Chapter 3 Binary Search

https://leetcode.com/tag/binary-search/

Search Insert Position
Given a sorted array and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You may assume no duplicates in the array.

Example 1:

```
Input: [1,3,5,6], 5
Output: 2
```

Example 2:

```
Input: [1,3,5,6], 2
Output: 1
```

Example 3:

```
Input: [1,3,5,6], 7
Output: 4
```

Example 4:

```
Input: [1,3,5,6], 0
Output: 0
```

Binary Search - Found the upper bound

```
public int searchInsert(int[] A, int target) {
    int low = 0, high = A.length-1;
    while(low<=high){
        int mid = (low+high)/2;
        if(A[mid] == target) return mid;
        else if(A[mid] > target) high = mid-1;
        else low = mid+1;
    }
    return low;
}
```

[Search in Rotated Sorted Array II]

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand. (i.e., [0,0,1,2,2,5,6] might become [2,5,6,0,0,1,2]).

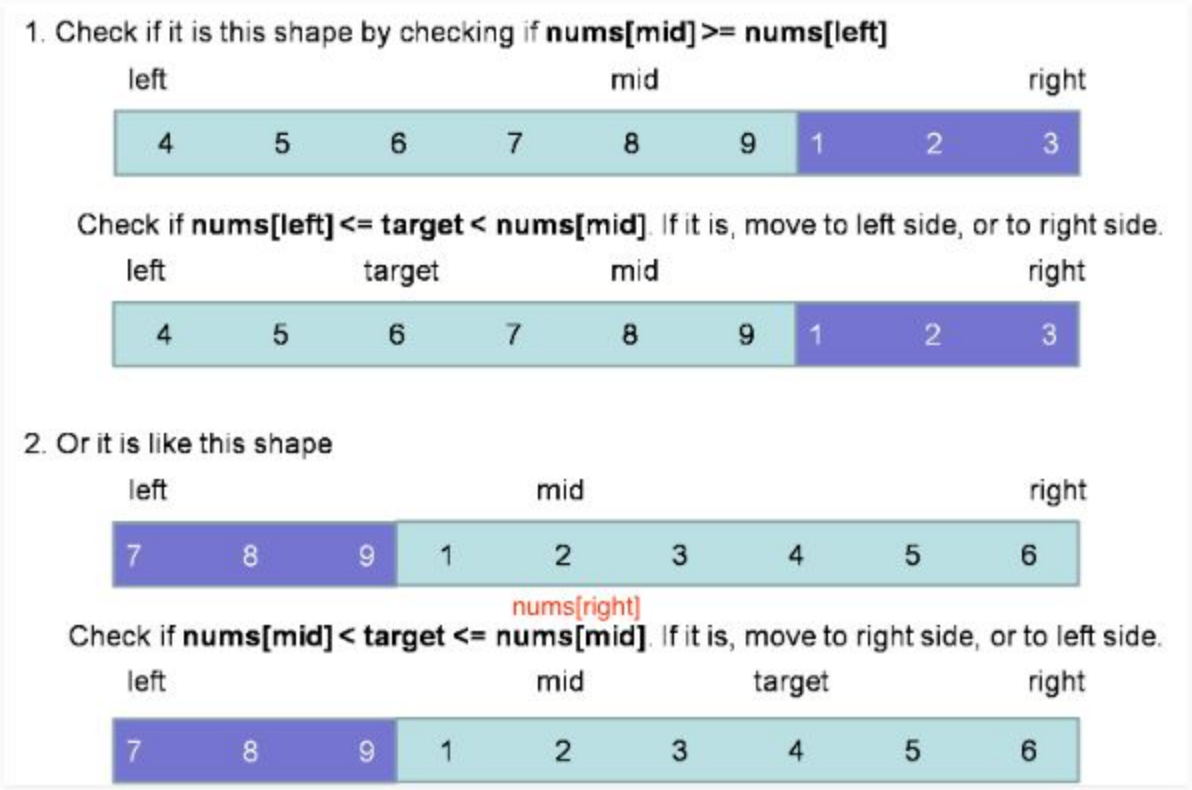
You are given a target value to search. If found in the array return true, otherwise return false.

