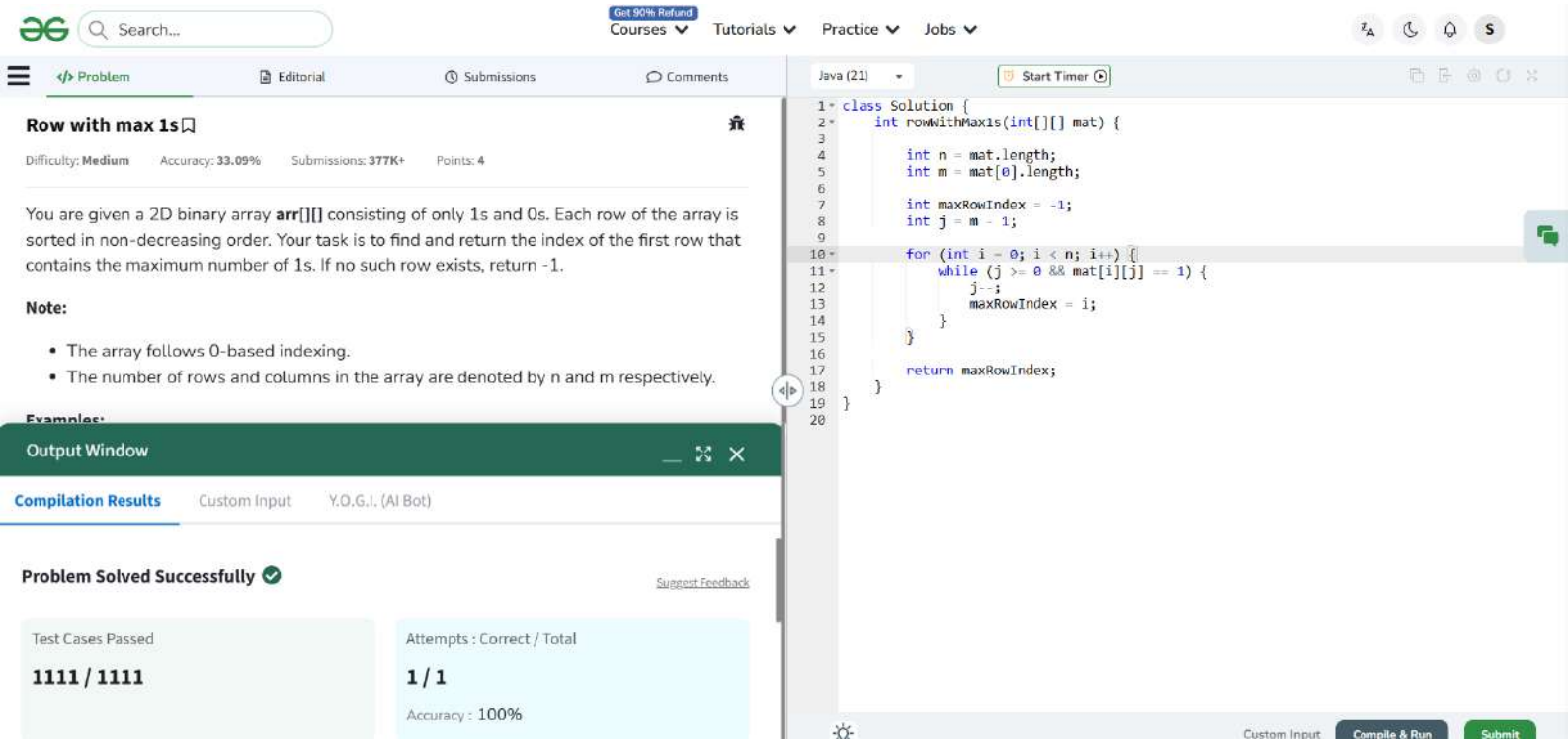
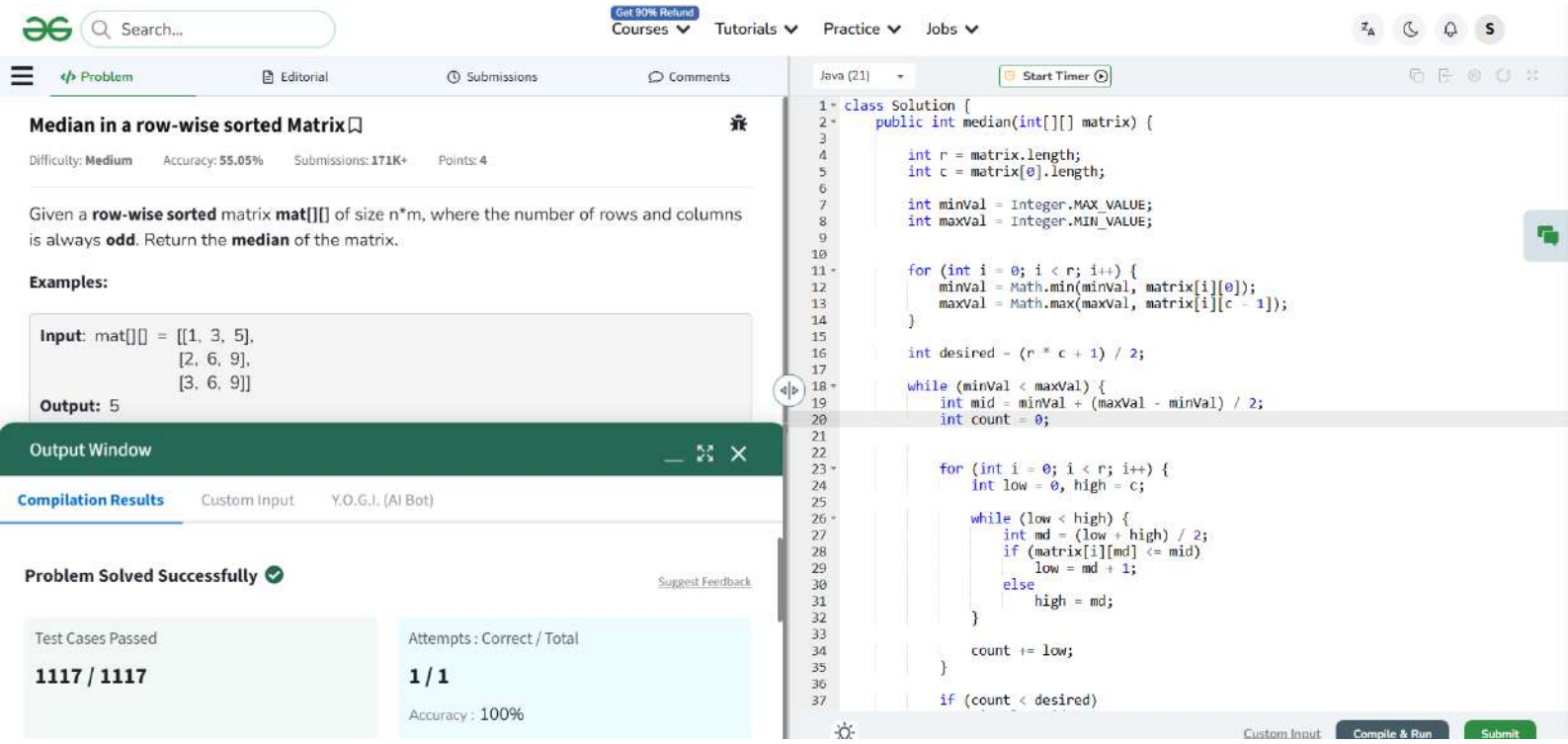
