Step 4 : Binary Search [1D, 2 D Arrays, Search space]

4.1 [704. Binary Search]

Given an array of integers nums which is sorted in ascending order, and an integer target, write a function to search target in nums. If target exists, then return its index. Otherwise, return -1.

You must write an algorithm with O(log n) runtime complexity.

Example 1:

Input: nums = [-1,0,3,5,9,12], target = 9

Output: 4

Explanation: 9 exists in nums and its index is 4

Example 2:

Input: nums = [-1,0,3,5,9,12], target = 2

Output: -1

Explanation: 2 does not exist in nums so return -1

Constraints:

- 1 <= nums.length <= 10^4
- -10^4 < nums[i], target < 10^4
- All the integers in nums are **unique**.
- nums is sorted in ascending order.

4.2 **Implement Lower Bound**

Difficulty: **Easy**Accuracy: **50.04%**Submissions: **16K+**Points: **2**

Given a sorted array **arr[]** and a number **target**, the task is to find the **lower bound** of the **target** in this given array. The **lower bound** of a number is defined as the smallest **index** in the sorted array where the element is **greater than or equal to** the given number.

Note: If all the elements in the given array are smaller than the **target**, the lower bound will be the length of the array.

Examples :

Input: arr[] = [2, 3, 7, 10, 11, 11, 25], target = 9

Output: 3

Explanation: 3 is the smallest index in arr[] where element (arr[3] = 10) is greater than or equal to 9.

Input: arr[] = [2, 3, 7, 10, 11, 11, 25], target = 11

Output: 4

Explanation: 4 is the smallest index in arr[] where element (arr[4] = 11) is greater than or equal to 11.

Input: arr[] = [2, 3, 7, 10, 11, 11, 25], target = 100

Output: 7

Explanation: As no element in arr[] is greater than 100, return the length of array.

Constraints:
1 ≤ arr.size() ≤ 10^6
1 ≤ arr[i] ≤ 10^6
1 ≤ target ≤ 10^6

4.3 **Implement Upper Bound**

Difficulty: **Easy**Accuracy: **57.32%**Submissions: **9K+**Points: **2**

Given a **sorted** array **arr[]** and a number **target**, the task is to find the **upper** bound of the **target** in this given array.

The **upper bound** of a number is defined as the smallest **index** in the sorted array where the element is greater than the given number.

Note: If all the elements in the given array are smaller than or equal to the **target**, the upper bound will be the length of the array.

Examples :

Input: arr[] = [2, 3, 7, 10, 11, 11, 25], target = 9

Output: 3

Explanation: 3 is the smallest index in arr[], at which element (arr[3] = 10) is larger than 9.

Input: arr[] = [2, 3, 7, 10, 11, 11, 25], target = 11

Output: 6

Explanation: 6 is the smallest index in arr[], at which element (arr[6] = 25) is larger than 11.

Input: arr[] = [2, 3, 7, 10, 11, 11, 25], target = 100

Output: 7

Explanation: As no element in arr[] is greater than 100, return the length of array.

Constraints:
$1 \le arr.size() \le 10^6$
$1 \le arr[i] \le 10^6$
$1 \le target \le 10^6$

4.4 [35. Search Insert Position]

Solved

Easy

Topics

Companies

Given a sorted array of distinct integers 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 must write an algorithm with O(log n) runtime complexity.

Example 1:

Input: nums = [1,3,5,6], target = 5

Output: 2

Example 2:

Input: nums = [1,3,5,6], target = 2

Output: 1

Example 3:

Input: nums = [1,3,5,6], target = 7

Output: 4

Constraints:

- 1 <= nums.length <= 10^4
- -10^4 <= nums[i] <= 10^4
- nums contains **distinct** values sorted in **ascending** order.
- -10^4 <= target <= 10^4

4.5 **Floor in a Sorted Array**

Difficulty: **Easy**Accuracy: **33.75%**Submissions: **482K+**Points: **2**Average Time: **30m**

Given a sorted array **arr[]** and an integer **x**, find the index (0-based) of the largest element in arr[] that is less than or equal to x. This element is called the **floor** of x. If such an element does not exist, return -1.