Example 1:

```
Input: nums = [2,5,6,0,0,1,2], target = 0
Output: true
```

Example 2:

```
Input: nums = [2,5,6,0,0,1,2], target = 3
Output: false
```



http://www.allenlipeng47.com/blog/index.php/2016/08/10/search-in-rotated-sorted-array-i-ii/

```
public boolean search(int[] nums, int target) {
    // note here end is initialized to len instead of (len-1)
    int start = 0, end = nums.length;
```

```
    while (start < end) {
        int mid = (start + end) / 2;
        if (nums[mid] == target) return true;
        if (nums[mid] > nums[start]) { // nums[start..mid] is sorted
            // check if target in left half
            if (target < nums[mid] && target >= nums[start]) end = mid;
            else start = mid + 1;
        } else if (nums[mid] < nums[start]) { // nums[mid..end] is sorted
            // check if target in right half
            if (target > nums[mid] && target < nums[start]) start = mid +
1;
            else end = mid;
        } else { // have no idea about the array, but we can exclude
nums[start] because nums[start] == nums[mid]
            start++;
        }
    }
    return false;
}
```

[Search a 2D Matrix]

Write an efficient algorithm that searches for a value in an m x n matrix. This matrix has the following properties:

- Integers in each row are sorted from left to right.
- The first integer of each row is greater than the last integer of the previous row.

Example 1:

```
Input:
matrix = [
  [1,   3,  5,  7],
  [10, 11, 16, 20],
  [23, 30, 34, 50]
]
target = 3
Output: true
```

Example 2:

```
Input:
matrix = [
  [1,   3,  5,  7],
  [10, 11, 16, 20],
```

```
  [23, 30, 34, 50]
]
target = 13
Output: false
```

把整个 matrix 当作一个大的 sorted list

```
public boolean searchMatrix(int[][] matrix, int target) {
    if (matrix == null || matrix.length == 0 || matrix[0].length == 0) {
        return false;
    }
    int row = 0;
    int col = matrix[0].length - 1;
    while (row < matrix.length && col >= 0) {
        if (matrix[row][col] == target) {
            return true;
        } else if (matrix[row][col] < target) {
            row++;
        } else {
            col--;
        }
    }
    return false;
}
```

Find the Smallest Divisor Given a Threshold

Given an array of integers nums and an integer threshold, we will choose a positive integer divisor and divide all the array by it and sum the result of the division. Find the smallest divisor such that the result mentioned above is less than or equal to threshold.

Each result of division is rounded to the nearest integer greater than or equal to that element. (For example: 7/3 = 3 and 10/2 = 5). It is guaranteed that there will be an answer.

nums.length <= threshold <= 10^6

Example 1:

```
Input: nums = [1,2,5,9], threshold = 6
Output: 5
Explanation: We can get a sum to 17 (1+2+5+9) if the divisor is 1.
If the divisor is 4 we can get a sum to 7 (1+1+2+3) and if the divisor is 5
the sum will be 5 (1+1+1+2).
```

Example 2:

```
Input: nums = [2,3,5,7,11], threshold = 11
Output: 3
```

Example 3:

```
Input: nums = [19], threshold = 5
Output: 4
```

sum > threshold => the divisor is too small.
sum <= threshold => the divisor is big enough

```
public int smallestDivisor(int[] A, int threshold) {
    int left = 1, right = (int)1e6;
    while (left < right) {
        int m = (left + right) / 2, sum = 0;
        for (int i : A)
            sum += (i + m - 1) / m;
        if (sum > threshold)
            left = m + 1;
        else
            right = m;
    }
    return left;
}
```

[Sum of Mutated Array Closest to Target]
Given an integer array arr and a target value target, return the integer value such that when we change all the integers larger than value in the given array to be equal to value, the sum of the array gets as close as possible (in absolute difference) to target.

In case of a tie, return the minimum such integer.