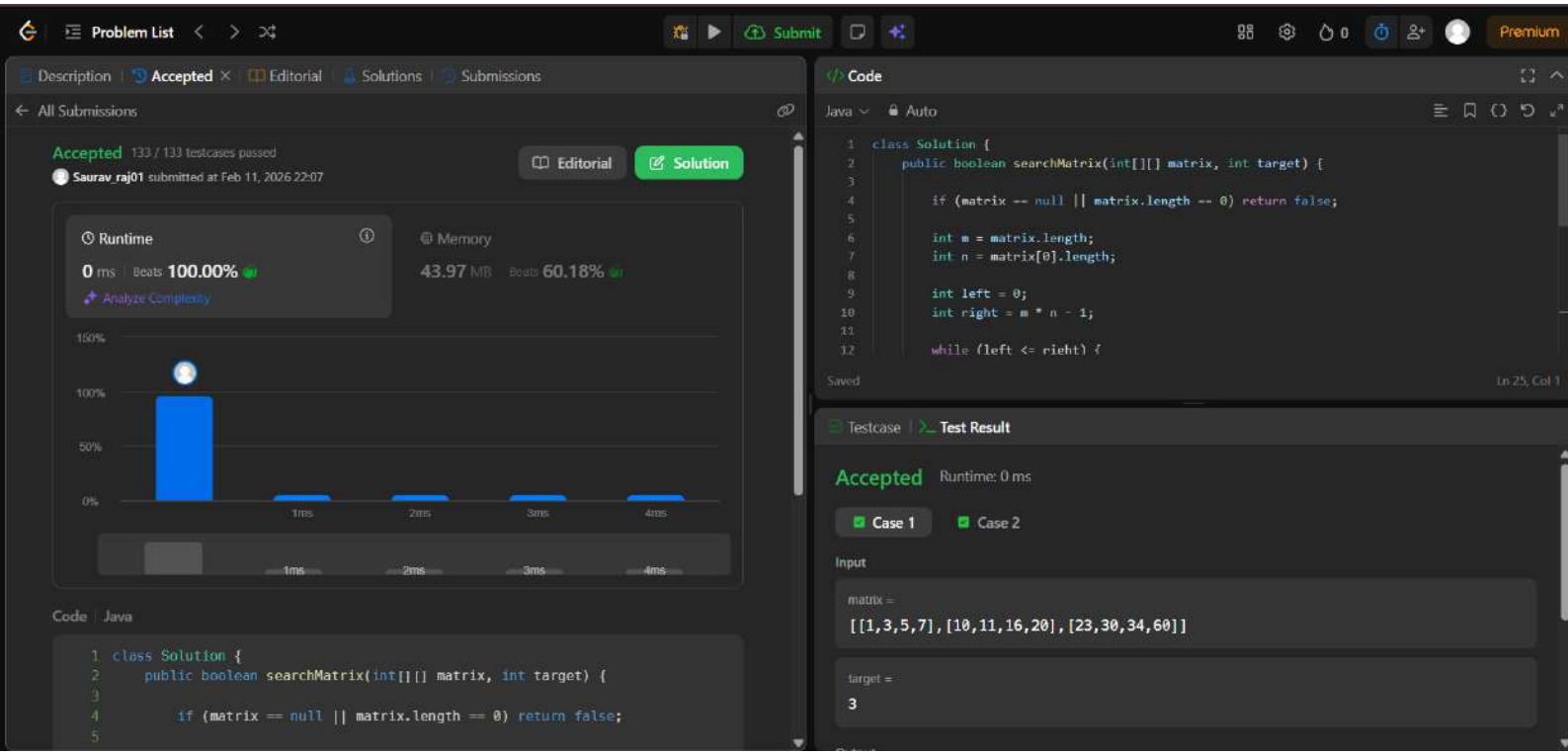
