JAVA PRACTICE SHEET (02/02/2026 – 07/02/2026)

1.

Given an array arr[] of positive integers, where each value represents the number of chocolates in a packet. Each packet can have a variable number of chocolates. There are m students, the task is to distribute chocolate packets among m students such that -

i. Each student gets exactly one packet.

ii. The difference between maximum number of chocolates given to a student and minimum number of chocolates given to a student is minimum and return that minimum possible difference.

Examples:

Input: arr = [3, 4, 1, 9, 56, 7, 9, 12], m = 5

Output: 6

Explanation: The minimum difference between maximum chocolates and minimum chocolates is 9 - 3 = 6 by choosing following m packets :[3, 4, 9, 7, 9].

Input: arr = [7, 3, 2, 4, 9, 12, 56], m = 3

Output: 2

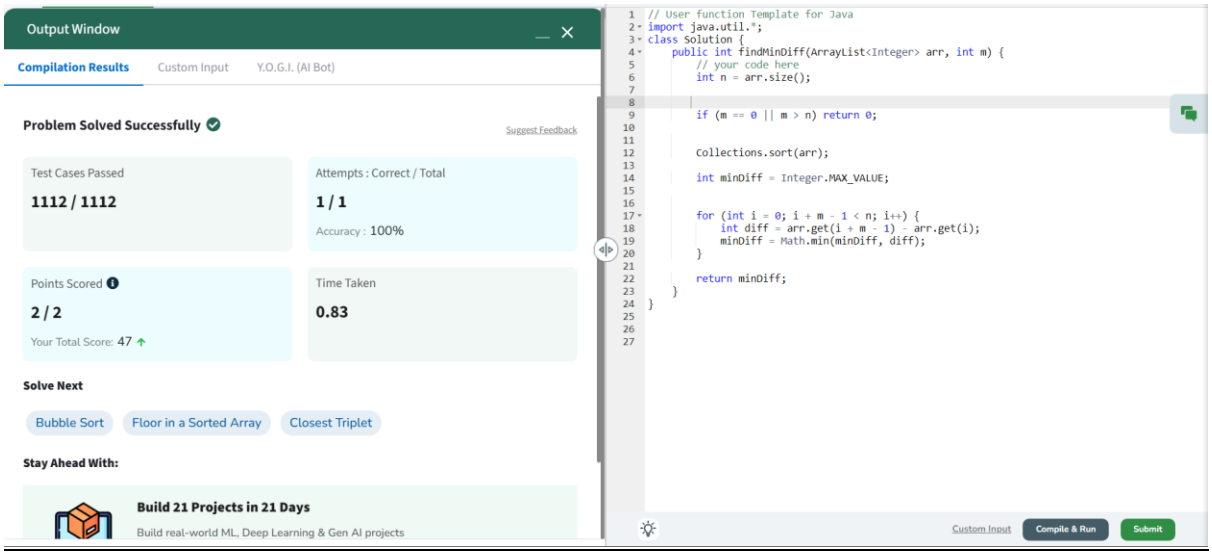
Explanation: The minimum difference between maximum chocolates and minimum chocolates is 4 - 2 = 2 by choosing following m packets :[3, 2, 4].

Input: arr = [3, 4, 1, 9, 56], m = 5

Output: 55

Explanation: With 5 packets for 5 students, each student will receive one packet, so the difference is 56 - 1 = 55.

Sol



2.

Given a number x and an array of integers arr, find the smallest subarray with sum greater than the given value. If such a subarray do not exist return 0 in that case.

