把问题抽象为Binary Search步骤：

1、确定 **x, f(x), target** 分别是什么，并写出函数 **f** 的代码。

x一般是问题要求得东西，target是约束的变量

2、找到 **x** 的取值范围作为⼆分搜索的搜索区间，初始化 **left** 和 **right** 变量。

left, right， 可能是最小，最大的integer，array中最小或者最大的变量

3、根据题⽬的要求，确定应该使⽤搜索左侧还是搜索右侧的⼆分搜索算法，写出解法代码。

注意f(x)是monotonic non-increasing or non-decreasing

**Seach Range [left, right)**

**int binary\_search** (**int**[] nums, **int** target) {

**if** (nums.length == 0)

**return** -1;

**int** left = 0, right = nums.length;

**while** (left < right) { -> **[a, a) 时break，不漏区间**

**int** mid = left + (right - left) / 2;

**if** (nums[mid] == target) {

return mid; // 注意

} **else** **if** (nums[mid] < target) {

left = mid + 1;

} **else** **if** (nums[mid] > target) {

right = mid;

}

}

**return** -1; // 注意

}

/\*

以单调递增为例(non-decreasing fuction)

\*/

**int** left\_bound(**int**[] nums, **int** target) {

**if** (nums.length == 0)

**return** -1;

**int** left = 0, right = nums.length;

**while** (left < right) {

**int** mid = left + (right - left) / 2;

**if** (nums[mid] == target) {

right = mid; // 注意

} **else** **if** (nums[mid] < target) {

left = mid + 1;

} **else** **if** (nums[mid] > target) {

right = mid;

}

}

if (left >= length || nums[left] != target) {

return -1;

}

**return** left; // 注意

}

**int** right\_bound(**int**[] nums, **int** target) {

**if** (nums.length == 0)

**return** -1;

**int** left = 0, right = nums.length;

**while** (left < right) {

**int** mid = left + (right - left) / 2;

**if** (nums[mid] == target) {

left = mid + 1; // 注意

} **else** **if** (nums[mid] < target) {

left = mid + 1;

} **else** **if** (nums[mid] > target) {

right = mid;

}

}

if (left-1 <= 0 || nums[left-1] != target) {

return -1;

}

**return** left - 1; // 注意

}