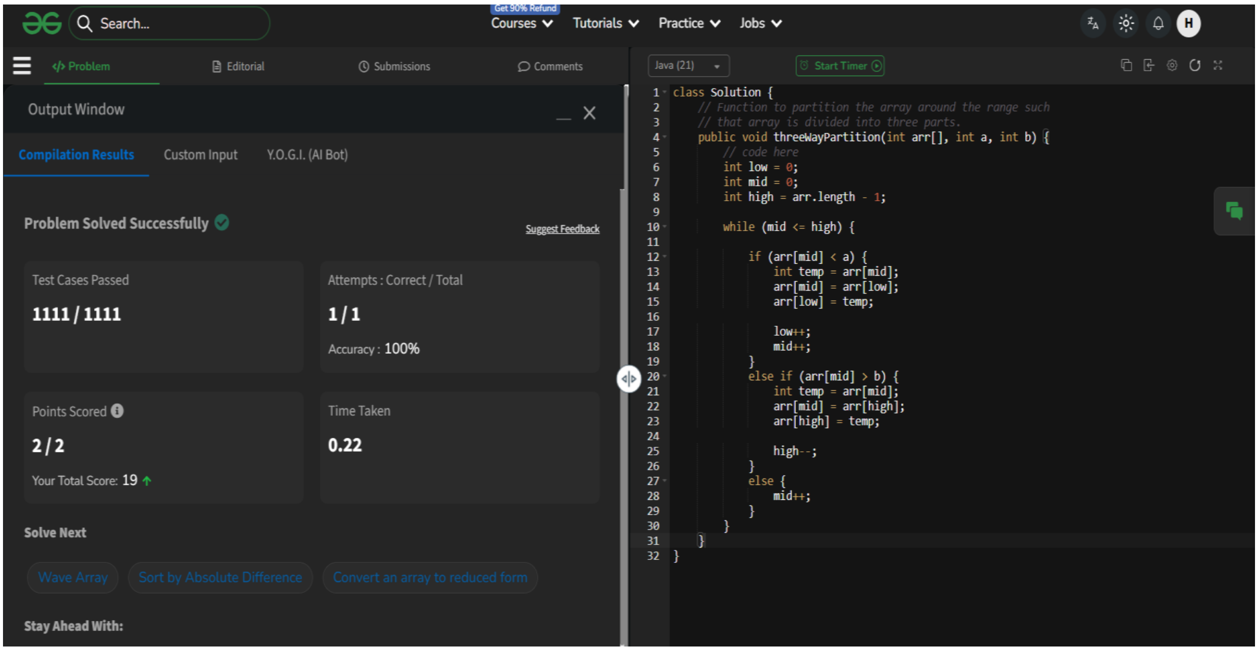
JAVA PRACTICE SHEET (02/02/2026 – 07/02/2026)

1. 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.

Solution-



Problem Solved Successfully

Test Cases Passed
1111 / 1111

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
2 / 2
Your Total Score: 19

Time Taken
0.22

```java
class Solution {
    // Function to partition the array around the range such
    // that array is divided into three parts.
    public void threeWayPartition(int arr[], int a, int b) {
        // code here
        int low = 0;
        int mid = 0;
        int high = arr.length - 1;

        while (mid <= high) {

            if (arr[mid] < a) {
                int temp = arr[mid];
                arr[mid] = arr[low];
                arr[low] = temp;

                low++;
                mid++;
            }
            else if (arr[mid] > b) {
                int temp = arr[mid];
                arr[mid] = arr[high];
                arr[high] = temp;

                high--;
            }
            else {
                mid++;
            }
        }
    }
}
```

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 :