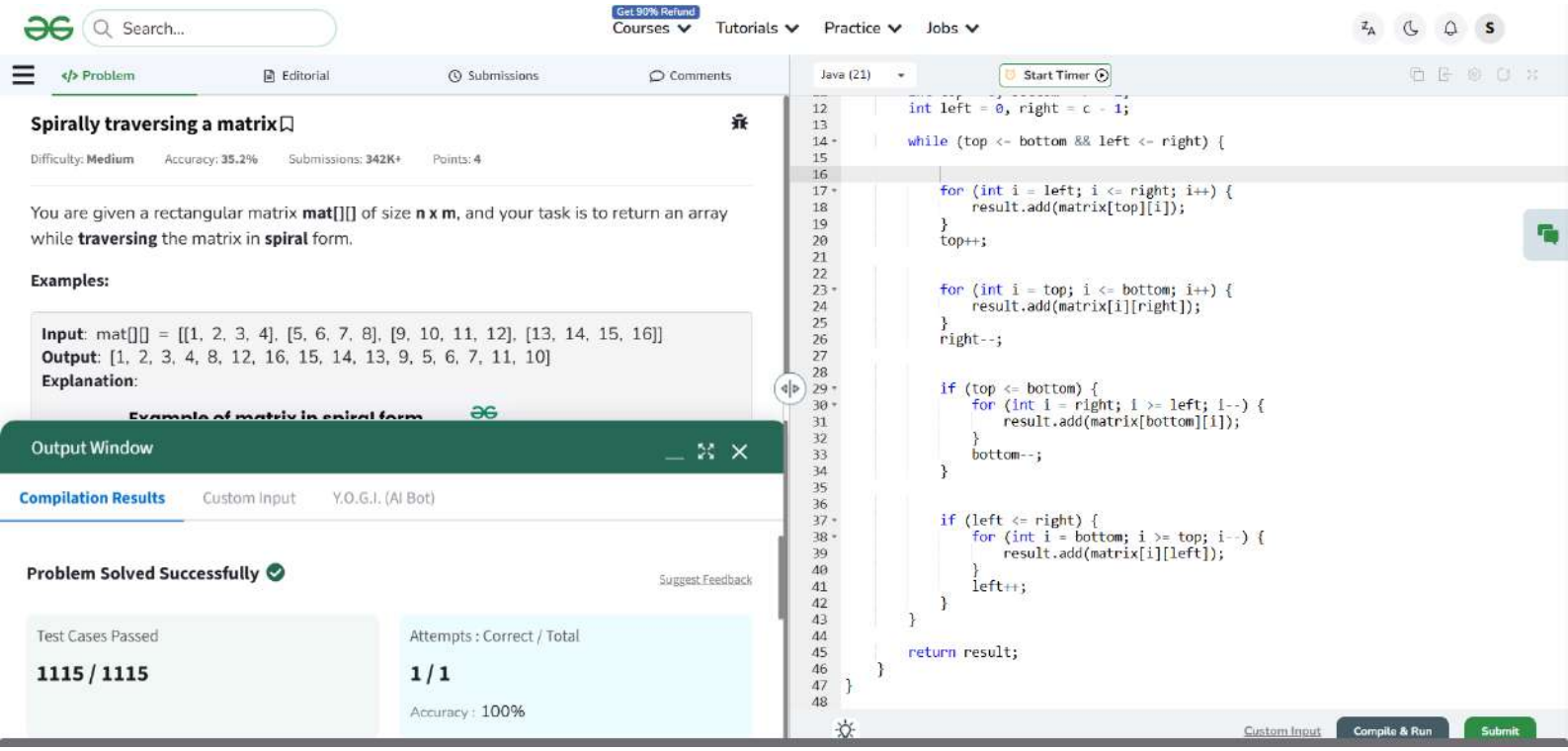
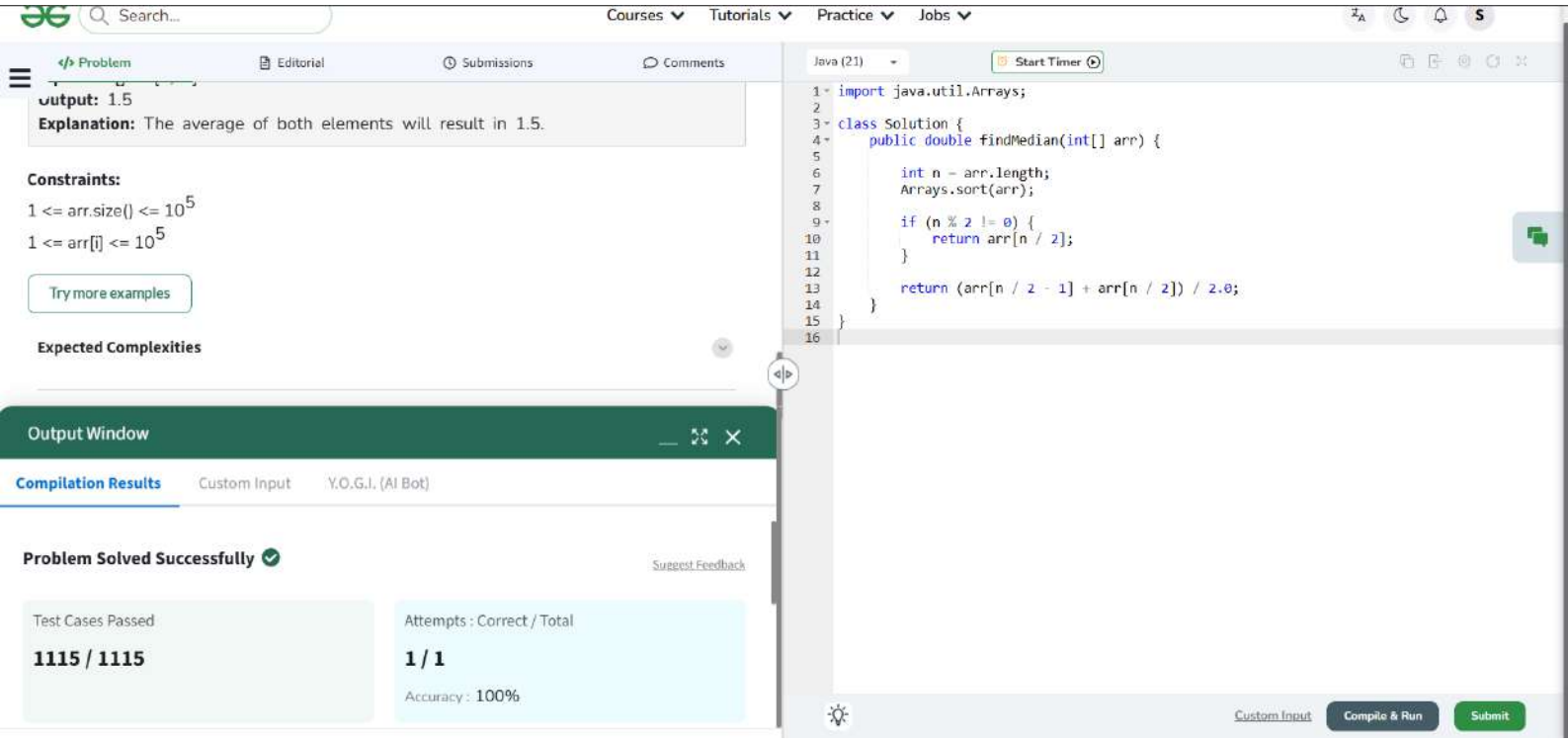
