题目：

704. 二分查找

难度 简单 506 收藏 分享 切换为英文 接收动态 反馈

给定一个 `n` 个元素有序的（升序）整型数组 `nums` 和一个目标值 `target` ，写一个函数搜索 `nums` 中的 `target`，如果目标值存在返回下标，否则返回 `-1`。

示例 1:

```
输入：nums = [-1,0,3,5,9,12], target = 9
输出：4
解释：9 出现在 nums 中并且下标为 4
```

示例 2:

```
输入：nums = [-1,0,3,5,9,12], target = 2
输出：-1
解释：2 不存在 nums 中因此返回 -1
```

提示：

1. 你可以假设 `nums` 中的所有元素是不重复的。
2. `n` 将在 `[1, 10000]` 之间。
3. `nums` 的每个元素都将在 `[-9999, 9999]` 之间。

提交结果	执行用时	内存消耗	语言	提交时间	备注
通过	24 ms	26.8 MB	C++	2021/11/24 18:10	添加备注

思路

肯定首先是二分，

因为是有序的，所以说，当当前的值小于目标值，那么肯定，这个数字就在[mid,r]之间的数以内，反之，则是在[l,mid]区间内。所以说每一次二分可以处理掉一半以上的数据。

但是如果我们每次都是 l=mid;或者 r=mid 那么会出现问题

eg： l=1 r =3 mid =2;

在这个时候，假设我们的目标值是 3 的那个数，但是我们现在可以发现，

当 l=mid=2 后，mid=（l+r）/2=2;所以将会陷入死循环，没法完成。
所以在此基础上，每次更新迭代的时候，我将 l=mid+1，因为 mid 是现在检验过的数，所以说 nums[mid]也必然不会是目标值，所以说我们可以将目标值的范围限制在[mid+1,r]之间，所以说可以直接变为 l=mid+1，同理有 r=mid-1
但是在这种情况下，会出现另外一个问题，就是数组下标越界，比如说如果当前的 mid=0 此时的数却还大于目标值时，这个时候我们可以发现其实目标值是不存在的，但是因为 r=mid-1，所以说会有数组越界的情况，那么在 while 循环中加上这个判定就好，所以这个时候有 4 个判定
即 nums[mid]=target 找到目标
l==r 若此点不是目标，则无解
l>=0,r<n 两个边界点