Note: In case of multiple occurrences of ceil of x, return the index of the last occurrence.

Examples

Input: arr[] = [1, 2, 8, 10, 10, 12, 19], x = 5

Output: 1

Explanation: Largest number less than or equal to 5 is 2, whose index is 1.

Input: arr[] = [1, 2, 8, 10, 10, 12, 19], x = 11

Output: 4

Explanation: Largest Number less than or equal to 11 is 10, whose indices are 3 and 4. The index of last occurrence is 4.

Input: arr[] = [1, 2, 8, 10, 10, 12, 19], x = 0
Output: -1

Explanation: No element less than or equal to 0 is found. So, output is -1.

Constraints:
$1 \le arr.size() \le 10^6$
$1 \le arr[i] \le 10^6$
$0 \le x \le arr[n-1]$

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

Solved

Medium

Topics

Companies

Given an array of integers nums sorted in non-decreasing order, find the starting and ending position of a given target value.

If target is not found in the array, return [-1, -1].

You must write an algorithm with O(log n) runtime complexity.

Example 1:

Input: nums = [5,7,7,8,8,10], target = 8

Output: [3,4]

Example 2:

Input: nums = [5,7,7,8,8,10], target = 6

Output: [-1,-1]

Example 3:

Input: nums = [], target = 0

Output: [-1,-1]

Constraints:

- 0 <= nums.length <= 10^5
- -10^9 <= nums[i] <= 10^9
- nums is a non-decreasing array.
- -10^9 <= target <= 10^9

4.7 **Number of occurrence**

Difficulty: **Easy**Accuracy: **59.34%**Submissions: **328K+**Points: **2**Average Time: **20m**

Given a **sorted** array, **arr[]** and a number **target**, you need to find the number of occurrences of **target** in **arr[]**.

Examples :

Input: arr[] = [1, 1, 2, 2, 2, 2, 3], target = 2

Output: 4

Explanation: target = 2 occurs 4 times in the given array so the output is 4.

Input: arr[] = [1, 1, 2, 2, 2, 2, 3], target = 4

Output: 0

Explanation: target = 4 is not present in the given array so the output is 0.

Input: arr[] = [8, 9, 10, 12, 12, 12], target = 12

Output: 3

Explanation: target = 12 occurs 3 times in the given array so the output is 3.

Constraints:
$1 \le arr.size() \le 10^6$
$1 \le arr[i] \le 10^6$
$1 \le target \le 10^6$

4.8 **. Search in Rotated Sorted Array**

There is an integer array nums sorted in ascending order (with **distinct** values).

Prior to being passed to your function, nums is **possibly rotated** at an unknown pivot index k (1 <= k < nums.length) such that the resulting array is [nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]] (**0-indexed**). For example, [0,1,2,4,5,6,7] might be rotated at pivot index 3 and become [4,5,6,7,0,1,2].

Given the array nums **after** the possible rotation and an integer target, return *the index of* target *if it is in* nums*, or* -1 *if it is not in* nums.

You must write an algorithm with O(log n) runtime complexity.

Example 1:

Input: nums = [4,5,6,7,0,1,2], target = 0

Output: 4

Example 2:

Input: nums = [4,5,6,7,0,1,2], target = 3

Output: -1

Example 3:

Input: nums = [1], target = 0

Output: -1

Constraints:

- 1 <= nums.length <= 5000
- -10^4 <= nums[i] <= 10^4
- All values of nums are **unique**.
- nums is an ascending array that is possibly rotated.
- -10^4 <= target <= 10^4

4.9 81. Search in Rotated Sorted Array II

There is an integer array nums sorted in non-decreasing order (not necessarily with **distinct** values).