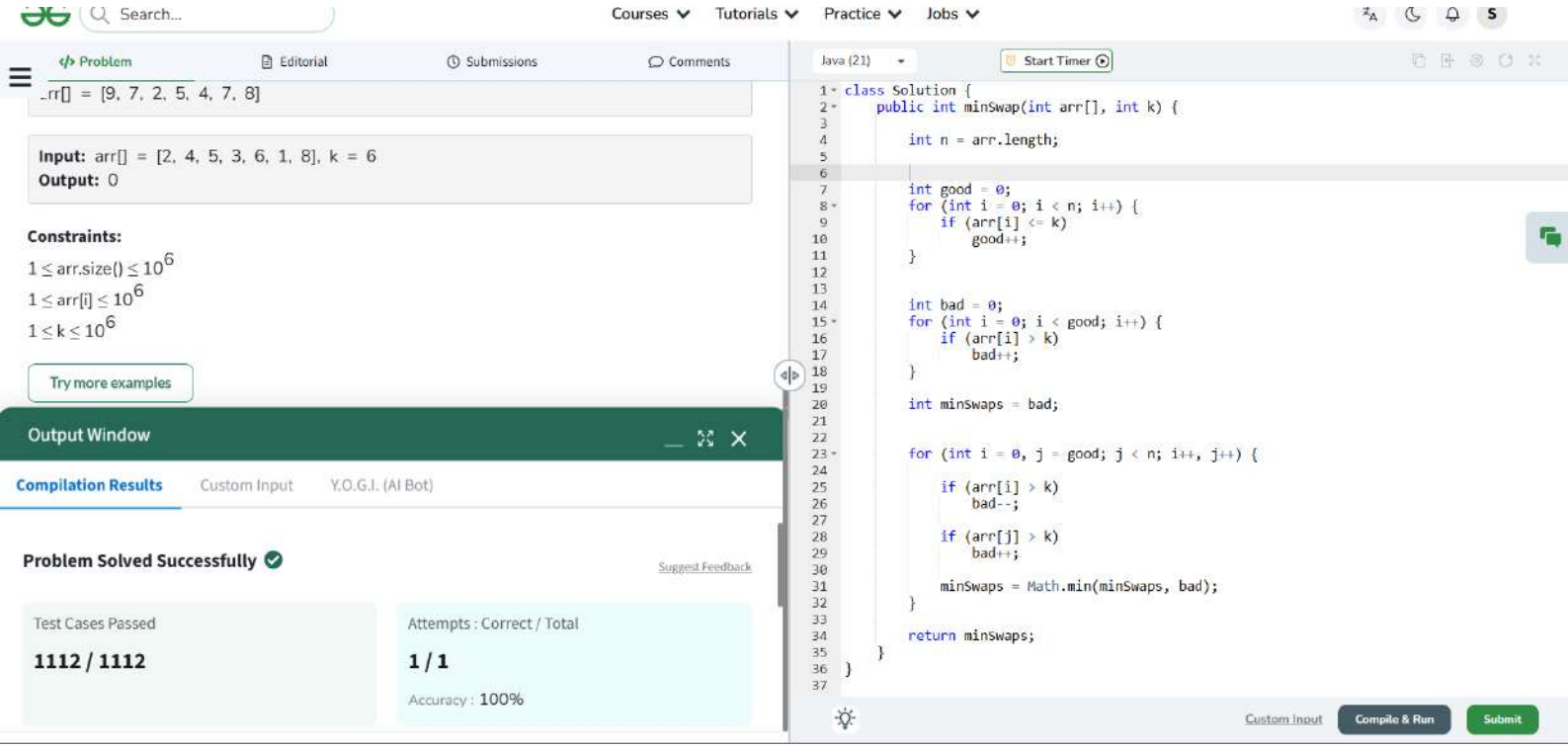

Search...
