```python
class Solution:
    def search(self, nums: List[int], target: int) -> int:
        low=0
        hign=len(nums)-1
        while hign>=low:
            mid=(low+hign)//2
            guess=nums[mid]
            if guess==target:
                return mid
            elif guess>target:
                hign=mid-1
            elif guess<target:
                low=mid+1
        return -1#return的对其问题!
```

*二分查找

这道题目的前提是数组为有序数组，同时题目还强调数组中无重复元素，因为一旦有重复元素，使用二分查找法返回的元素下标可能不是唯一的，这些都是使用二分法的前提条件，当大家看到题目描述满足如上条件的时候，可要想一想是不是可以用二分法了。

二分查找涉及的很多的边界条件，逻辑比较简单，但就是写不好。例如到底是 `while(left < right)` 还是 `while(left <= right)`，到底是`right = middle`呢，还是要`right = middle - 1`呢?

大家写二分法经常写乱，主要是因为对区间的定义没有想清楚，区间的定义就是不变量。要在二分查找的过程中，保持不变量，就是在while寻找中每一次边界的处理都要坚持根据区间的定义来操作，这就是**循环不变量**规则。

写二分法，区间的定义一般为两种，左闭右闭即[left, right]，或者左闭右开即[left, right)。

下面我用这两种区间的定义分别讲解两种不同的二分写法。

第一种写法，我们定义 target 是在一个在左闭右闭的区间里，也就是[left, right] （这个很重要非常重要）。

区间的定义这就决定了二分法的代码应该如何写，因为定义target在[left, right]区间，所以有如下两点：

- while (left <= right) 要使用 <= ，因为left == right是有意义的，所以使用 <=
- if (nums[middle] > target) right 要赋值为 middle - 1，因为当前这个nums[middle]一定不是target，那么接下来要查找的左区间结束下标位置就是 middle - 1

二分法第二种写法

如果说定义 target 是在一个在左闭右开的区间里，也就是[left, right) ，那么二分法的边界处理方式则截然不同。

有如下两点：

- while (left < right)，这里使用 < ,因为left == right在区间[left, right)是没有意义的
- if (nums[middle] > target) right 更新为
  middle，因为当前nums[middle]不等于target，去左区间继续寻找，而寻找区间是左闭右开区间，所以right更新为middle，即：下一个查询区间不会去比较nums[middle]

回去画个图！