Examples:

Input: x = 51, arr[] = [1, 4, 45, 6, 0, 19]

Output: 3

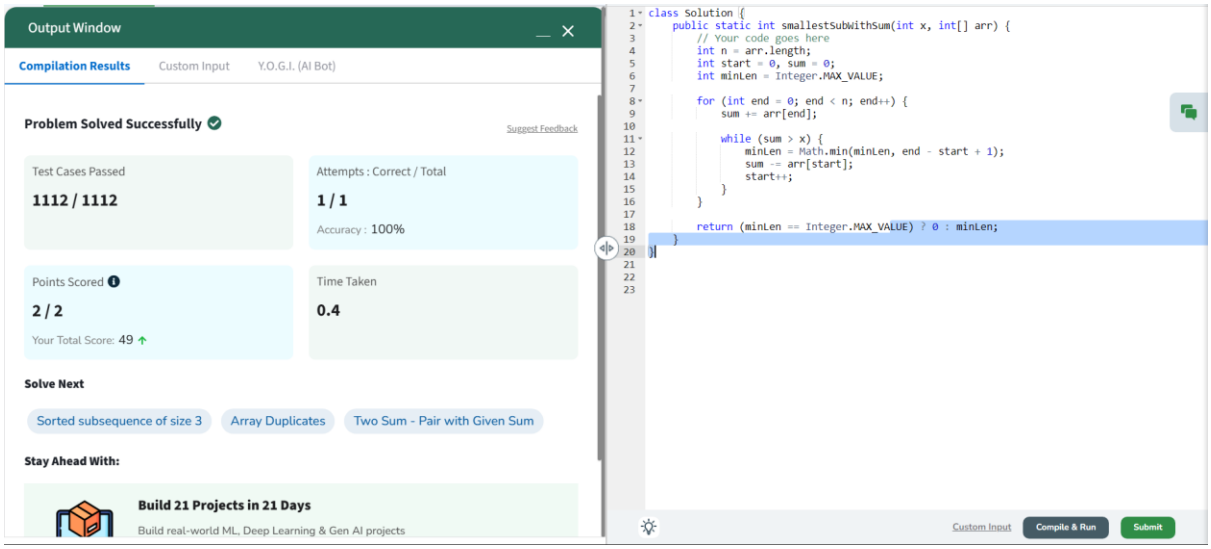
Explanation: Minimum length subarray is [4, 45, 6]

Input: x = 100, arr[] = [1, 10, 5, 2, 7]

Output: 0

Explanation: No subarray exist

Sol



3.

Given an array and a range a, b. The task is to partition the array around the range such that the array is divided into three parts.

1) All elements smaller than a come first.

2) All elements in range a to b come next.

3) All elements greater than b appear in the end.

The individual elements of three sets can appear in any order. You are required to return the modified array.

Note: The generated output is true if you modify the given array successfully.

Otherwise false.

Geeky Challenge: Solve this problem in O(n) time complexity.

Examples:

Input: arr[] = [1, 2, 3, 3, 4], a = 1, b = 2

Output: true

Explanation: One possible arrangement is: {1, 2, 3, 3, 4}. If you return a valid arrangement, output will be true.

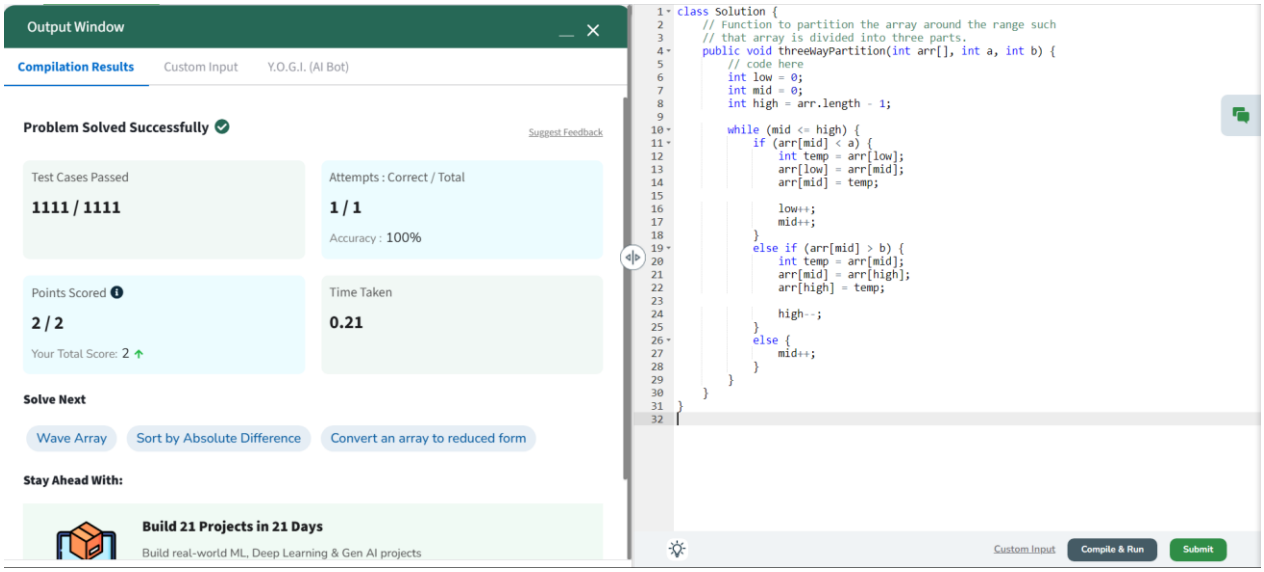
Input: arr[] = [1, 4, 3, 6, 2, 1], a = 1, b = 3

