二分法前提：1.有序数组；
2.无重复元素；

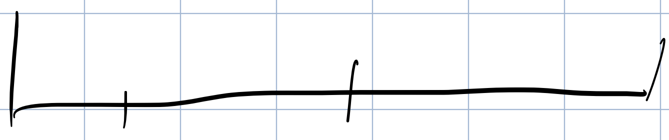
给定一个 n 个元素有序的（升序）整型数组 nums 和一个目标值 target ，写一个函数搜索 nums 中的 target，如果目标值存在返回下标，否则返回 -1。

```
int nums[n]={0};
for(                    )
{
    nums[i]= i;
}
int target =    ;
```

遍历查找：

```
int   targetfind(int* nums, int target, int n)
{
    for(int i=0; i< n; i++)
    {
        if(nums(i)== target)
            return i;
        else
            return -1;
    }
```

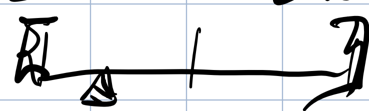
二分查找：(利用有序)



区间：

二种常用 {
- 左闭右闭 [left, right]
- 左闭右开 [left, right)

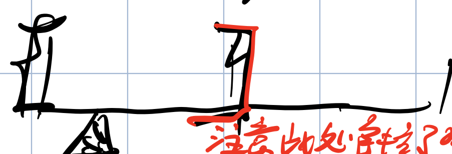
左闭右闭：




```
left = 0
right = num.size - 1;
```

[1]

```
while (left <= right)   // 由于[left, right], 所以有 left = right
{
    middle = (left + right)/2
    if (nums[middle] > target)
        right = middle - 1;   // middle 已经判定过了
                              // (处理过的值不进行二分搜索)
    else (nums[middle] < target)
        left = middle + 1;
    else return middle;
}
return -1;
```



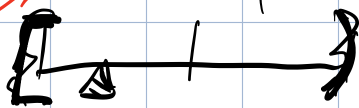
左闭右开：[left, right)

middle

right边界改变

[0, … [size-1], size)

left = 0;
right = num.size ;



while (left < right) // 由于[left, right), [1,1);

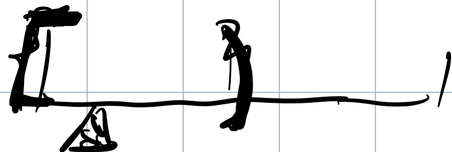
{

middle = (left+right)/2

if (nums[middle] > target)

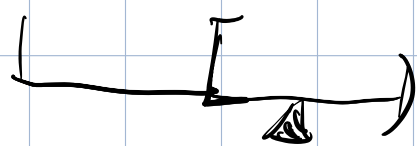
right = middle ;// middle已经判定过了；但左闭右开，right本来就不算判定的，若还是 middle-1 则漏了一个数。

right要更新



else (nums[middle] < target

left = middle+1 ; //



else return middle;

}

return -1;