Before being passed to your function, nums is **rotated** at an unknown pivot index k (0 <= k < nums.length) such that the resulting array is [nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]] (**0-indexed**). For example, [0,1,2,4,4,4,5,6,6,7] might be rotated at pivot index 5 and become [4,5,6,6,7,0,1,2,4,4].

Given the array nums **after** the rotation and an integer target, return true *if* target *is in* nums*, or* false *if it is not in* nums.

You must decrease the overall operation steps as much as possible.

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

Constraints:

- 1 <= nums.length <= 5000
- -10^4 <= nums[i] <= 10^4
- nums is guaranteed to be rotated at some pivot.
- -10^4 <= target <= 10^4

4.10 153. Find Minimum in Rotated Sorted Array

Medium

Topics

Companies

Hint

Suppose an array of length n sorted in ascending order is **rotated** between 1 and n times. For example, the array nums = [0,1,2,4,5,6,7] might become:

- [4,5,6,7,0,1,2] if it was rotated 4 times.
- [0,1,2,4,5,6,7] if it was rotated 7 times.

Notice that **rotating** an array [a[0], a[1], a[2], ..., a[n-1]] 1 time results in the array [a[n-1], a[0], a[1], a[2], ..., a[n-2]].

Given the sorted rotated array nums of **unique** elements, return *the minimum element of this array*.

You must write an algorithm that runs in O(log n) time.

Example 1:

Input: nums = [3,4,5,1,2]

Output: 1

Explanation: The original array was [1,2,3,4,5] rotated 3 times.

Example 2:

Input: nums = [4,5,6,7,0,1,2]

Output: 0

Explanation: The original array was [0,1,2,4,5,6,7] and it was rotated 4 times.

Example 3:

Input: nums = [11,13,15,17]

Output: 11

Explanation: The original array was [11,13,15,17] and it was rotated 4 times.

Constraints:

- n == nums.length

- 1 <= n <= 5000
- -5000 <= nums[i] <= 5000
- All the integers of nums are **unique**.
- nums is sorted and rotated between 1 and n times.

4.11 **Find Kth Rotation**

Difficulty: **Easy**Accuracy: **23.16%**Submissions: **287K+**Points: **2**Average Time: **20m**

Given an increasing sorted rotated array **arr** of distinct integers. The array is right-rotated **k** times. Find the value of **k**.
Let's suppose we have an array arr = [2, 4, 6, 9], so if we rotate it by 2 times so that it will look like this:
After 1st Rotation : [9, 2, 4, 6]
After 2nd Rotation : [6, 9, 2, 4]

Examples:

Input: arr = [5, 1, 2, 3, 4]

Output: 1

Explanation: The given array is 5 1 2 3 4. The original sorted array is 1 2 3 4 5. We can see that the array was rotated 1 times to the right.

Input: arr = [1, 2, 3, 4, 5]

Output: 0

Explanation: The given array is not rotated.

Expected Time Complexity: O(log(n))
Expected Auxiliary Space: O(1)

Constraints:
1 <= n <=10^5
1 <= arr_i <= 10^7

4.12 [540. Single Element in a Sorted Array]

Medium

Topics

Companies

You are given a sorted array consisting of only integers where every element appears exactly twice, except for one element which appears exactly once.

Return *the single element that appears only once*.

Your solution must run in O(log n) time and O(1) space.

Example 1:

Input: nums = [1,1,2,3,3,4,4,8,8]

Output: 2

Example 2:

Input: nums = [3,3,7,7,10,11,11]

Output: 10

Constraints:

- 1 <= nums.length <= 10^5
- 0 <= nums[i] <= 10^5

4.13 162. Find Peak Element

Medium

Topics

Companies

A peak element is an element that is strictly greater than its neighbors.

Given a **0-indexed** integer array nums, find a peak element, and return its index. If the array contains multiple peaks, return the index to **any of the peaks**.

You may imagine that nums[-1] = nums[n] = -∞. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.

You must write an algorithm that runs in O(log n) time.