Output: true

Explanation: One possible arrangement is: {1, 3, 2, 1, 4, 6}. If you return a valid

arrangement, output will be true.

Sol



4.

Given an array arr and a number k. One can apply a swap operation on the array any number of times, i.e choose any two index i and j (i < j) and swap arr[i] , arr[j] . Find the minimum number of swaps required to bring all the numbers less than or equal to k together, i.e. make them a contiguous subarray.

Examples :

Input: arr[] = [2, 1, 5, 6, 3], k = 3

Output: 1

Explanation: To bring elements 2, 1, 3 together, swap index 2 with 4 (0-based indexing), i.e. element arr[2] = 5 with arr[4] = 3 such that final array will be- arr[] = [2, 1, 3, 6, 5]

Input: arr[] = [2, 7, 9, 5, 8, 7, 4], k = 6

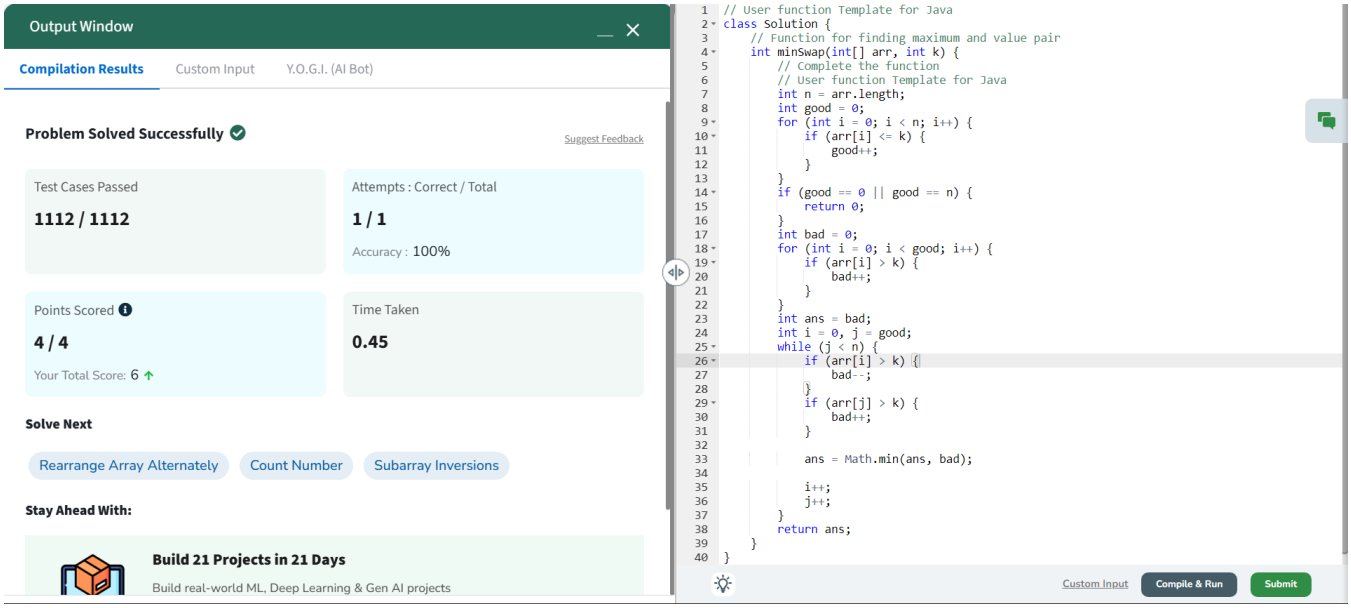
Output: 2