Get 90% Refund
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
Java (21)
Start Timer
Row with max 1s
Difficulty: Medium
Accuracy: 33.09%
Submissions: 377K+
Points: 4
You are given a 2D binary array arr[][] consisting of only 1s and 0s. Each row of the array is sorted 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:
class Solution {
int rowWithMax1s(int[][] mat) {
int n = mat.length;
int m = mat[0].length;
int maxRowIndex = -1;
int j = m - 1;
for (int i = 0; i < n; i++) {
while (j >= 0 && mat[i][j] == 1) {
j--;
maxRowIndex = i;
}
}
return maxRowIndex;
}
}
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Problem Solved Successfully
Suggest Feedback
Test Cases Passed
1111 / 1111
Attempts : Correct / Total
1 / 1
Accuracy : 100%
Custom Input
Compile & Run
Submit

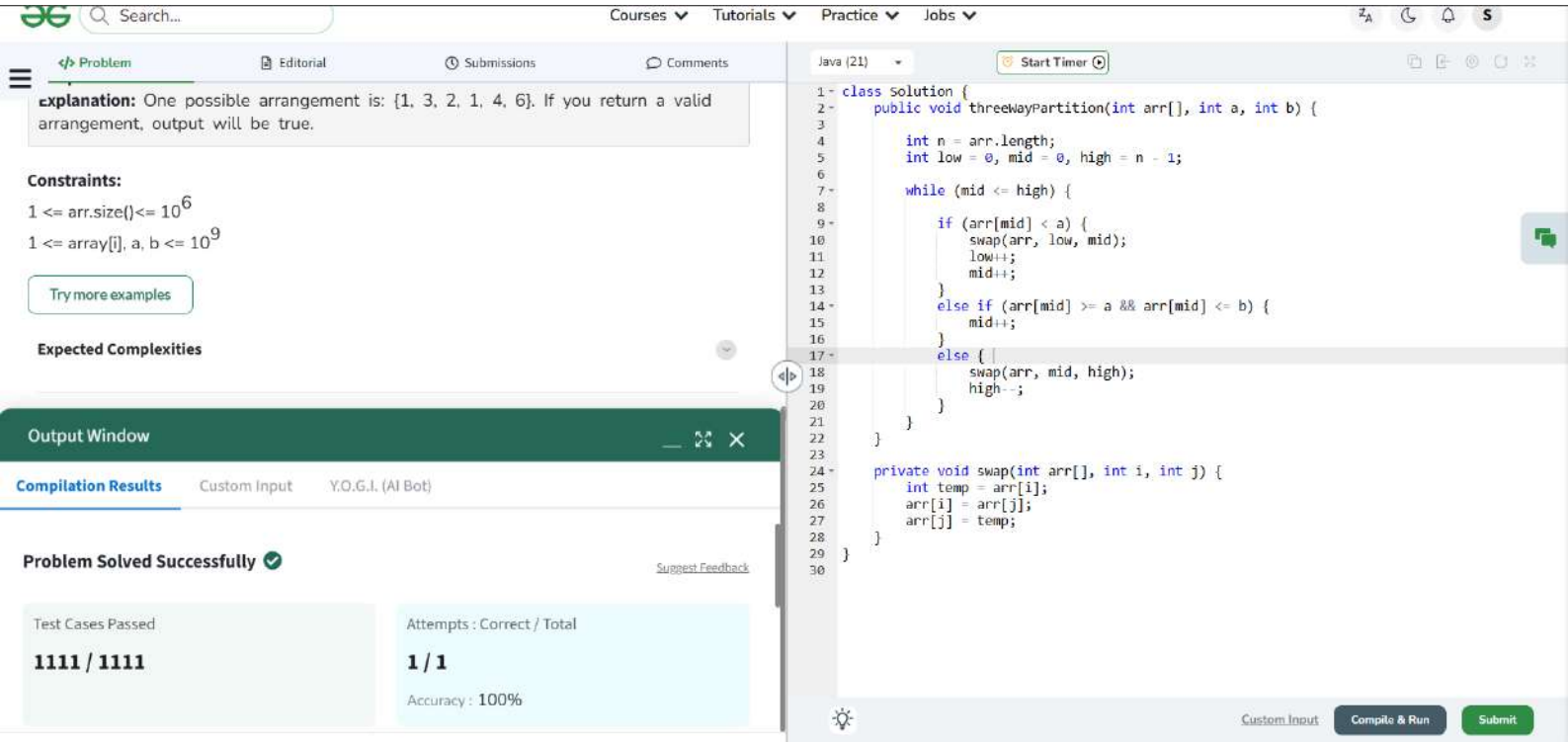

Search...