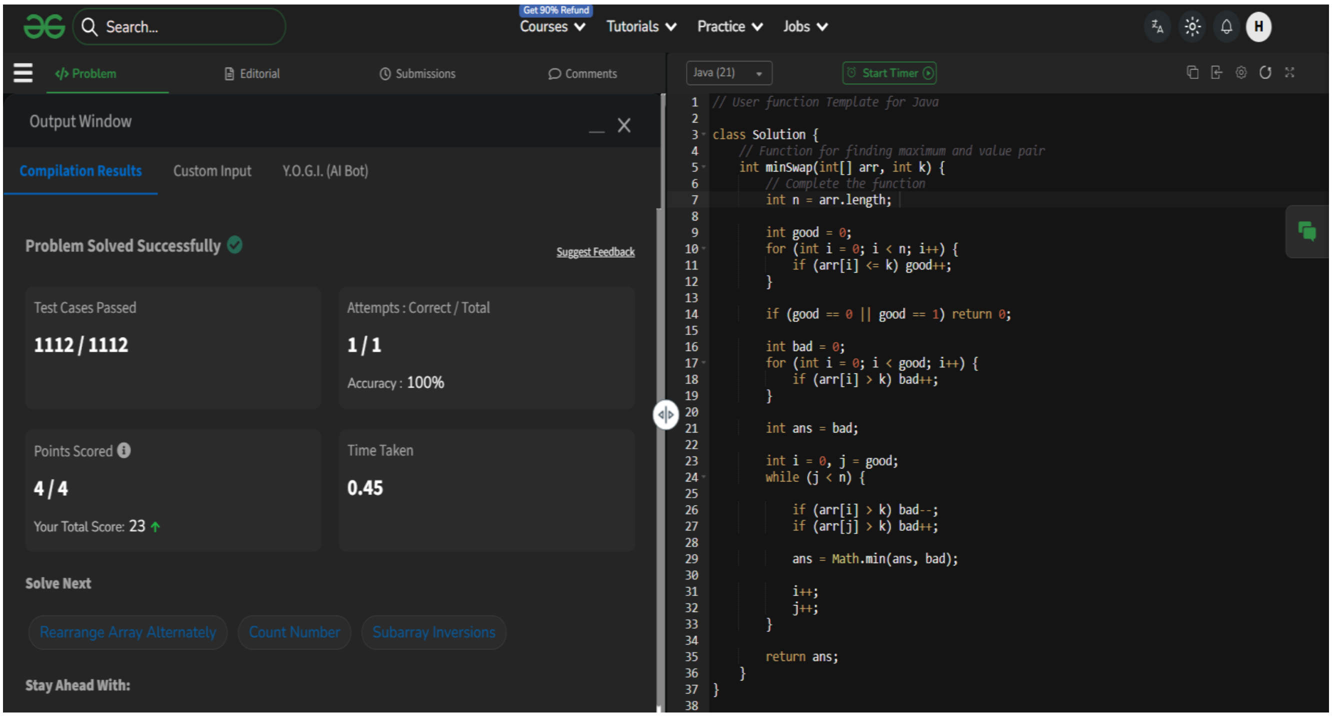
Input: arr[] = [2, 1, 5, 6, 3], k = 3
Output: 1
Explanation: To bring elements 2, 1, 3 together, swap index 2 with 4 (0-based indexing), i.e. element arr[2] = 5 with arr[4] = 3 such that final array will be- arr[] = [2, 1, 3, 6, 5]

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

2.

Solution-



Output Window

Compilation Results   Custom Input   Y.O.G.I. (AI Bot)

Problem Solved Successfully ✓   Suggest Feedback

Test Cases Passed
1112 / 1112

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
4 / 4
Your Total Score: 23

Time Taken
0.45

Solve Next

Rearrange Array Alternately   Count Number   Subarray Inversions

Stay Ahead With:

```java
// User function Template for Java

class Solution {
    // Function for finding maximum and value pair
    int minSwap(int[] arr, int k) {
        // Complete the function
        int n = arr.length;

        int good = 0;
        for (int i = 0; i < n; i++) {
            if (arr[i] <= k) good++;
        }

        if (good == 0 || good == 1) return 0;

        int bad = 0;
        for (int i = 0; i < good; i++) {
            if (arr[i] > k) bad++;
        }

        int ans = bad;

        int i = 0, j = good;
        while (j < n) {

            if (arr[i] > k) bad--;
            if (arr[j] > k) bad++;

            ans = Math.min(ans, bad);

            i++;
            j++;
        }

        return ans;
    }
}
```

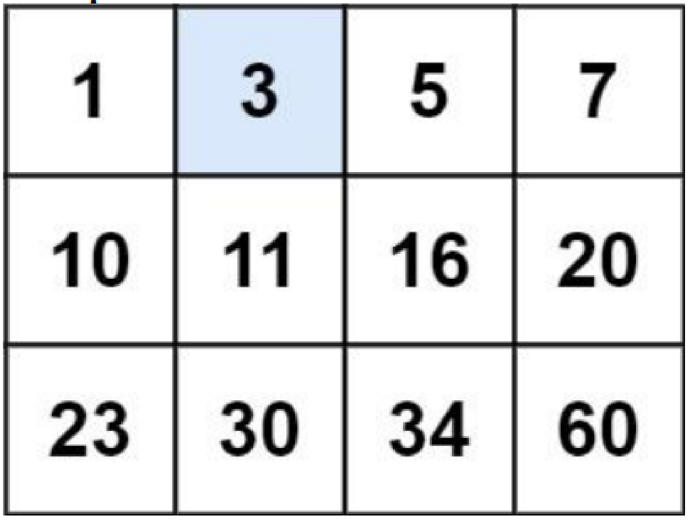
3. You are given an m x n integer matrix matrix with the following two properties:

- 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:

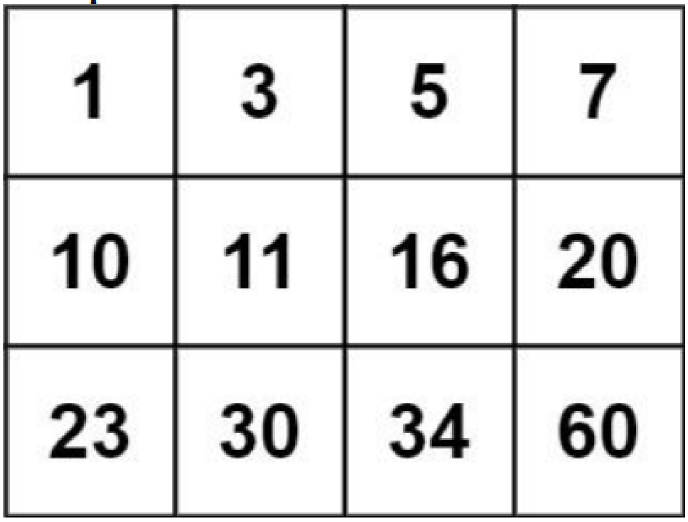


1	3	5	7
10	11	16	20
23	30	34	60

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

Output: true

Example 2:

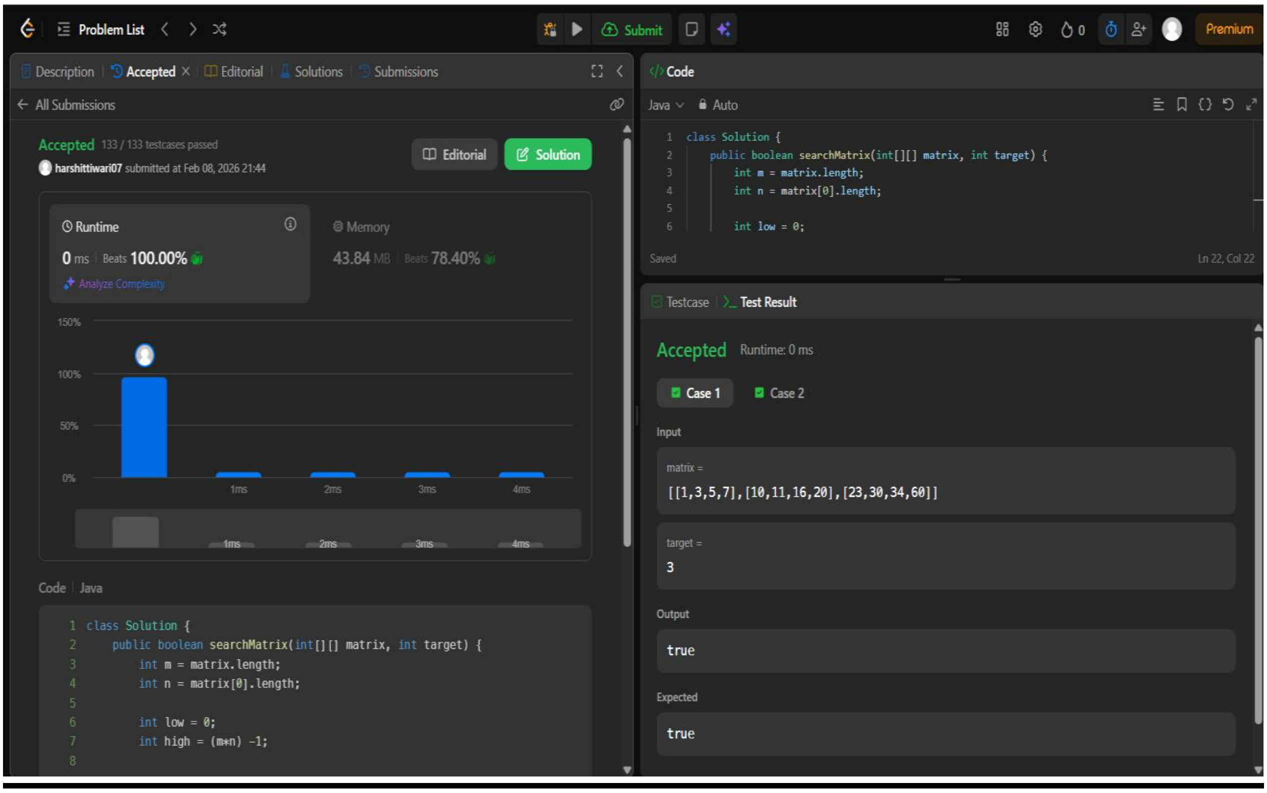


1	3	5	7
10	11	16	20
23	30	34	60

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

Output: false

Solution-



Accepted 133 / 133 testcases passed

harshittiwari07 submitted at Feb 08, 2026 21:44

Runtime
0 ms | Beats 100.00%

Memory
43.84 MB | Beats 78.40%

Code | Java

```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int m = matrix.length;
        int n = matrix[0].length;

        int low = 0;
        int high = (m*n) -1;