Explanation: To bring elements 2, 5, 4 together, swap index 0 with 2 (0-based indexing) and index 4 with 6 (0-based indexing) such that final array will be- arr[] = [9, 7, 2, 5, 4, 7, 8]

Input: arr[] = [2, 4, 5, 3, 6, 1, 8], k = 6

Output: 0

Sol



5.

Given an array arr[] of positive integers. Return true if all the array elements are palindrome otherwise, return false.

Examples:

Input: arr[] = [111, 222, 333, 444, 555]

Output: true

Explanation:

arr[0] = 111, which is a palindrome number.

arr[1] = 222, which is a palindrome number.

arr[2] = 333, which is a palindrome number.

arr[3] = 444, which is a palindrome number.

arr[4] = 555, which is a palindrome number.

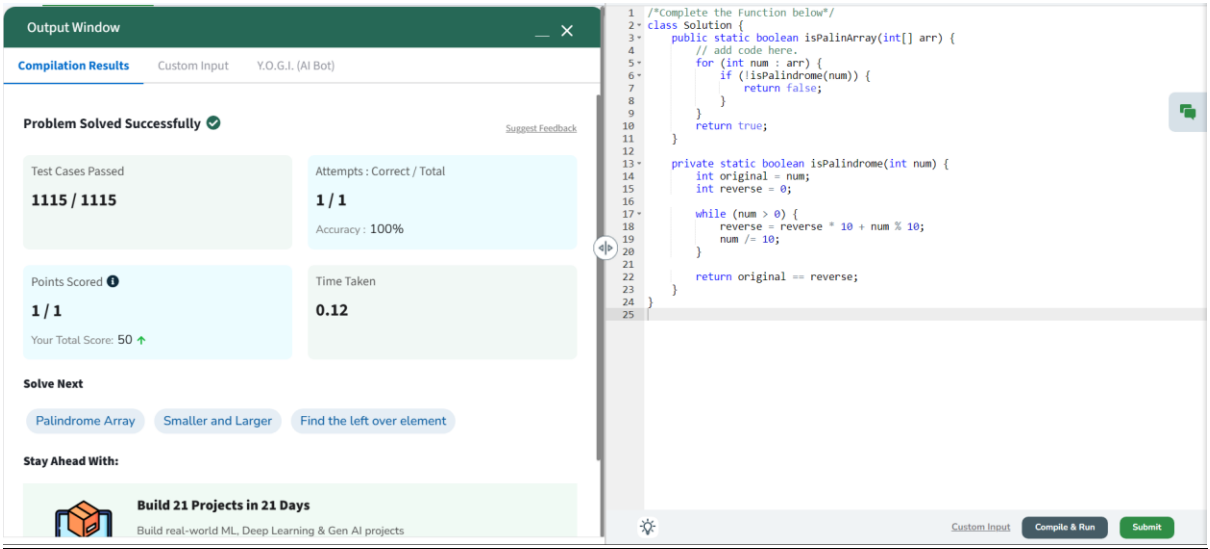
As all numbers are palindrome so This will return true.

Input: arr[] = [121, 131, 20]

Output: false

Explanation: 20 is not a palindrome hence the output is false.

Sol



6.

Given an array arr[] of integers, calculate the median.

