二分查找的前提

单调性 1. 目标函数单调性（单调递增或者递减）

上下界 2. 存在上下界（bounded）

索引 3. 能够通过索引访问（index accessible）

↳ 可以用下标进行访问

二分查找：在有序的里面进行查找

（无序的只能从头到尾遍历）

↳ 是有序的

↳ 可以通过判断它的某些特征排除掉（前半部分/后半）

↳ 要 <单调的>

没有上下界的空间可能是无穷大的（则没法往中间扩）

代码模版

```java
// Java
public int binarySearch(int[] array, int target) {
    int left = 0, right = array.length - 1, mid;
    while (left <= right) {
        mid = (right - left) / 2 + left;

        if (array[mid] == target) {
            return mid;
        } else if (array[mid] > target) {
            right = mid - 1;
        } else {
            left = mid + 1;
        }
    }

    return -1;
}
```

中间值

判断mid是否等于target，[然后来break或者是return这个result (python)]

先用==，只要等于的话马上return即可

左界：0　　右界：数组长度-1

（左右的下标值）

思路

假设升序排列

如果target > array[mid] →说明在右侧

↳ 继续向右查找

↳ left，就把左界向右进行移动，变成mid+1

否则的话说明在左侧，那么右界的话就要向左移动，变成mid-1

[左下界 & 右下界为整型的情况下]

Interger

在有些时候可能没数的情况下，

就没有'+1' or '-1'，

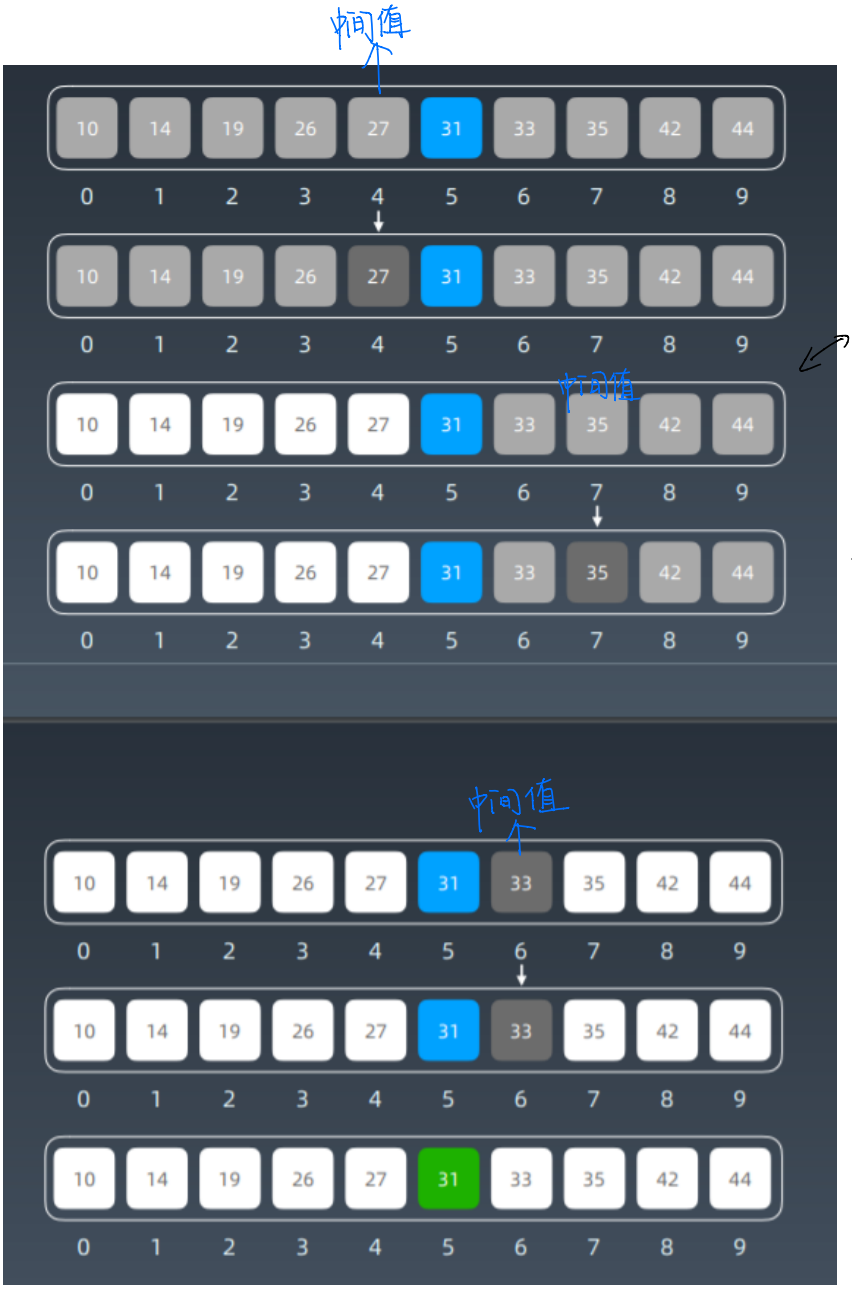
直接等于mid即可

示例

在递增数组里　　单调递增

[10, 14, 19, 26, 27, 31, 33, 35, 42, 44]

查找：31



31与27进行对比，大于27

说明可以排除前半部分，

继续从后半部分进行查找

前半部分白色就不用考虑了，

继续在灰色部分查找，

灰色的中间值部分继续算

mid的话 → 35

查35，比35小就说明

继续在35的右侧，35的左侧

开始找

mid值算出来等于33

小于33

继续往左边找

找到

二分查找：

由左右两个边界不断地向中间进行夹逼的过程

里→又因为这个数组本身它是单调递增的，所以每次可以

排除一半 (数组实现)