二分搜索算法的细节在于：

1、mid是加一还是减一

2、while里面到底用<=还是<

二分搜索的框架：

```
int binarySearch(int[] nums,int target){
 int left=0,right=...;
 while(...){
 int mid=left+(right-left)/2;
 if(nums[mid]==target){
 ...
 }else if(nums[mid]<target){
 left=...
 }else if{
  right=....
  }
  }
  return ...
}
```

技巧1：不要出现**else**,而是把所有情况用**else if** 写清楚，可以把所有情况都展现所有细节.

技巧2：用 **mid=left+(right-left)/2**代替**mid=(left+right)/2**;这样就不会有溢出.

基本的二分搜索框架：

```
寻找一个数,条件数组里面的数值为有序(递增的)数组
int binarySearch(int[] nums,int target){
 left=0;
 right=nums.length-1;
 while(left<=right){
 int mid=left+(right-left)/2;
 if(nums[mid]==target) return mid;
 else if(nums[mid]<target)left=mid+1;
 else if(nums[mid]>target)right=mid-1;
  }
  return -1;
}
```

> **为什么right=nums.length-1 而不是right=nums.length?**
> 如果right=nums.length那么就会出现左闭右开的情况[left,right),而right=nums.length-1,则就是左闭右闭的情况,[left,right]
> **为什么left<=right而不是left<right?**

如果是left<right则终止条件为left==right,在左闭右闭区间的情况下还有一个元素[left,right]没有被检索,而left<=right则终止条件为left=right+1,那么不存在元素还没被检索.

寻找左侧边界的二分搜索

```
int left_bound(int[] nums,int target){
 if(nums.length==0) return -1;
 int left=0;
 int right=nums.length;
 while(left<right){
 int mid=left+(right-left)/2;
 if(nums[mid]==target){
 right=mid;
 }else if(nums[mid]<target){
  left=mid+1;
  }else if(nums[mid]>target){
  right=mid;
  }
  }
  return left;
 }
```

1、此时的区间为左闭右开[left,right)则对应的循环终止条件为**left==right**
2、return left;此时**返回的left为数组中比target小的数个数**
3、寻找左侧区间的边界的核心在于if(nums[mid]==target)right=mid;
4、**为什么if(nums[mid]>target) right=mid;而不是right=mid+1;**
因为已经检查过元素nums[mid]了自然要分成两个区间,由于要满足左闭右开的性质因此[left,mid),[mid+1,right)

寻找右侧边界的二分搜索

```
int right_bound(int[] nums,int target){
 if(nums.length==0) return -1;
 int left=0;
 int right=nums.length;
 while(left<right){
 int mid=left+(right-left)/2;
 if(nums[mid]==target){
 left=mid+1;
 }else if(nums[mid]<target){
  left=mid+1;
  }else if(nums[mid]>target){
  right=mid;
  }
  }
```

```
 return left-1;//return right-1;
}
```

综上所述可以对二分搜索寻找某一个值或者寻找左侧边界或者右侧边界进行统一如下所示:

```
if(num.length==0) return -1;
int left=0;
int right=nums.length-1;
while(left<=right){
 int mid=left+(right-left)/2;
 if(nums[mid]==target){
 //寻找某个值 return mid;
 //寻找左侧边界 right=mid-1;
 //寻找右侧边界 left=mid+1;
  }else if(nums[mid]<target){
  left=mid+1;
  }else if(nums[mid]>target){
  right=mid-1;
  }
  //寻找某个值return -1;
  /**
  寻找左侧边界:当数组内所有的值都比target的值大时,left=nums.length所以要检测边界溢出
  if(left>=nums.length||nums[left]!=target)
  return -1
  return left;
  */
  /***
  寻找右侧边界:当数组内所有的值都比target的值小时, right=-1,所以要检测右侧溢出
  if(right<0||nums[right]!=target)
  return -1;
  return right;
  ***/
}
```