Examples:

Input: arr[] = [90, 100, 78, 89, 67]

Output: 89

Explanation: After sorting the array middle element is the median

Input: arr[] = [56, 67, 30, 79]

Output: 61.5

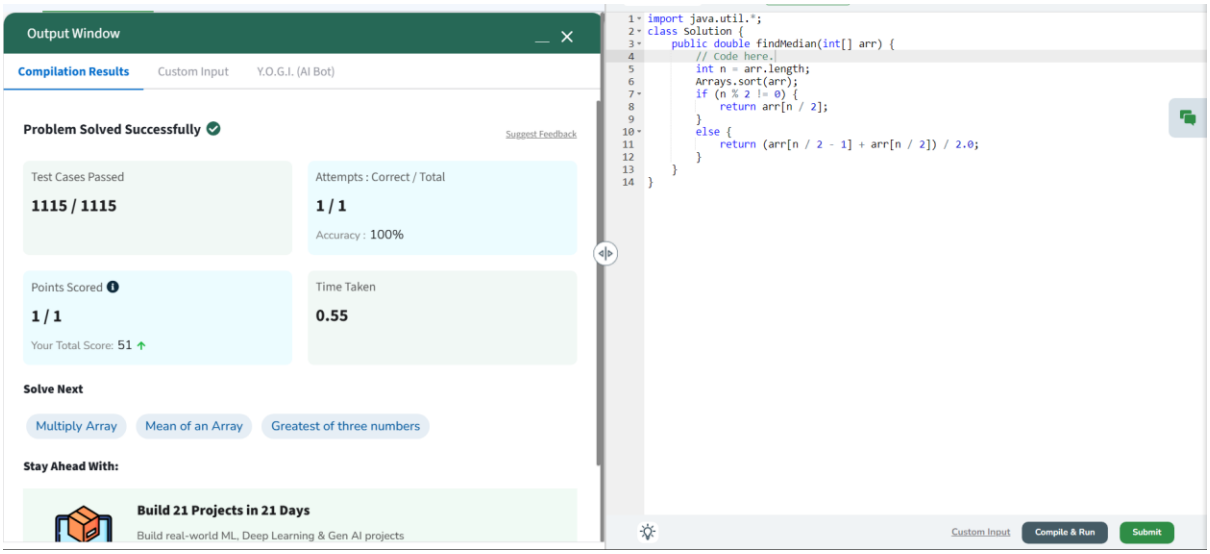
Explanation: In case of even number of elements, average of two middle elements is the median.

Input: arr[] = [1, 2]

Output: 1.5

Explanation: The average of both elements will result in 1.5.

Sol



7.

You are given a rectangular matrix mat[][] of size n x m, and your task is to return an array while traversing the matrix in spiral form.

Examples:

Input: mat[][] = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]]

Output: [1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5, 6, 7, 11, 10]

Explanation:

Input: mat[][] = [[1, 2, 3, 4, 5, 6], [7, 8, 9, 10, 11, 12], [13, 14, 15, 16, 17, 18]]

Output: [1, 2, 3, 4, 5, 6, 12, 18, 17, 16, 15, 14, 13, 7, 8, 9, 10, 11]