Example 1:

Input: nums = [1,2,3,1]

Output: 2

Explanation: 3 is a peak element and your function should return the index number 2.

Example 2:

Input: nums = [1,2,1,3,5,6,4]

Output: 5

Explanation: Your function can return either index number 1 where the peak element is 2, or index number 5 where the peak element is 6.

Constraints:

- 1 <= nums.length <= 1000
- -2^{31} <= nums[i] <= $2^{31} - 1$
- nums[i] != nums[i + 1] for all valid i.

Lecture 2 :

2.1 **69. Sqrt(x)** [Leetcode]

Given a non-negative integer x, return *the square root of* x *rounded down to the nearest integer*. The returned integer should be **non-negative** as well.

You **must not use** any built-in exponent function or operator.

- For example, do not use pow(x, 0.5) in c++ or x ** 0.5 in python.

Example 1:

Input: x = 4

Output: 2

Explanation: The square root of 4 is 2, so we return 2.

Example 2:

Input: x = 8

Output: 2

Explanation: The square root of 8 is 2.82842..., and since we round it down to the nearest integer, 2 is returned.

Constraints:

- $0 <= x <= 2^{31} - 1$

2.2 **Find nth root of m**

Difficulty: **Easy**Accuracy: **25.06%**Submissions: **207K+**Points: **2**Average Time: **15m**

You are given 2 numbers **n and m,** the task is to find $\sqrt[n]{m}$ (n^{th} root of m). If the root is not integer then returns -1.

Examples :

Input: n = 2, m = 9

Output: 3

Explanation: $3^2 = 9$

Input: n = 3, m = 9

Output: -1

Explanation: 3rd root of 9 is not integer.

Input: n = 1, m = 14

Output: 14

Constraints:
1 <= n <= 30
1 <= m <= 10^9

2.3 [875. Koko Eating Bananas]

Medium

Topics

Companies

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 will not eat any more bananas during this hour.

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

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

Constraints:

- 1 <= piles.length <= 10^4
- piles.length <= h <= 10^9
- 1 <= piles[i] <= 10^9

2.4 [1482. Minimum Number of Days to Make m Bouquets]

You are given an integer array bloomDay, an integer m and an integer k.

You want to make m bouquets. To make a bouquet, you need to use k **adjacent flowers** from the garden.

The garden consists of n flowers, the i^{th} flower will bloom in the bloomDay[i] and then can be used in **exactly one** bouquet.

Return *the minimum number of days you need to wait to be able to make* m *bouquets from the garden*. If it is impossible to make m bouquets return -1.

Example 1:

Input: bloomDay = [1,10,3,10,2], m = 3, k = 1

Output: 3

Explanation: Let us see what happened in the first three days. x means flower bloomed and _ means flower did not bloom in the garden.

We need 3 bouquets each should contain 1 flower.

After day 1: [x, _, _, _, _] // we can only make one bouquet.

After day 2: [x, _, _, _, x] // we can only make two bouquets.

After day 3: [x, _, x, _, x] // we can make 3 bouquets. The answer is 3.

Example 2:

Input: bloomDay = [1,10,3,10,2], m = 3, k = 2

Output: -1

Explanation: We need 3 bouquets each has 2 flowers, that means we need 6 flowers. We only have 5 flowers so it is impossible to get the needed bouquets and we return -1.

Example 3:

Input: bloomDay = [7,7,7,7,12,7,7], m = 2, k = 3

Output: 12

Explanation: We need 2 bouquets each should have 3 flowers.

Here is the garden after the 7 and 12 days:

After day 7: [x, x, x, x, _, x, x]

We can make one bouquet of the first three flowers that bloomed. We cannot make another bouquet from the last three flowers that bloomed because they are not adjacent.

After day 12: [x, x, x, x, x, x, x]

It is obvious that we can make two bouquets in different ways.

