted array
this can be done by binary search
if(nums[mid] > nums[mid+1]) return mid+1

***after that* check if target lies in (0,pivot) or in(pivot+1,n-1)**
and* perform binary search *on that part of array
else return -1

CODE:

```
int findPivot(vector<int> nums,int start,int end){
        if(start >= end)
            return -1;
        int mid = (start+end)/2;
        if(nums[mid] > nums[mid+1])
            return mid+1;
        int a = findPivot(nums,start,mid);
        int b = findPivot(nums,mid+1,end);
        if(a == -1 && b == -1)
            return -1;
        else
            return a==-1?b:a;
    }
    int findIndex(vector<int> nums,int start,int end,int target){
      if(start > end)
          return -1;
     int mid = (start+end)/2;
     if(nums[mid] == target)
         return mid;
      else if(nums[mid] > target)
          return findIndex(nums,start,mid-1,target);
      return findIndex(nums,mid+1,end,target);
    }
    int search(vector<int>& nums, int target) {
```

```
    if(nums.size() == 1)
        return nums[0]==target?0:-1;
  int n = nums.size();
  int pivot = findPivot(nums,0,n-1);
  if(pivot == -1)
    return findIndex(nums,0,n-1,target);
  int index = -1;
    if(target>= nums[0] && target<= nums[pivot-1])
        index = findIndex(nums,0,pivot-1,target);
    else if(target >= nums[pivot] && target <= nums[n-1])
        index = findIndex(nums,pivot,n-1,target);
  return index;
}
```