题目：

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 `[0,1,2,4,5,6,7]` 可能变为 `[4,5,6,7,0,1,2]`)。

搜索一个给定的目标值，如果数组中存在这个目标值，则返回它的索引，否则返回 `-1` 。

你可以假设数组中不存在重复的元素。

你的算法时间复杂度必须是 *O*(log *n*) 级别。

示例 1:

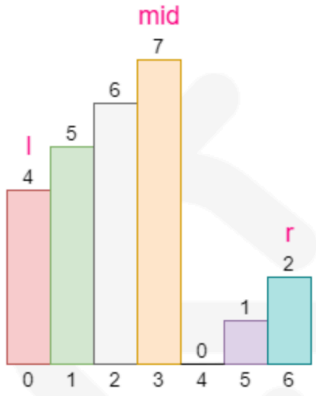
```
输入: nums = [4,5,6,7,0,1,2], target = 0
输出: 4
```

示例 2:

```
输入: nums = [4,5,6,7,0,1,2], target = 3
输出: -1
```

思路；

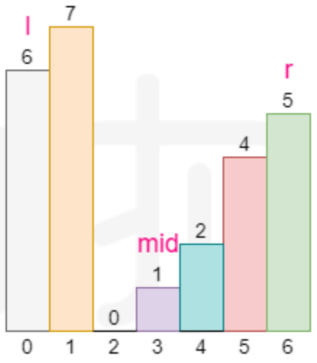
有序，且有范围，递增：===》考虑二分法：



[l, mid] 是有序数组

如果 target = 5，在 [l, mid - 1] 中寻找

如果 target = 2，在 [mid + 1, r] 中寻找



[mid + 1, r] 是有序数组

如果 target = 6，在 [l, mid - 1] 中寻找

如果 target = 4，在 [mid + 1, r] 中寻找

注意：可以根据mid将数组进行切分，根据tagert是否在有序的那部分，

```python
def search(self, nums, target):
        """
        :type nums: List[int]
        :type target: int
        :rtype: int
        """
        if not nums:
            return -1
        left = 0
         right = len(nums) - 1
         while left <= right:
             # 将数组进行切分，为 left到mid，  min到right
             mid = (left + right) // 2
             # 如果mid为targrt 相等，终止循环
             if nums[mid] == target:
                 return mid
             # 判断mid切分的左半部分是否是有序的，是，就判断target是否是在这边 (上
             # 是，查找的范围是[left, mid -1]
             # 否，查找的范围是[mid+1, right]
             if nums[left] <= nums[mid]:
                 if nums[left] <= target < nums[mid]:
                     right = mid - 1
                 else:
                     left = mid + 1
             # 有序的不在左半边，就会在右半边，在从右半边判断target的位置
             else:
                 if nums[mid] < target <= nums[len(nums) -1] :
                     left = mid + 1
                 else:
                     right = mid - 1
         return -1
```