Constraints:

- bloomDay.length == n
- 1 <= n <= 10^5
- 1 <= bloomDay[i] <= 10^9
- 1 <= m <= 10^6
- 1 <= k <= n

2.5 [1283. Find the Smallest Divisor Given a Threshold]

Medium

Topics

Companies

Hint

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

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

The test cases are generated so that there will be an answer.

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 of 7 (1+1+2+3) and if the divisor is 5 the sum will be 5 (1+1+1+2).

Example 2:

Input: nums = [44,22,33,11,1], threshold = 5

Output: 44

Constraints:

- 1 <= nums.length <= 5 * 10^4
- 1 <= nums[i] <= 10^6
- nums.length <= threshold <= 10^6

2.6 **1011. Capacity To Ship Packages Within D Days**

Medium

Topics

Companies

Hint

A conveyor belt has packages that must be shipped from one port to another within days 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 days days.

Example 1:

Input: weights = [1,2,3,4,5,6,7,8,9,10], days = 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], days = 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], days = 4

Output: 3

Explanation:

1st day: 1

2nd day: 2

3rd day: 3

4th day: 1, 1

Constraints:

- 1 <= days <= weights.length <= $5 * 10^4$
- 1 <= weights[i] <= 500

2.7 1539. Kth Missing Positive Number

Given an array arr of positive integers sorted in a **strictly increasing order**, and an integer k.

Return *the* k^{th} ***positive*** *integer that is* ***missing*** *from this array.*

Example 1:

Input: arr = [2,3,4,7,11], k = 5

Output: 9

Explanation: The missing positive integers are [1,5,6,8,9,10,12,13,...]. The 5^{th} missing positive integer is 9.

Example 2:

Input: arr = [1,2,3,4], k = 2

Output: 6

Explanation: The missing positive integers are [5,6,7,...]. The 2^{nd} missing positive integer is 6.

Constraints:

- 1 <= arr.length <= 1000
- 1 <= arr[i] <= 1000
- 1 <= k <= 1000
- arr[i] < arr[j] for 1 <= i < j <= arr.length

2.8 **Aggressive Cows**

Difficulty: **Medium**Accuracy: **59.57%**Submissions: **133K+**Points: **4**Average Time: **30m**

You are given an array with unique elements of stalls[], which denote the position of a **stall**. You are also given an integer **k** which denotes the number of aggressive cows. Your task is to assign **stalls** to **k** cows such that the **minimum distance** between any two of them is the **maximum** possible.

Examples :

Input: stalls[] = [1, 2, 4, 8, 9], k = 3

Output: 3

Explanation: The first cow can be placed at stalls[0],
the second cow can be placed at stalls[2] and

the third cow can be placed at stalls[3].

The minimum distance between cows, in this case, is 3, which also is the largest among all possible ways.

Input: stalls[] = [10, 1, 2, 7, 5], k = 3

Output: 4

Explanation: The first cow can be placed at stalls[0],

the second cow can be placed at stalls[1] and

the third cow can be placed at stalls[4].

The minimum distance between cows, in this case, is 4, which also is the largest among all possible ways.

Input: stalls[] = [2, 12, 11, 3, 26, 7], k = 5

Output: 1

Explanation: Each cow can be placed in any of the stalls, as the no. of stalls are exactly equal to the number of cows.

The minimum distance between cows, in this case, is 1, which also is the largest among all possible ways.

Constraints:
2 <= stalls.size() <= 10^6
0 <= stalls[i] <= 10^8
2 <= k <= stalls.size()

2.9 **Allocate Minimum Pages**

Difficulty: **Medium**Accuracy: **35.51%**Submissions: **275K+**Points: **4**Average Time: **35m**

You are given an array arr[] of integers, where each element arr[i] represents the number of pages in the ith book. You also have an integer k representing the number of students. The task is to allocate books to each student such that:

- Each student receives **atleast** one book.
- Each student is assigned a contiguous sequence of books.
- No book is assigned to more than one student.

The objective is to minimize the maximum number of pages assigned to any student. In other words, out of all possible allocations, find the arrangement where the student who receives the most pages still has the smallest possible maximum.

