## 701.Binary Search

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

        return -1;
    }
}
```

## 162.Find Peak Element

```
class Solution {
    public int findPeakElement(int[] nums) {
        if(nums.length==1)return 0;
        if(nums[0]>nums[1])return 0;
            if(nums[nums.length-1]>nums[nums.length-2])return nums.length-1;
            for(int i=1;i<nums.length-1;i++){
                if(nums[i-1]<nums[i] && nums[i]>nums[i+1])return i;
        }return 0;
    }
}
```

## 33.search in rotated sorted array

```
class Solution {
    public int search(int[] nums, int target) {
        int l=0,r=nums.length-1;
        while(l<=r){
            int m=(r+l)/2;
            if(target==nums[m])return m;
            if(nums[l]<=nums[m]){
                if(target<nums[m] && target >= nums[l]){
                    r=m-1;
                }else l=m+1;
            }else{
                if(nums[r]>=target && nums[m]<target) l=m+1;
                else r=m-1;
            }
        }return -1;
    }
}
```

## 153.Find Minimum In Rotated Sorted Array

```
class Solution {
    public int findMin(int[] nums) {
        for(int i=0,j=i+1;j<nums.length;i++,j++)
        {
                if(nums[i]>nums[j])
            {
                return nums[j];
            }
        }
        return nums[0];
    }
}
```

## 875.koko Eating Bananas

```
class Solution {
    public int minEatingSpeed(int[] piles, int h) {
        int l=1,r= max(piles);
        while(l<r){
            int mid=(r+l)/2;
            int hours=0;
            for(int pile:piles){
                hours+=(pile/mid);//hours+=(pile+mid-1)/mid;
                if(pile%mid!=0)hours++;
            }
            if(hours<=h)
                r=mid;
            else
                l=mid+1;
        }
            return l;
    }
public int max(int []piles)
{
    int max= Integer.MIN_VALUE;
    for(int pile:piles)//hours+=math.ceill((double)pile/m);
    {
        max=Math.max(max,pile);
    }
    return max;
}
}
```

## 1046.Last Stone Weight

```
class Solution {
    public int lastStoneWeight(int[] stones) {
        PriorityQueue<Integer> d = new PriorityQueue<>((a,b)->b-a);
        for(int s : stones)d.offer(s);
        while(d.size()>1){
            int s1=d.poll();
            int s2=d.poll();
            if(s1!=s2) d.offer(s1-s2);
        }return(d.isEmpty())?0:d.poll();
    }
}
```