4. Median of Two Sorted Arrays

```
class Solution:
    def findMedianSortedArrays(self, nums1: List[int], nums2: List[int]) -> float:
        ans = sorted(nums1+nums2)
        if len(ans) %2 ==1:
            return  ans[len(ans)//2]
        else:
            return (ans[(len(ans)//2) -1] + ans[len(ans)//2])/2
```

33. Search in Rotated Sorted Array

https://www.youtube.com/watch?v=U8XENwh8Oy8 (https://www.youtube.com/watch?v=U8XENwh8Oy8)

```
class Solution:
    def search(self, nums: List[int], target: int) -> int:
        left,right  = 0 , len(nums)-1

        while left <= right:
            mid = (left+right)//2
            if nums[mid] == target:
                return mid
            # left sorted portion
            elif nums[left] <= nums[mid]:
                if target > nums[mid] or target < nums[left] :
                    left = mid+1
                else:
                    right = mid-1
            # Right sorted arr
            else:
                if target < nums[mid] or nums[right] < target:
                    right = mid - 1
                else:
                    left = mid + 1
        return -1
```

34. Find First and Last Position of Element in Sorted Array

```
class Solution:
    def searchRange(self, nums: List[int], target: int) -> List[int]:
        first = last = -1
        i,j = 0,len(nums)-1
        while i <= j:
            mid  = (i+j)//2
            if nums[mid] == target:
                i = mid
                j = mid
                while i>=0 and nums[i] == target:
                    i-=1
                while j<= len(nums)-1 and nums[j] == target:
                    j+=1
                return i+1,j-1
            elif nums[mid] < target:
                i = mid+1
            else:
                j = mid-1
        return [-1,-1]
```

35. Search Insert Position

```
class Solution:
    def searchInsert(self, nums: List[int], target: int) -> int:
        i = 0
        j = len(nums)-1
        while i<=j:
            mid = (i+j)//2
            if nums[mid] == target:
                return mid
            elif nums[mid] < target:
                i = mid+1
            else:
                j = mid-1
        return i
```

74. Search a 2D Matrix

```
class Solution:
    def searchMatrix(self, matrix: List[List[int]], target: int) -> bool:
        left = 0
        right = len(matrix)-1
        def find_target(arr):
            left,right = 0,len(arr)-1
            while left <= right:
                mid = (left+right)//2
                if arr[mid]== target:return True
                elif arr[mid] < target:left = mid+1
                else:right = mid-1
            return False
        while left <= right:
            mid = (left+right)//2
            if matrix[mid][0] <= target and target <= matrix[mid][-1]:
                if find_target(matrix[mid]):return True
                return False
            elif matrix[mid][0]> target:right = mid-1
            elif matrix[mid][-1]< target:left = mid+1
        return False
```

81. Search in Rotated Sorted Array II

```
class Solution:
    def search(self, nums: List[int], target: int) -> bool:
        left, right = 0, len(nums) - 1

        while left <= right:
            mid = (left + right) // 2

            if nums[mid] == target:
                return True

            if nums[mid] == nums[left]:
                left += 1
                continue

            if nums[left] <= nums[mid]:
                if nums[left] <= target < nums[mid]:
                    right = mid - 1
                else:
                    left = mid + 1
            else:
                if nums[mid] < target <= nums[right]:
                    left = mid + 1
                else:
                    right = mid - 1

        return False
```

153. Find Minimum in Rotated Sorted Array

```
class Solution:
    def findMin(self, nums: List[int]) -> int:
        left,right = 0,len(nums)-1
        while left < right:
            mid = (left+right)//2
            if nums[mid] > nums[right] :
                left = mid +1
            else:
                right = mid
        return nums[left]
```

162. Find Peak Element

```
class Solution:
    def findPeakElement(self, nums: List[int]) -> int:
        peak = 0
        nums.append(-float("inf"))
        n = len(nums)
        if n < 3 :
            if n==1:return 0
            if nums[0] < nums[1] : return 1
            else:return 0
        for i in range(1,n-1):
            if nums[i-1] < nums[i] > nums[i+1]:
                return i
        return peak
```