Note: Return **-1** if a valid assignment is not possible, and allotment should be in contiguous order (see the explanation for better understanding).

Examples:

Input: arr[] = [12, 34, 67, 90], k = 2

Output: 113

Explanation: Allocation can be done in following ways:

[12] and [34, 67, 90] Maximum Pages = 191

[12, 34] and [67, 90] Maximum Pages = 157

[12, 34, 67] and [90] Maximum Pages = 113.

Therefore, the minimum of these cases is 113, which is selected as the output.

Input: arr[] = [15, 17, 20], k = 5

Output: -1

Explanation: Allocation can not be done.

Input: arr[] = [22, 23, 67], k = 1

Output: 112

Constraints:
1 <= arr.size() <= 10^6
1 <= arr[i] <= 10^3
1 <= k <= 10^3

2 .10 410. Split Array Largest Sum

Given an integer array nums and an integer k, split nums into k non-empty subarrays such that the largest sum of any subarray is **minimized**.

Return *the minimized largest sum of the split*.

A **subarray** is a contiguous part of the array.

Example 1:

Input: nums = [7,2,5,10,8], k = 2

Output: 18

Explanation: There are four ways to split nums into two subarrays.

The best way is to split it into [7,2,5] and [10,8], where the largest sum among the two subarrays is only 18.

Example 2:

Input: nums = [1,2,3,4,5], k = 2

Output: 9

Explanation: There are four ways to split nums into two subarrays.

The best way is to split it into [1,2,3] and [4,5], where the largest sum among the two subarrays is only 9.

Constraints:

- 1 <= nums.length <= 1000
- 0 <= nums[i] <= 10^6
- 1 <= k <= min(50, nums.length)

2.11 **The Painter's Partition Problem-II**

Difficulty: **Hard**Accuracy: **27.52%**Submissions: **137K+**Points: **8**

Dilpreet wants to paint his dog's home that has **n** boards with different lengths. The length of i^{th} board is given by **arr[i]** where **arr[]** is an array of **n** integers. He hired **k** painters for this work and each painter takes **1 unit time to paint 1 unit of the board.**

Return the minimum time to get this job done if all painters start together with the constraint that any painter will only paint continuous boards, say boards numbered **[2,3,4]** or only board **[1]** or nothing but not boards **[2,4,5]**.

Examples:

Input: arr[] = [5, 10, 30, 20, 15], k = 3

Output: 35

Explanation: The most optimal way will be: Painter 1 allocation : [5,10], Painter 2 allocation : [30], Painter 3 allocation : [20,15], Job will be done when all painters finish i.e. at time = max(5+10, 30, 20+15) = 35

Input: arr[] = [10, 20, 30, 40], k = 2

Output: 60

Explanation: The most optimal way to paint: Painter 1 allocation : [10,20,30], Painter 2 allocation : [40], Job will be complete at time = 60

Input: arr[] = [100, 200, 300, 400], k = 1

Output: 1000

Explanation: There is only one painter, so the painter must paint all boards sequentially. The total time taken will be the sum of all board lengths, i.e., 100 + 200 + 300 + 400 = 1000.

Constraints:
$1 \le arr.size() \le 10^5$
$1 \le arr[i] \le 10^5$
$1 \le k \le 10^5$

2.12 **Minimize Max Distance to Gas Station**

Difficulty: **Hard**Accuracy: **38.36%**Submissions: **79K+**Points: **8**Average Time: **40m**

We have a horizontal number line. On that number line, we have gas **stations** at positions stations[0], stations[1], ..., stations[n-1], where **n** is the size of the stations array. Now, we add **k** more gas stations so that **d**, the maximum distance between adjacent gas stations, is minimized. We have to find the smallest possible value of d. Find the answer **exactly** to 2 decimal places.
Note: **stations** is in a **strictly increasing** order.