Get 90% Refund
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
Median in a row-wise sorted Matrix
Difficulty: Medium
Accuracy: 55.05%
Submissions: 171K+
Points: 4
Given a row-wise sorted matrix mat[][] of size n*m, where the number of rows and columns is always odd. Return the median of the matrix.
Examples:
Input: mat[][] = [[1, 3, 5], [2, 6, 9], [3, 6, 9]]
Output: 5
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Problem Solved Successfully
Suggest Feedback
Test Cases Passed
1117 / 1117
Attempts : Correct / Total
1 / 1
Accuracy : 100%
Java (21)
Start Timer
class Solution {
public int median(int[][] matrix) {
int r = matrix.length;
int c = matrix[0].length;
int minVal = Integer.MAX_VALUE;
int maxVal = Integer.MIN_VALUE;
for (int i = 0; i < r; i++) {
minVal = Math.min(minVal, matrix[i][0]);
maxVal = Math.max(maxVal, matrix[i][c - 1]);
}
int desired = (r * c + 1) / 2;
while (minVal < maxVal) {
int mid = minVal + (maxVal - minVal) / 2;
int count = 0;
for (int i = 0; i < r; i++) {
int low = 0, high = c;
while (low < high) {
int md = (low + high) / 2;
if (matrix[i][md] <= mid)
low = md + 1;
else
high = md;
}
count += low;
}
if (count < desired)
Custom Input
Compile & Run
Submit

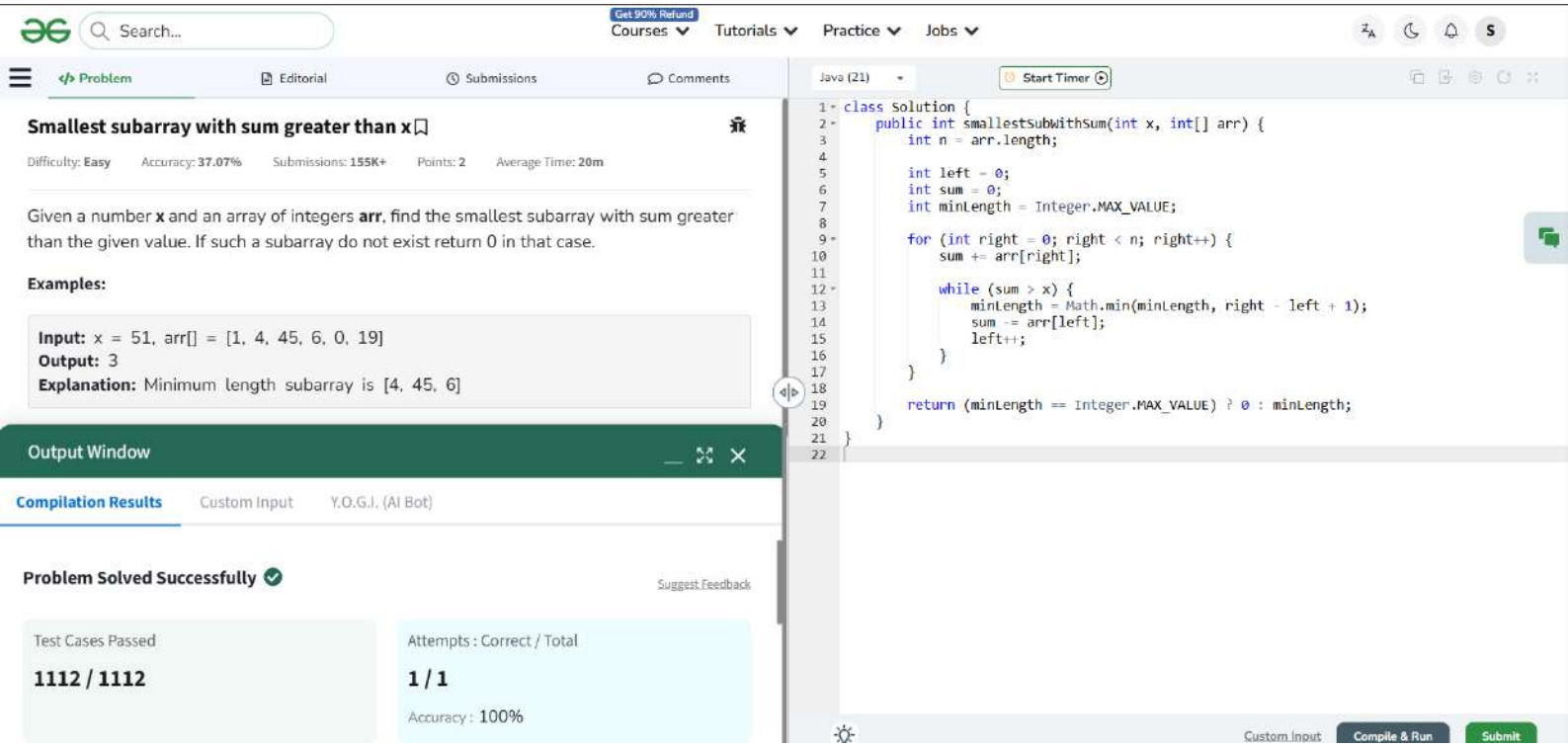

Problem List
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 133 / 133 testcases passed
Saurav_raj01 submitted at Feb 11, 2026 22:07
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
43.97 MB Beats 60.18%
150%
100%
50%
0%
1ms
2ms
3ms
4ms
Code | Java
class Solution {
public boolean searchMatrix(int[][] matrix, int target) {
if (matrix == null || matrix.length == 0) return false;
Code
Java
Auto
class Solution {
public boolean searchMatrix(int[][] matrix, int target) {
if (matrix == null || matrix.length == 0) return false;
int m = matrix.length;
int n = matrix[0].length;
int left = 0;
int right = m * n - 1;
while (left <= right) {
Saved
Ln 25, Col 1
Testcase | Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
Input
matrix =
[[1,3,5,7],[10,11,16,20],[23,30,34,60]]
target =
3

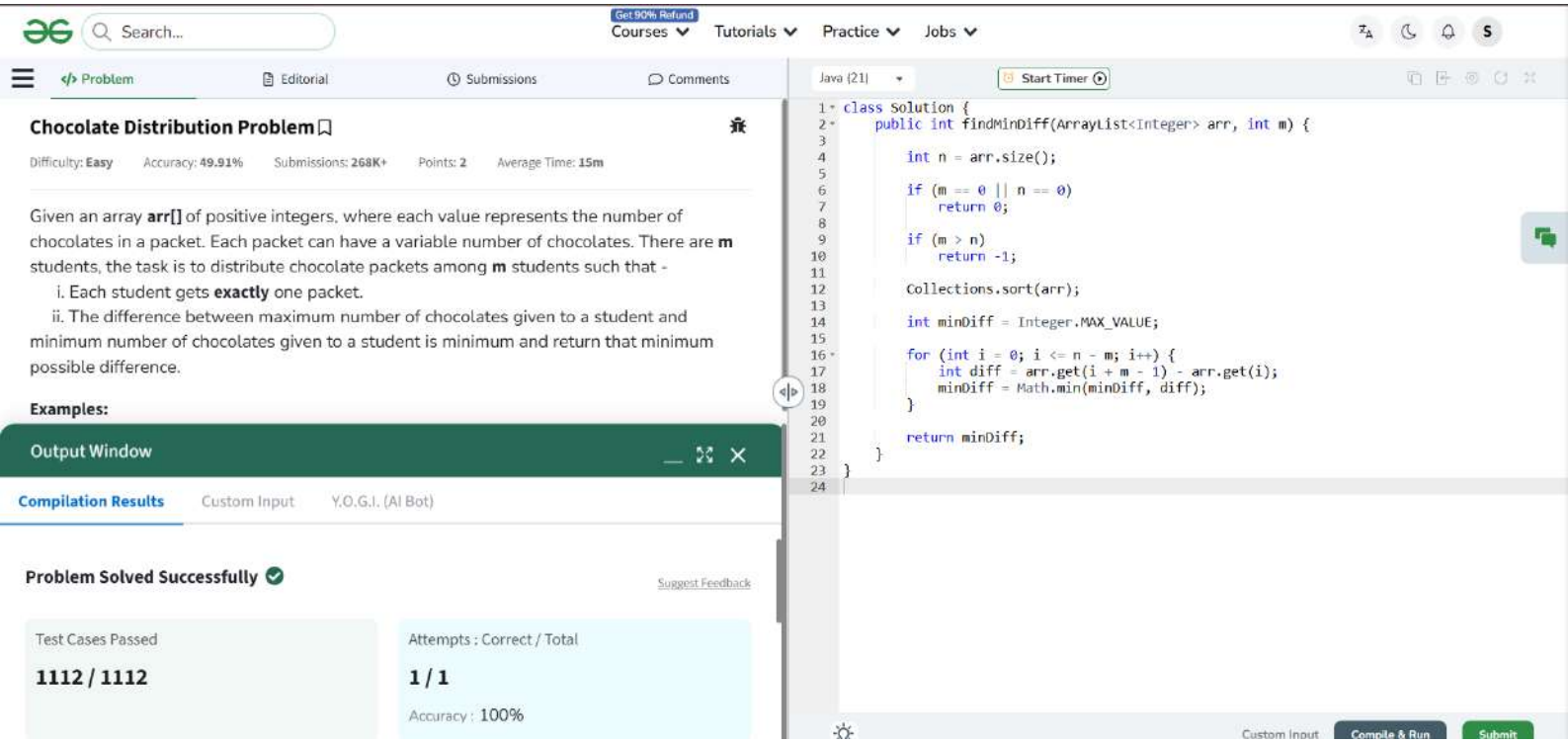

Search...