540. Single Element in a Sorted Array

```
class Solution:
    def singleNonDuplicate(self, nums: List[int]) -> int:
        n = len(nums)
        left,right = 0,n-1

        while left<right:
            mid = (left+right)//2
            if mid%2==1:
                mid-=1
            if nums[mid] == nums[mid+1]:
                left = mid+2
            else:
                right = mid
        return nums[left]
```

704. Binary Search

```
class Solution:
    def search(self, nums: List[int], target: int) -> int:
        n = len(nums)
        i =0
        j = n-1
        while i <= j:
            mid = (i+j)//2
            if nums[mid] == target:
                return mid
            elif nums[mid] < target:
                i = mid+1
            elif nums[mid] > target:
                j = mid - 1
        return -1
```

875. Koko Eating Bananas

```
class Solution:
    import math
    def minEatingSpeed(self, piles: List[int], h: int) -> int:
        left,right  = 1,max(piles)
        def find_hour(piles,mid):
            total =0
            for i in range(len(piles)):
                total += math.ceil(piles[i]/mid)
            return total
        while left <= right:
            mid = (left+right)//2
            totalhours = find_hour(piles,mid)
            if totalhours <= h:
                right = mid-1
            else:
                left = mid+1
        return left
```

1011. Capacity To Ship Packages Within D Days

```
class Solution:
    def shipWithinDays(self, weights: List[int], days: int) -> int:
        left,right = max(weights),sum(weights)
        def find_days(arr,weight):
            s = 0
            d=1
            for i in arr:
                if s+i>weight:
                    s = i
                    d+=1
                else:
                    s+=i
            return d

        while left <= right:
            mid = (left+right)//2
            day = find_days(weights,mid)
            if day <= days:
                right = mid-1
            else:
                left = mid+1
        return left
```

1283. Find the Smallest Divisor Given a Threshold

```
class Solution:
    import math
    def smallestDivisor(self, nums: List[int], threshold: int) -> int:
        left,right = 1,max(nums)
        def find_total(nums,num):
            s= 0
            for i in nums:
                s+= math.ceil(i/num)
            return s

        min_ans = right
        while left <= right:
            mid = (left+right)//2
            summ = find_total(nums,mid)
            if summ<=threshold:
                min_ans = min(min_ans,mid)
            if summ <= threshold:
                right = mid-1
            else:
                left = mid+1
        return min_ans
```

1482. Minimum Number of Days to Make m Bouquets

```
class Solution:
    def minDays(self, bloomDay: List[int], m: int, k: int) -> int:
        left,right = min(bloomDay),max(bloomDay)
        if len(bloomDay) < m*k:
            return -1
        def find_bouquet(arr,day):
            total = 0
            temp = 0
            for i in range(len(arr)):
                if arr[i] <=day:
                    temp+=1
                    if temp == k:
                        temp = 0
                        total+=1
                else:
                    temp = 0
            return total
        while left<=right:
            mid = (left+right)//2
            cor = find_bouquet(bloomDay,mid)
            if cor >= m:
                right = mid-1
            else:
                left = mid+1
        return left
```

1539. Kth Missing Positive Number

```
def findKthPositive(self, arr: List[int], k: int) -> int:
        left = 0
        right = len(arr)
        while left < right:
                mid = (left+right)//2
                if arr[mid]-1 -mid < k:
                        left = mid+1
                else:
                        right = mid
        return right+k
```

1901. Find a Peak Element II

```
class Solution:
    def findPeakGrid(self, mat: List[List[int]]) -> List[int]:
        def find_row(arr,col,low,high):
            r = -1
            value = -1
            for row in range(0,low):
                if arr[row][col] > value:
                    value = arr[row][col]
                    r = row
            return r

        left,right = 0,len(mat[0])-1
        n = len(mat)
        m = len(mat[0])
        while left <= right:
            mid = (left+right)//2
            row = find_row(mat,mid,n,m) # find the row and col with max_ele
            l = -1 if mid == 0 else mat[row][mid-1]
            r = -1 if mid == m-1 else mat[row][mid+1]
            if l < mat[row][mid] > r:
                return [row,mid]
            elif l > mat[row][mid]:
                right = mid -1
            else:
                left = mid+1
        return [-1,-1]
```