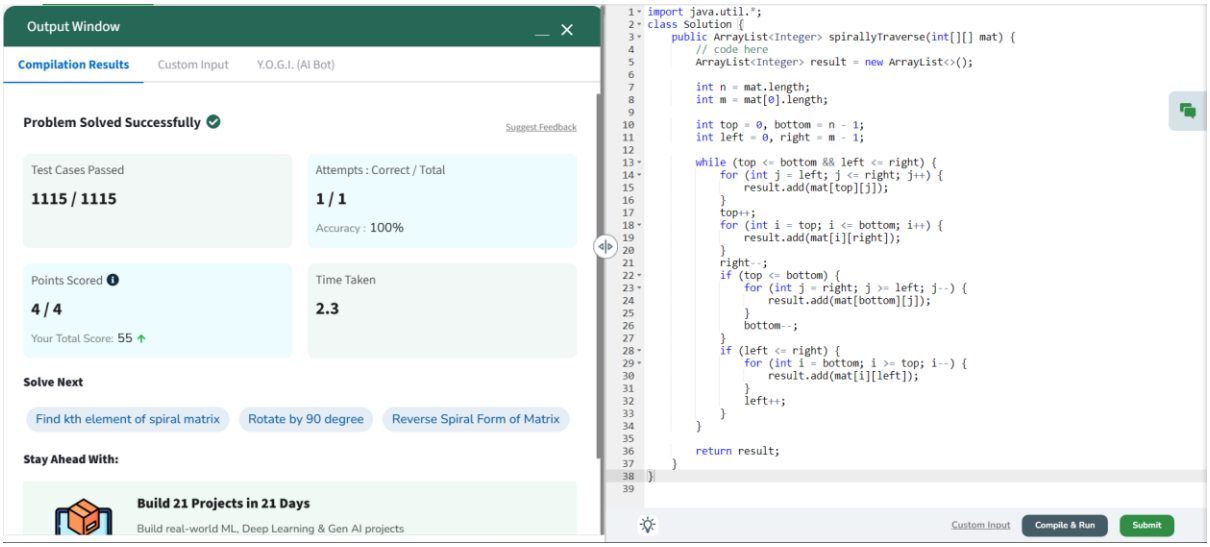
Explanation: Applying same technique as shown above.

Input: mat[][] = [[32, 44, 27, 23], [54, 28, 50, 62]]

Output: [32, 44, 27, 23, 62, 50, 28, 54]

Explanation: Applying same technique as shown above, output will be [32, 44, 27, 23, 62, 50, 28, 54].

Sol




```java
import java.util.*;
class Solution {
    public ArrayList<Integer> spirallyTraverse(int[][] mat) {
        // code here
        ArrayList<Integer> result = new ArrayList<>();

        int n = mat.length;
        int m = mat[0].length;

        int top = 0, bottom = n - 1;
        int left = 0, right = m - 1;

        while (top <= bottom && left <= right) {
            for (int j = left; j <= right; j++) {
                result.add(mat[top][j]);
            }
            top++;
            for (int i = top; i <= bottom; i++) {
                result.add(mat[i][right]);
            }
            right--;
            if (top <= bottom) {
                for (int j = right; j >= left; j--) {
                    result.add(mat[bottom][j]);
                }
                bottom--;
            }
            if (left <= right) {
                for (int i = bottom; i >= top; i--) {
                    result.add(mat[i][left]);
                }
                left++;
            }
        }

        return result;
    }
}
```

8.

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

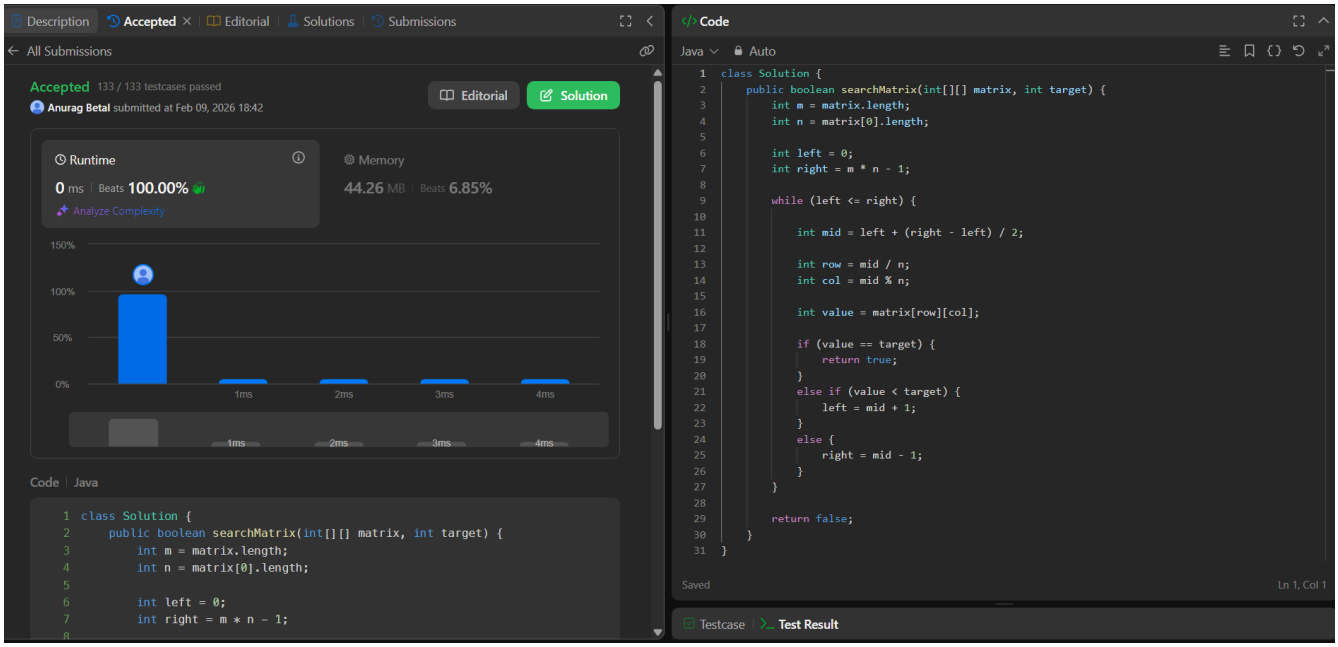
Example 2:

