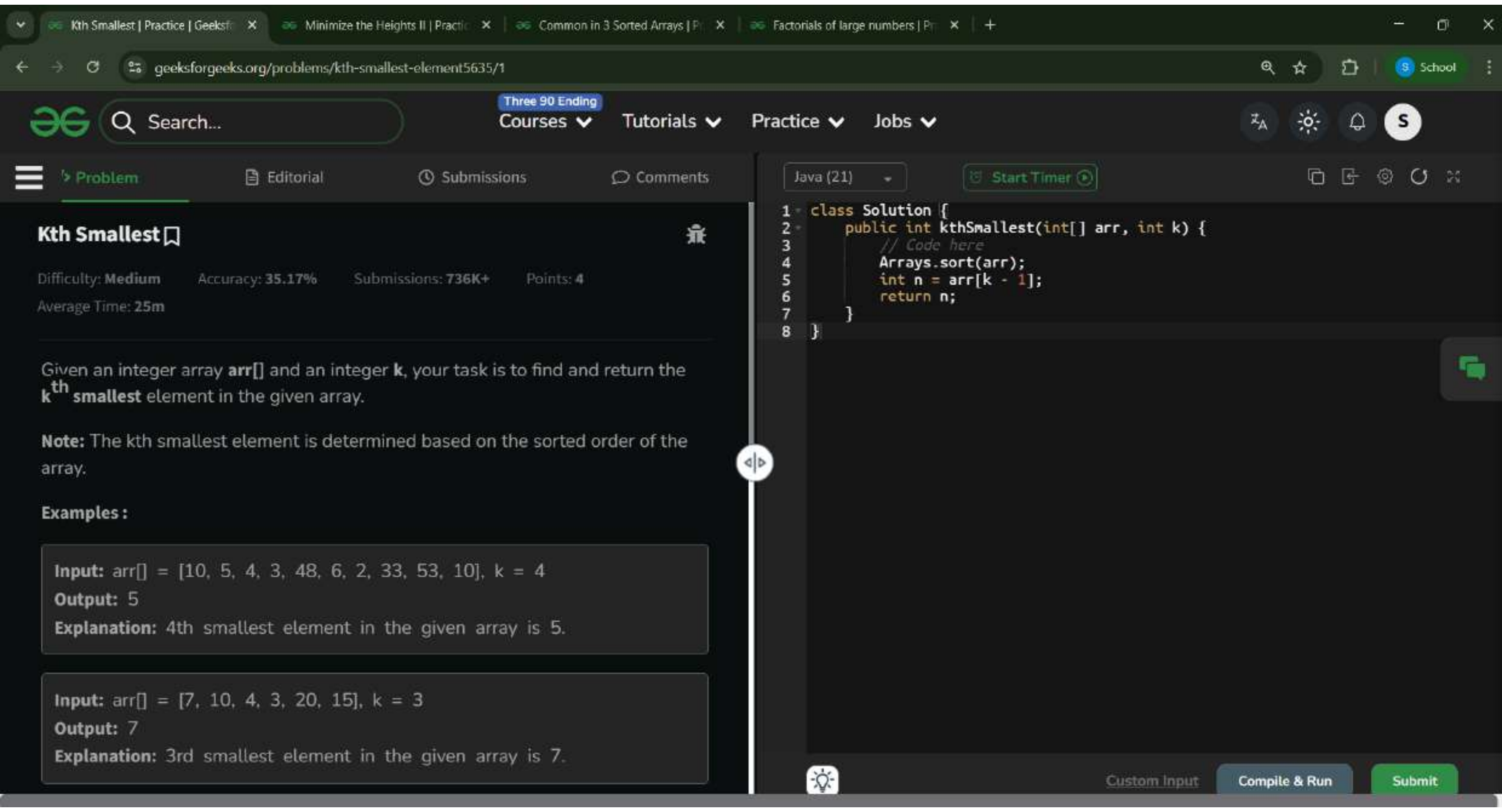

Kth Smallest | Practice | Geeksf
Minimize the Heights II | Practic
Common in 3 Sorted Arrays | P
Factorials of large numbers | P
geeksforgeeks.org/problems/kth-smallest-element5635/1
School
Search...
Three 90 Ending
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
Java (21)
Start Timer
Custom Input
Compile & Run
Submit

Kth Smallest

Difficulty: **Medium** Accuracy: **35.17%** Submissions: **736K+** Points: **4**
Average Time: **25m**

Given an integer array **arr[]** and an integer **k**, your task is to find and return the **k**th **smallest** element in the given array.