```

```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int m = matrix.length;
        int n = matrix[0].length;

        int low = 0;
```

Accepted Runtime: 0 ms

Input

matrix =

```
[[1,3,5,7],[10,11,16,20],[23,30,34,60]]
```

target =

```
3
```

Output

```
true
```

Expected

```
true
```

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.

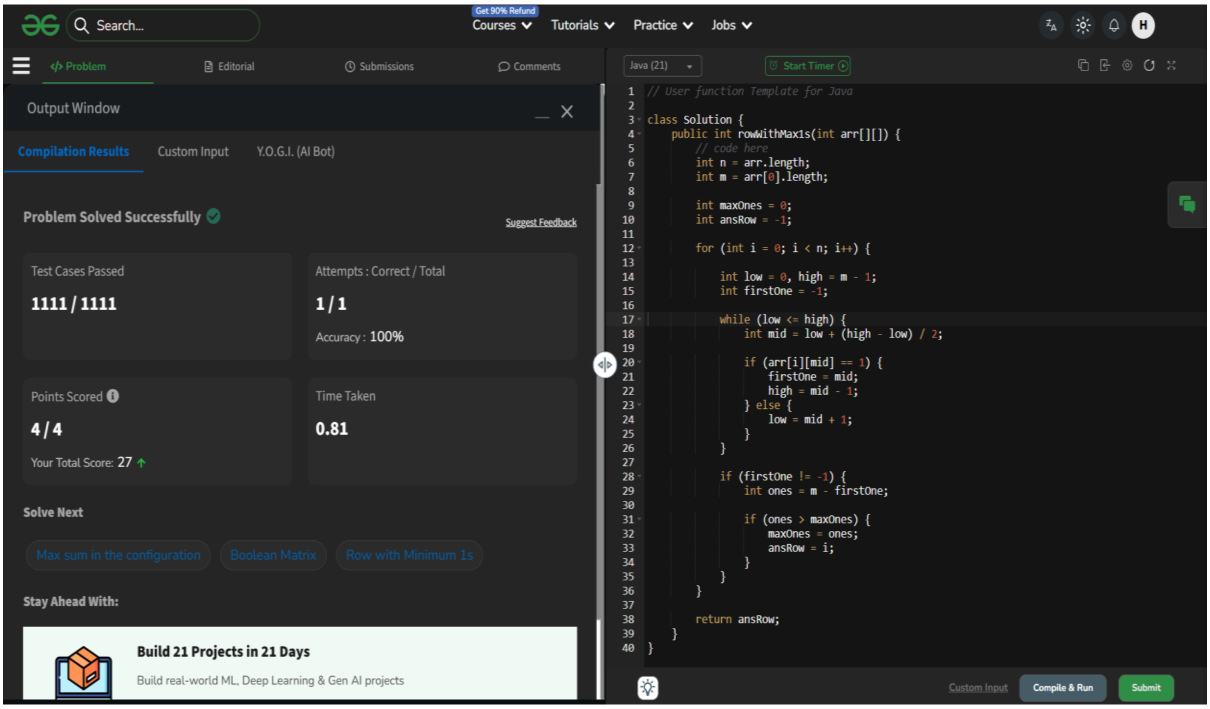
Note:

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

Solution-



Problem Solved Successfully

Test Cases Passed
1111 / 1111

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
4 / 4
Your Total Score: 27

Time Taken
0.81

```java
// User function Template for Java

class Solution {
    public int rowWithMax1s(int arr[][]) {
        // code here
        int n = arr.length;
        int m = arr[0].length;

        int maxOnes = 0;
        int ansRow = -1;

        for (int i = 0; i < n; i++) {

            int low = 0, high = m - 1;
            int firstOne = -1;

            while (low <= high) {
                int mid = low + (high - low) / 2;

                if (arr[i][mid] == 1) {
                    firstOne = mid;
                    high = mid - 1;
                } else {
                    low = mid + 1;
                }
            }

            if (firstOne != -1) {
                int ones = m - firstOne;

                if (ones > maxOnes) {
                    maxOnes = ones;
                    ansRow = i;
                }
            }
        }

        return ansRow;
    }
}
```