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

Sol



9.

Given a row-wise sorted matrix mat[][] of size n*m, where the number of rows and columns is always odd. Return 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[][] = [[2, 4, 9],

[3, 6, 7],

[4, 7, 10]]

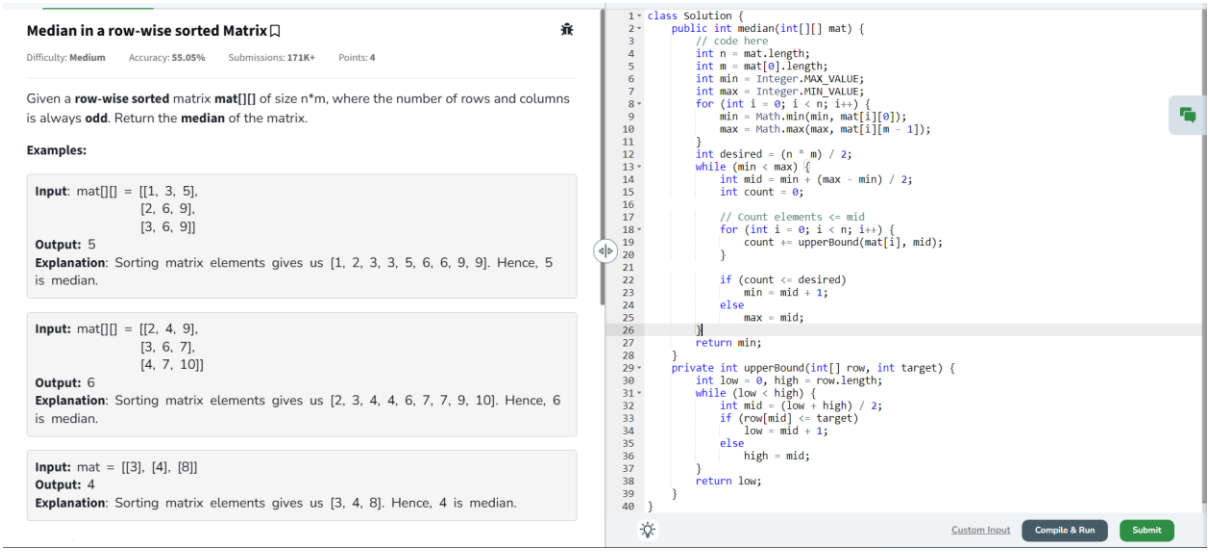
Output: 6

Explanation: Sorting matrix elements gives us [2, 3, 4, 4, 6, 7, 7, 9, 10]. Hence, 6 is median.

Input: mat = [[3], [4], [8]]

Output: 4

Explanation: Sorting matrix elements gives us [3, 4, 8]. Hence, 4 is median.



10.

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.

Sol