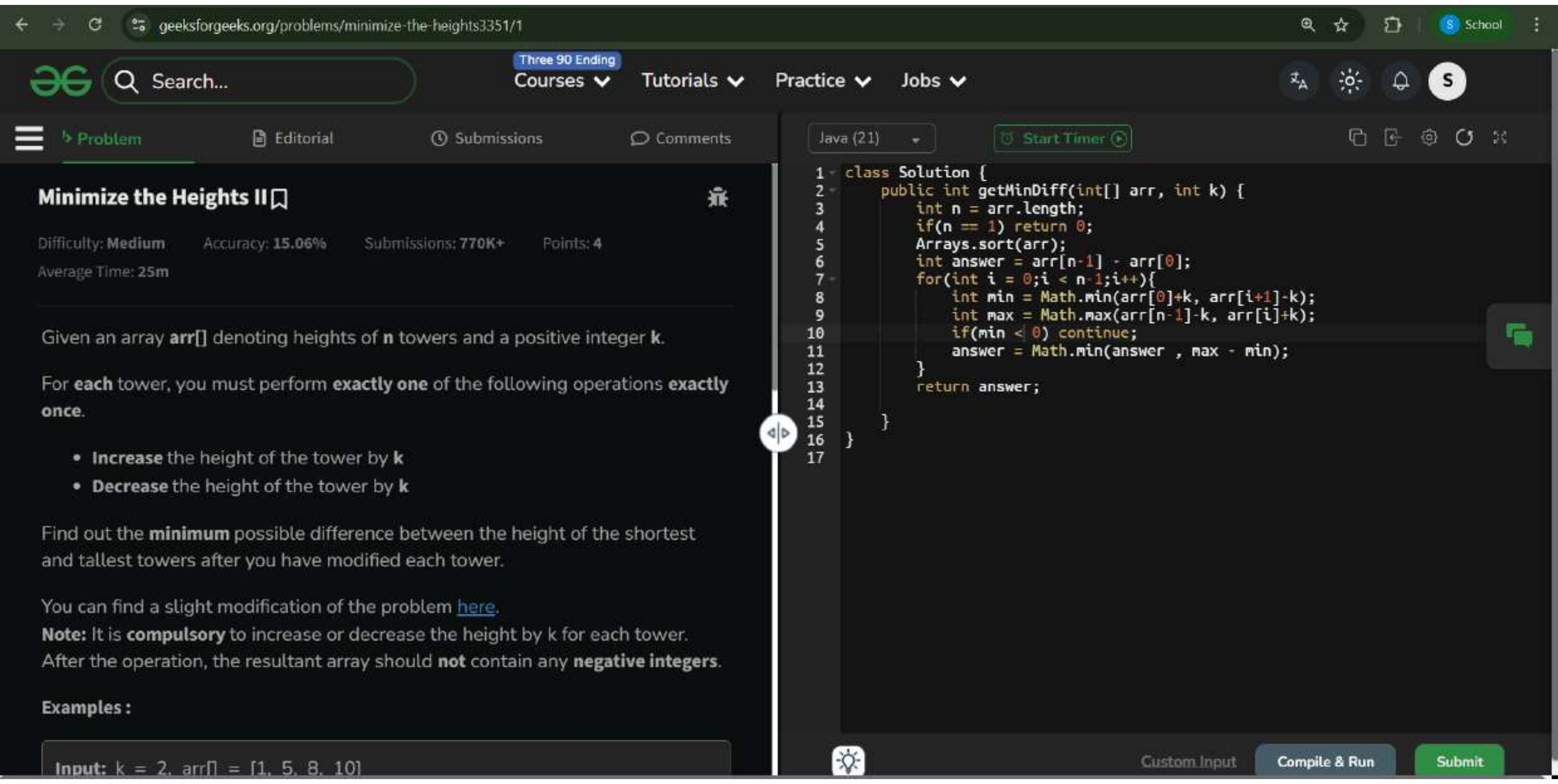
Note: The kth smallest element is determined based on the sorted order of the array.