Example 1:

Input:

n = 10

stations[] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

k = 9

Output: 0.50

Explanation: Each of the 9 stations can be added mid way between all the existing adjacent stations.

Example 2:

Input:

n = 10

stations[] = [3, 6, 12, 19, 33, 44, 67, 72, 89, 95]
k = 2
Output: 14.00
Explanation: Construction of gas stations at 8th(between 72 and 89) and 6th(between 44 and 67) locations.

Your Task:
You don't need to read input or print anything. Your task is to complete the function **findSmallestMaxDist()** which takes a list of stations and integer k as inputs and returns the smallest possible value of d. Find the answer **exactly** to 2 decimal places.

Constraint:
10 <= n <= 10000
0 <= stations[i] <= 10^9
0 <= k <= 10^5

2.13 4. Median of Two Sorted Arrays

Solved

Hard

Topics

Companies

Given two sorted arrays nums1 and nums2 of size m and n respectively, return **the median** of the two sorted arrays.

The overall run time complexity should be O(log (m+n)).

Example 1:

Input: nums1 = [1,3], nums2 = [2]

Output: 2.00000

Explanation: merged array = [1,2,3] and median is 2.

Example 2:

Input: nums1 = [1,2], nums2 = [3,4]

Output: 2.50000

Explanation: merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5.

Constraints:

- nums1.length == m
- nums2.length == n
- 0 <= m <= 1000
- 0 <= n <= 1000
- 1 <= m + n <= 2000
- -10^6 <= nums1[i], nums2[i] <= 10^6

2.14 **K-th element of two Arrays**

Difficulty: **Medium**Accuracy: **37.4%**Submissions: **343K+**Points: **4**Average Time: **15m**

Given two sorted arrays **a[]** and **b[]** and an element **k**, the task is to find the element that would be at the **kth** position of the combined sorted array.

Examples :

Input: a[] = [2, 3, 6, 7, 9], b[] = [1, 4, 8, 10], k = 5

Output: 6

Explanation: The final combined sorted array would be [1, 2, 3, 4, 6, 7, 8, 9, 10]. The 5th element of this array is 6.

Input: a[] = [100, 112, 256, 349, 770], b[] = [72, 86, 113, 119, 265, 445, 892], k = 7

Output: 256

Explanation: Combined sorted array is [72, 86, 100, 112, 113, 119, 256, 265, 349, 445, 770, 892]. The 7th element of this array is 256.

Constraints:

- 1 <= a.size(), b.size() <= 10^6
- 1 <= k <= a.size() + b.size()
- 0 <= a[i], b[i] < 10^8

Lecture 3 :

3.1 **Row with max 1s**

Difficulty: **Medium**Accuracy: **33.09%**Submissions: **347K+**Points: **4**

You are given a 2D binary array **arr[][]** consisting of only 1s and 0s. Each row of the array is sorted in non-decreasing order. Your task is to find and return the index of the first row that contains the maximum number of 1s. If no such row exists, return -1.

Note:

- The array follows 0-based indexing.
- The number of rows and columns in the array are denoted by n and m respectively.

Examples:

Input: arr[][] = [[0,1,1,1], [0,0,1,1], [1,1,1,1], [0,0,0,0]]

Output: 2

Explanation: Row 2 contains the most number of 1s (4 1s). Hence, the output is 2.

Input: arr[][] = [[0,0], [1,1]]

Output: 1

Explanation: Row 1 contains the most number of 1s (2 1s). Hence, the output is 1.

Input: arr[][] = [[0,0], [0,0]]

Output: -1

Explanation: No row contains any 1s, so the output is -1.

Constraints:

1 ≤ arr.size(), arr[i].size() ≤ 10^3
0 ≤ arr[i][j] ≤ 1

3.2 [74. Search a 2D Matrix]

Medium

Topics

Companies

You are given an m x n integer matrix matrix with the following two properties:

- Each row is sorted in non-decreasing order.
- The first integer of each row is greater than the last integer of the previous row.

