**33. Search in Rotated Sorted Array**

Medium

6211540Add to ListShare

You are given an integer array nums sorted in ascending order, and an integer target.

Suppose that nums 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]).

*If target is found in the array return its index, otherwise, return -1.*

**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

**Example 3:**

**Input:** nums = [1], target = 0

**Output:** -1

关键点：这是一个变种的binary Search, Binary Search特殊点在于array一定要是Sorted，这里看似被一个PIVOT打乱了，但是

7890123，左边大右边小

假设mid是0时，如果他小于最右边的数，那么他必然不在左边大的数里，也就是说Mid到最右是sorted的

相反，如果mid大于最左边的数，那么他必然在左边的大数里，左边是sorted

常规知识：binary search，left>right的时候，return -1

如果mid正好等于target,return mid

然后循环left,mid-1

和mid+1,right

class Solution {

public int search(int[] nums, int target) {

if(nums.length==0||nums==null) return -1;

return searchRecursive(nums,target,0,nums.length-1);

}

public int searchRecursive(int[] nums, int target, int left, int right){

if(left>right) return -1;

int mid=(left+right)/2;

if(nums[mid]==target) return mid;

//这三步是binary search必备的

if(nums[mid]<nums[right]){ //右边sorted

if(target>nums[mid]&&target<=nums[right]) //如果在右边的范围内

return searchRecursive(nums,target,mid+1,right); //循环右边

else

return searchRecursive(nums,target,left,mid-1); //要么循环左边

}

else if(nums[mid]>=nums[left]){ //左边sorted

if(target<nums[mid]&&target>=nums[left]) //如果在左边的范围内

return searchRecursive(nums,target,left,mid-1); //循环左边

else

return searchRecursive(nums,target,mid+1,right);

}

return -1;

边界问题，我们必须循环到所有的可能

mid等于 return mid

(mid right] //mid在右边的范围的时候，右边的范围就是大于mid，小于等于right

mid left mid在左边的范围的时候，左边的范围就是小于mid，大于等于lleft

}

}