Examples :

Input: arr[] = [10, 5, 4, 3, 48, 6, 2, 33, 53, 10], k = 4
Output: 5
Explanation: 4th smallest element in the given array is 5.

Input: arr[] = [7, 10, 4, 3, 20, 15], k = 3
Output: 7
Explanation: 3rd smallest element in the given array is 7.

```java
class Solution {
    public int kthSmallest(int[] arr, int k) {
        // Code here
        Arrays.sort(arr);
        int n = arr[k - 1];
        return n;
    }
}
```



Minimize the Heights II

Difficulty: **Medium** Accuracy: **15.06%** Submissions: **770K+** Points: **4**
Average Time: **25m**

Given an array **arr[]** denoting heights of **n** towers and a positive integer **k**.

For **each** tower, you must perform **exactly one** of the following operations **exactly once**.

- **Increase** the height of the tower by **k**
- **Decrease** the height of the tower by **k**

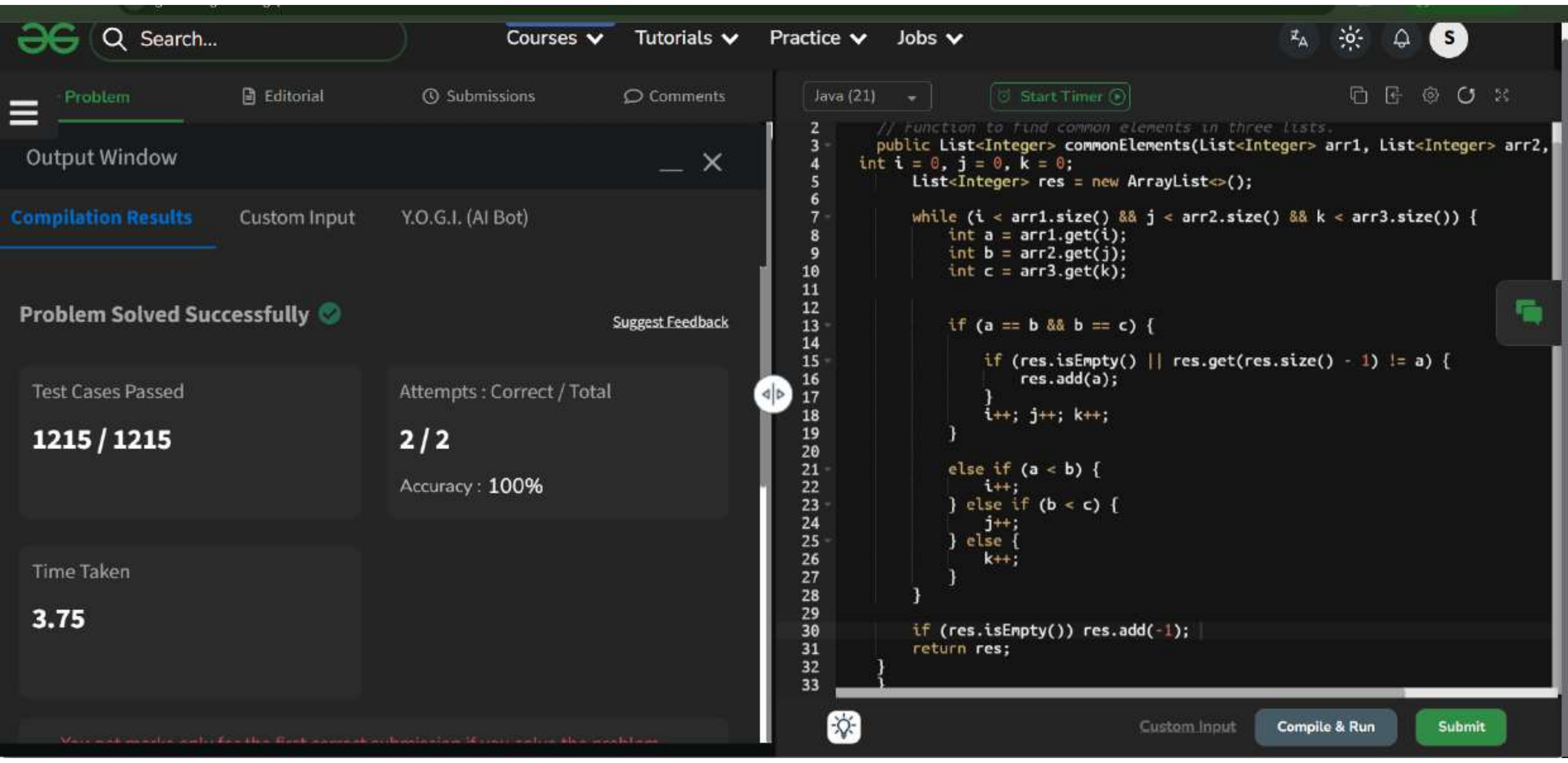
Find out the **minimum** possible difference between the height of the shortest and tallest towers after you have modified each tower.