Given an integer target, return true *if* target *is in* matrix *or* false *otherwise*.

You must write a solution in O(log(m * n)) time complexity.

Example 1:

1	3	5	7
10	11	16	20
23	30	34	60

Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3

Output: true

Example 2:

1	3	5	7
10	11	16	20
23	30	34	60

Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 13

Output: false

Constraints:

- m == matrix.length
- n == matrix[i].length
- 1 <= m, n <= 100
- -10^4 <= matrix[i][j], target <= 10^4

3.3 240. Search a 2D Matrix II

Medium

Topics

Companies

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

- Integers in each row are sorted in ascending from left to right.
- Integers in each column are sorted in ascending from top to bottom.

Example 1:

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

Input: matrix = [[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]], target = 5

Output: true

Example 2:

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

Input: matrix = [[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]], target = 20

Output: false

Constraints:

- m == matrix.length
- n == matrix[i].length
- 1 <= n, m <= 300
- -10^9 <= matrix[i][j] <= 10^9
- All the integers in each row are **sorted** in ascending order.
- All the integers in each column are **sorted** in ascending order.
- -10^9 <= target <= 10^9

3.4 [1901. Find a Peak Element II]

Medium

Topics

Companies

Hint

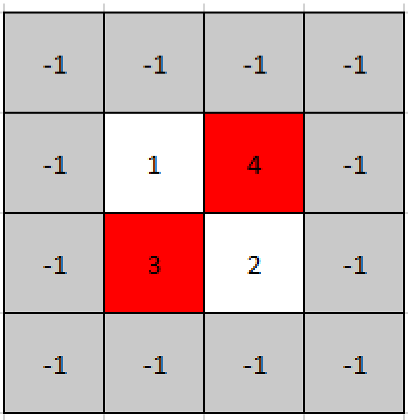
A **peak** element in a 2D grid is an element that is **strictly greater** than all of its **adjacent** neighbors to the left, right, top, and bottom.

Given a **0-indexed** m x n matrix mat where **no two adjacent cells are equal**, find **any** peak element mat[i][j] and return *the length 2 array* [i,j].

You may assume that the entire matrix is surrounded by an **outer perimeter** with the value -1 in each cell.

You must write an algorithm that runs in O(m log(n)) or O(n log(m)) time.

Example 1:



Input: mat = [[1,4],[3,2]]

Output: [0,1]

Explanation: Both 3 and 4 are peak elements so [1,0] and [0,1] are both acceptable answers.

Example 2:

-1	-1	-1	-1	-1
-1	10	20	15	-1
-1	21	30	14	-1
-1	7	16	32	-1
-1	-1	-1	-1	-1

Input: mat = [[10,20,15],[21,30,14],[7,16,32]]

Output: [1,1]

Explanation: Both 30 and 32 are peak elements so [1,1] and [2,2] are both acceptable answers.

Constraints:

- m == mat.length
- n == mat[i].length
- 1 <= m, n <= 500
- 1 <= mat[i][j] <= 10^5
- No two adjacent cells are equal.

3.5 **Median in a row-wise sorted Matrix**

Difficulty: **Hard**Accuracy: **55.05%**Submissions: **140K+**Points: **8**

Given a row-wise sorted matrix where the number of rows and columns is always **odd**, find the median of the matrix.

Examples:

Input: mat = [[1, 3, 5], [2, 6, 9], [3, 6, 9]]

Output: 5

Explanation: Sorting matrix elements gives us {1,2,3,3,5,6,6,9,9}. Hence, 5 is median.

Input: mat = [[1], [2], [3]]

Output: 2

Explanation: Sorting matrix elements gives us {1,2,3}. Hence, 2 is median

Input: mat = [[3], [5], [8]]

Output: 5

Explanation: Sorting matrix elements gives us {3,5,8}. Hence, 5 is median.

Constraints:
1 <= mat.size(), mat[0].size() <= 400
1 <= mat[i][j] <= 2000