Get 90% Refund
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
Spirally traversing a matrix
Difficulty: Medium
Accuracy: 35.2%
Submissions: 342K+
Points: 4
You are given a rectangular matrix mat[][] of size n x m, and your task is to return an array while traversing the 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:
Example of matrix in spiral form
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Problem Solved Successfully
Suggest Feedback
Test Cases Passed
1115 / 1115
Attempts : Correct / Total
1 / 1
Accuracy : 100%
Java (21)
Start Timer
int left = 0, right = c - 1;
while (top <= bottom && left <= right) {
for (int i = left; i <= right; i++) {
result.add(matrix[top][i]);
}
top++;
for (int i = top; i <= bottom; i++) {
result.add(matrix[i][right]);
}
right--;
if (top <= bottom) {
for (int i = right; i >= left; i--) {
result.add(matrix[bottom][i]);
}
bottom--;
}
if (left <= right) {
for (int i = bottom; i >= top; i--) {
result.add(matrix[i][left]);
}
left++;
}
}
return result;
}
}
Custom Input
Compile & Run
Submit


Search...
Courses
Tutorials
Practice
Jobs
S
Problem
Editorial
Submissions
Comments
Output: 1.5
Explanation: The average of both elements will result in 1.5.
Constraints:
1 <= arr.size() <= 10^5
1 <= arr[i] <= 10^5
Try more examples
Expected Complexities
Java (21)
Start Timer
1 import java.util.Arrays;
2
3 class Solution {
4 public double findMedian(int[] arr) {
5
6 int n = arr.length;
7 Arrays.sort(arr);
8
9 if (n % 2 != 0) {
10 return arr[n / 2];
11 }
12
13 return (arr[n / 2 - 1] + arr[n / 2]) / 2.0;
14 }
15 }
16
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Problem Solved Successfully
Suggest Feedback
Test Cases Passed
1115 / 1115
Attempts : Correct / Total
1 / 1
Accuracy : 100%
Custom Input
Compile & Run
Submit


Search...
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
_rr[] = [9, 7, 2, 5, 4, 7, 8]
Input: arr[] = [2, 4, 5, 3, 6, 1, 8], k = 6
Output: 0
Constraints:
1 ≤ arr.size() ≤ 10^6
1 ≤ arr[i] ≤ 10^6
1 ≤ k ≤ 10^6
Try more examples
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%
Java (21)
Start Timer
class Solution {
public int minSwap(int arr[], int k) {
int n = arr.length;
int good = 0;
for (int i = 0; i < n; i++) {
if (arr[i] <= k)
good++;
}
int bad = 0;
for (int i = 0; i < good; i++) {
if (arr[i] > k)
bad++;
}
int minSwaps = bad;
for (int i = 0, j = good; j < n; i++, j++) {
if (arr[i] > k)
bad--;
if (arr[j] > k)
bad++;
minSwaps = Math.min(minSwaps, bad);
}
return minSwaps;
}
}
Custom Input
Compile & Run
Submit


Search...
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
explanation: One possible arrangement is: {1, 3, 2, 1, 4, 6}. If you return a valid arrangement, output will be true.
Constraints:
1 <= arr.size()<= 10^6
1 <= array[i], a, b <= 10^9
Try more examples
Expected Complexities
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Problem Solved Successfully
Suggest Feedback
Test Cases Passed
1111 / 1111
Attempts : Correct / Total
1 / 1
Accuracy : 100%
Java (21)
Start Timer
class Solution {
public void threeWayPartition(int arr[], int a, int b) {
int n = arr.length;
int low = 0, mid = 0, high = n - 1;
while (mid <= high) {
if (arr[mid] < a) {
swap(arr, low, mid);
low++;
mid++;
}
else if (arr[mid] >= a && arr[mid] <= b) {
mid++;
}
else {
swap(arr, mid, high);
high--;
}
}
}
private void swap(int arr[], int i, int j) {
int temp = arr[i];
arr[i] = arr[j];
arr[j] = temp;
}
}
Custom Input
Compile & Run
Submit


Search...
Get 90% Refund
Courses
Tutorials
Practice
Jobs
S
Problem
Editorial
Submissions
Comments
Smallest subarray with sum greater than x
Difficulty: Easy
Accuracy: 37.07%
Submissions: 155K+
Points: 2
Average Time: 20m
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]
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%
Java (21)
Start Timer
class Solution {
public int smallestSubWithSum(int x, int[] arr) {
int n = arr.length;
int left = 0;
int sum = 0;
int minLength = Integer.MAX_VALUE;
for (int right = 0; right < n; right++) {
sum += arr[right];
while (sum > x) {
minLength = Math.min(minLength, right - left + 1);
sum -= arr[left];
left++;
}
}
return (minLength == Integer.MAX_VALUE) ? 0 : minLength;
}
}
Custom Input
Compile & Run
Submit


Search...
Get 90% Refund
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
Chocolate Distribution Problem
Difficulty: Easy Accuracy: 49.91% Submissions: 268K+ Points: 2 Average Time: 15m
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:
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%
Java (21)
Start Timer
class Solution {
public int findMinDiff(ArrayList<Integer> arr, int m) {
int n = arr.size();
if (m == 0 || n == 0)
return 0;
if (m > n)
return -1;
Collections.sort(arr);
int minDiff = Integer.MAX_VALUE;
for (int i = 0; i <= n - m; i++) {
int diff = arr.get(i + m - 1) - arr.get(i);
minDiff = Math.min(minDiff, diff);
}
return minDiff;
}
}
Custom Input
Compile & Run
Submit