Notice that the answer is not neccesarilly a number from arr.
Example 1:

```
Input: arr = [4,9,3], target = 10
Output: 3
Explanation: When using 3 arr converts to [3, 3, 3] which sums 9 and that's
the optimal answer.
```

Example 2:

```
Input: arr = [2,3,5], target = 10
Output: 5
```

Example 3:

```
Input: arr = [60864,25176,27249,21296,20204], target = 56803
Output: 11361
```

可以取的值是 1 到 max(val in array, target), 在这之中 binary search

```
public int findBestValue(int[] arr, int target) {
    int l, r, mi, s=0, m=-1;
    for(int v:arr) { s += v; m=Math.max(m,v); }

    if(s<=target) return m; // return the max value since we will keep all
nums as is

    for(l=1,r=m;l<r;) {
        mi=(l+r)/2;
        s=0;
        for(int v:arr) s += (v>mi)?mi:v;
        if(s>=target) r=mi;
        else          l=mi+1;
    }
    // check if we are 1 step off the target
    int s1=0,s2=0;
    for(int v:arr) {
        s1 += (v>l)?(l):v;
        s2 += (v>l-1)?(l-1):v;
    }

    return (Math.abs(s2-target) <= Math.abs(s1-target)) ? l-1 : l;
}
```

[Capacity To Ship Packages Within D Days]
A conveyor belt has packages that must be shipped from one port to another within D days.

The i-th package on the conveyor belt has a weight of weights[i]. Each day, we load the ship with packages on the conveyor belt (in the order given by weights). We may not load more weight than the maximum weight capacity of the ship.

Return the least weight capacity of the ship that will result in all the packages on the conveyor belt being shipped within D days.

Example 1:

```
Input: weights = [1,2,3,4,5,6,7,8,9,10], D = 5
Output: 15
Explanation:
A ship capacity of 15 is the minimum to ship all the packages in 5 days
like this:
1st day: 1, 2, 3, 4, 5
2nd day: 6, 7
3rd day: 8
4th day: 9
5th day: 10
```

Note that the cargo must be shipped in the order given, so using a ship of capacity 14 and splitting the packages into parts like (2, 3, 4, 5), (1, 6, 7), (8), (9), (10) is not allowed.

Example 2:

```
Input: weights = [3,2,2,4,1,4], D = 3
Output: 6
Explanation:
A ship capacity of 6 is the minimum to ship all the packages in 3 days like
this:
1st day: 3, 2
2nd day: 2, 4
3rd day: 1, 4
```

Example 3:

```
Input: weights = [1,2,3,1,1], D = 4
Output: 3
Explanation:
1st day: 1
2nd day: 2
3rd day: 3
4th day: 1, 1
```

Note:
1 <= D <= weights.length <= 50000
1 <= weights[i] <= 500

Capacity 的取值范围: max weight of weights ~ sum of weights

```
public int shipWithinDays(int[] weights, int D) {
    int left = 0, right = 0;
    for (int w: weights) {
        left = Math.max(left, w);
        right += w;
    }
    while (left < right) {
        int mid = (left + right) / 2, need = 1, cur = 0;
        for (int w: weights) {
            if (cur + w > mid) {
                need += 1;
                cur = 0;
            }
            cur += w;
        }
        if (need > D) left = mid + 1;
        else right = mid;
    }
    return left;
}
```

Koko Eating Bananas

Koko loves to eat bananas. There are N piles of bananas, the i-th pile has piles[i] bananas. The guards have gone and will come back in H hours.

Koko can decide her bananas-per-hour eating speed of K. Each hour, she chooses some pile of bananas, and eats K bananas from that pile. If the pile has less than K bananas, she eats all of them instead, and won't eat any more bananas during this hour.

Koko likes to eat slowly, but still wants to finish eating all the bananas before the guards come back.

Return the minimum integer K such that she can eat all the bananas within H hours.

Example 1:

```
Input: piles = [3,6,7,11], H = 8
Output: 4
```

Example 2:

```
Input: piles = [30,11,23,4,20], H = 5
Output: 30
```

Example 3:

```
Input: piles = [30,11,23,4,20], H = 6
Output: 23
```

Note:
1 <= piles.length <= 10^4
piles.length <= H <= 10^9
1 <= piles[i] <= 10^9

K 的取值范围: 1 ~ max pile of piles

```
public int minEatingSpeed(int[] piles, int H) {
    int l = 1, r = 1000000000;
    while (l < r) {
        int m = (l + r) / 2, total = 0;
        for (int p : piles) total += (p + m - 1) / m;
        if (total > H) l = m + 1; else r = m;
    }
    return l;
}
```