You can find a slight modification of the problem here.
Note: It is **compulsory** to increase or decrease the height by k for each tower. After the operation, the resultant array should **not** contain any **negative integers**.

Examples :

Input: k = 2, arr[] = [1, 5, 8, 10]

```java
class Solution {
    public int getMinDiff(int[] arr, int k) {
        int n = arr.length;
        if(n == 1) return 0;
        Arrays.sort(arr);
        int answer = arr[n-1] - arr[0];
        for(int i = 0;i < n-1;i++){
            int min = Math.min(arr[0]+k, arr[i+1]-k);
            int max = Math.max(arr[n-1]-k, arr[i]+k);
            if(min < 0) continue;
            answer = Math.min(answer , max - min);
        }
        return answer;

    }
}
```



Output Window

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

Problem Solved Successfully

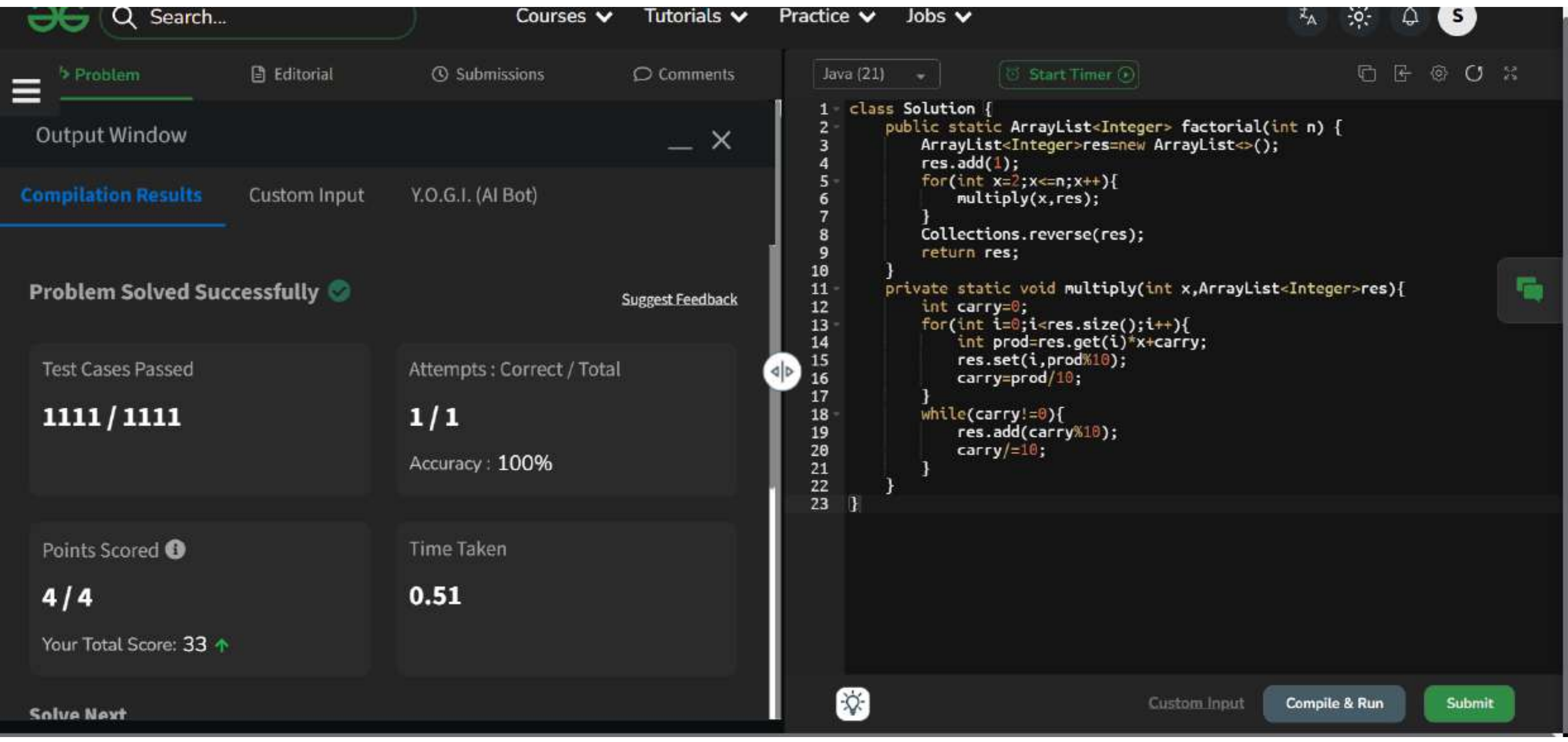
Suggest Feedback

Test Cases Passed
1215 / 1215

Attempts : Correct / Total
2 / 2
Accuracy : 100%

Time Taken
3.75

```java
    // Function to find common elements in three lists.
    public List<Integer> commonElements(List<Integer> arr1, List<Integer> arr2,
  int i = 0, j = 0, k = 0;
        List<Integer> res = new ArrayList<>();

        while (i < arr1.size() && j < arr2.size() && k < arr3.size()) {
            int a = arr1.get(i);
            int b = arr2.get(j);
            int c = arr3.get(k);


            if (a == b && b == c) {

                if (res.isEmpty() || res.get(res.size() - 1) != a) {
                    res.add(a);
                }
                i++; j++; k++;
            }

            else if (a < b) {
                i++;
            } else if (b < c) {
                j++;
            } else {
                k++;
            }
        }

        if (res.isEmpty()) res.add(-1);
        return res;
    }
```



Problem Solved Successfully

Suggest Feedback

Test Cases Passed

1111 / 1111

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

4 / 4

Your Total Score: 33

Time Taken

0.51

Solve Next

```java
class Solution {
    public static ArrayList<Integer> factorial(int n) {
        ArrayList<Integer>res=new ArrayList<>();
        res.add(1);
        for(int x=2;x<=n;x++){
            multiply(x,res);
        }
        Collections.reverse(res);
        return res;
    }
    private static void multiply(int x,ArrayList<Integer>res){
        int carry=0;
        for(int i=0;i<res.size();i++){
            int prod=res.get(i)*x+carry;
            res.set(i,prod%10);
            carry=prod/10;
        }
        while(carry!=0){
            res.add(carry%10);
            carry/=10;
        }
    }
}
```