Binary Search

35. Search Insert Position

解法一：二分查找

时间复杂度：O(log(n))； 空间复杂度：O(1)

直接二分查找，建立左闭右开的区间，这样区间内所有的数字都是有可能的，最后当左边和右边的边界到达同一个位置的时候区间的大小为0，这个时候返回low（或者high）即可，如果查找到了则是这个数字，否则就是需要插入的位置。

136. Single Number

540. Single Element in a Sorted Array

解法一：顺序查找

时间复杂度：O(n)； 空间复杂度：O(1)

找第一个和后一位不一样的数字，或是在最后一位落单的数字

解法二：二分查找

时间复杂度：O(log(n))； 空间复杂度：O(1)

1. 当mid为偶数且下一位和mid位数字相同或 mid为奇数且下一位和mid不同

   此时target在左侧

2. 其他情况

   此时target在右侧/当前位置

解法三：XOR

时间复杂度：O(n)； 空间复杂度：O(1)

参考

将所有的数字一起异或，相同的数字经过异或会变成零，最后留下的就是落单的数字

```
class Solution {
    public int singleNumber(int[] nums) {
        int ans = 0;
        for (int i = 0; i < nums.length; i++)
            ans = ans ^ nums[i];
        return ans;
    }
}
```

---

275. H-Index-II

解法一：二分查找

时间复杂度：O(log(n))； 空间复杂度：O(1)

二分查找，比较当前点所代表的paper数量（len - mid，因为需要计算这张加上后面一共有多少paper）是否大于等于这个paper的citation数量。制作一个左开右闭区间。

1. 如果当前点的citation数量和paper数量一样，说明这个就是我们要找的值。例如一个人有四张paper的citation数量都大于等于4，再加paper也不可能有多的citation数量为5的paper。
2. 如果当前点的citation数量大于paper数量，说明还需要往前面找，就把high挪到当前mid
3. 否则因为1和2已经排除，说明这个点也不是正确答案，把low挪到mid + 1

---

278. First Bad Version

解法一：二分查找

时间复杂度：O(log(n))； 空间复杂度：O(1)

二分查找，如果当前找到的点是Bad Version，则往前找，否则往后找（包括当前这个）。最后当左侧pointer大于等于右侧pointer时，返回左侧pointer

所指的值。

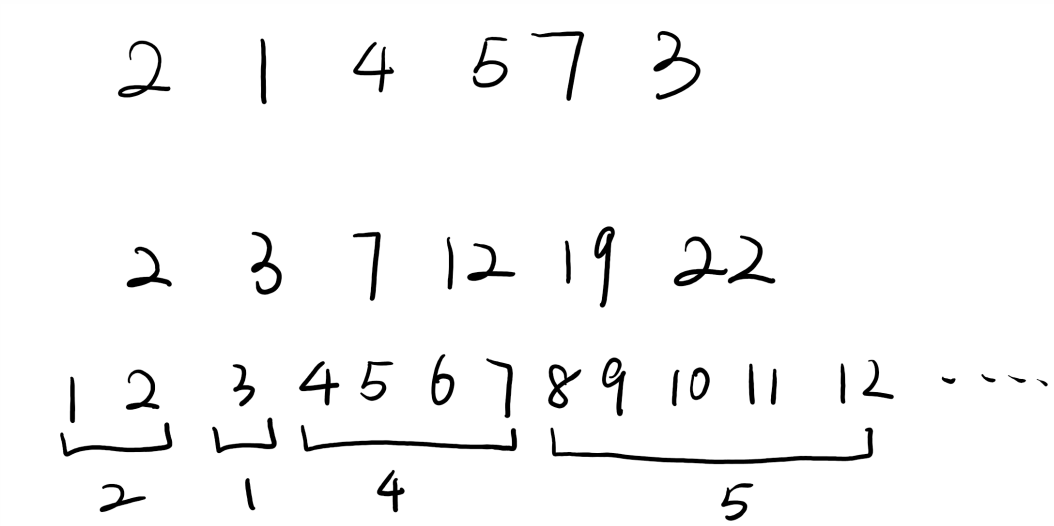
建立一个左闭右开区间，最后当区间大小为0（即left == right）的时候，判定找到了，返回left或right皆可。

528. Random Pick with Weight

解法一：累积概率 + 二分查找

参考

因为需要每个物品被选择的概率和重量相同，但是想要让物品的权重有规律且不需要被存储，那么可以用累积概率的方式，计算整个数组的累积概率分布，这样的话整个数组是升序排列的且每两个数字之间的间隔的大小就是后一个数字被选中的权重。



得到随机数之后，需要找最前面的一个累积和大于这个数值的数组Index，这个Index对应的数就是被选中的数字。

解法二：累积概率 + 顺序查找

用顺序查找来找到指定的数字。