

DSA CODING PRACTICE – 5

1. kth Smallest Element

The screenshot shows a coding practice interface for the 'kth Smallest Element' problem. The interface includes a navigation bar with links to Courses, Tutorials, Jobs, Practice, and Contests. The problem is solved successfully, with 1110/1110 test cases passed, 1/2 attempts correct, and a time taken of 0.3 seconds. The code is in Java and uses a sorting-based approach.

```
1 // Driver Code Ends
41
42
43 // User function Template for Java
44
45 class Solution {
46     public static int kthSmallest(int[] arr, int k) {
47         // Your code here
48         int s = 0;
49         Arrays.sort(arr);
50         for(int i=0; i<k; i++) {
51             if(i==k-1){
52                 s = arr[i];
53                 break;
54             }
55         }
56         return s;
57     }
58 }
59
```

Time Complexity: $O(n \log n)$

Space Complexity: $O(1)$

2. Minimize the Height 2

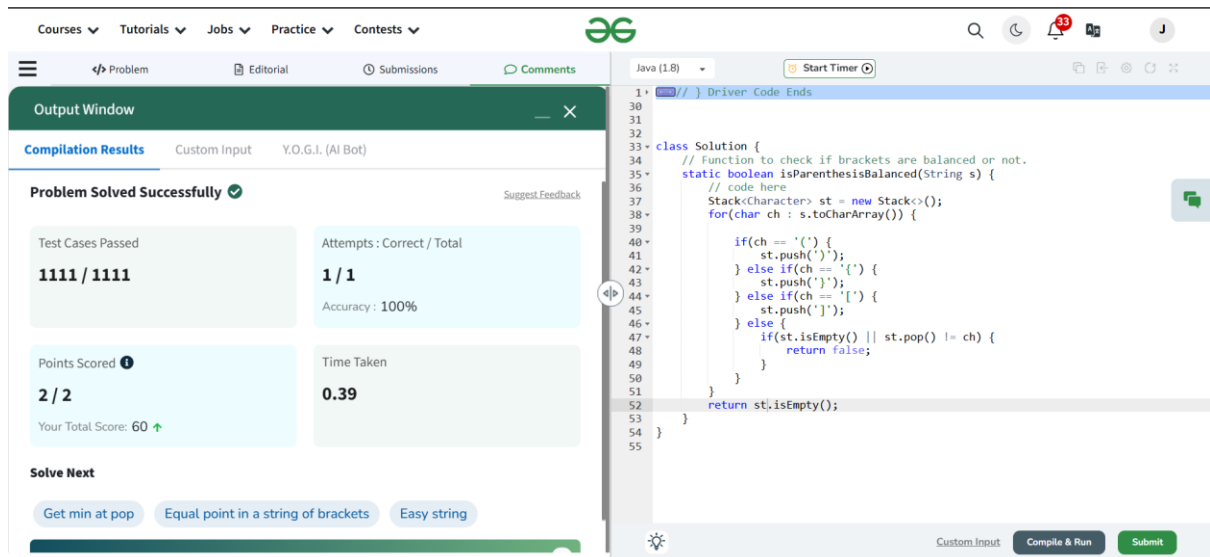
The screenshot shows a coding practice interface for the 'Minimize the Height 2' problem. The interface includes a navigation bar with links to Courses, Tutorials, Jobs, Practice, and Contests. The problem is solved successfully, with 1115/1115 test cases passed, 1/1 attempts correct, and a time taken of 0.8 seconds. The code is in Java and uses a sorting-based approach.

```
1 // Driver Code Ends
34
35
36 // User function Template for Java
37
38 class Solution {
39     int getMinDiff(int[] arr, int k) {
40         // code here
41         int n = arr.length;
42         if(n == 1){
43             return 0;
44         }
45         Arrays.sort(arr);
46
47         int ans = arr[n-1] - arr[0];
48         int min = arr[0] + k;
49         int max = arr[n-1] - k;
50
51         for(int i=0; i<n-1; i++){
52             int min_ele = Math.min(min, arr[i+1]-k);
53             int max_ele = Math.max(max, arr[i]+k);
54             if(min_ele < 0)
55                 continue;
56             ans = Math.min(ans, max_ele - min_ele);
57         }
58         return ans;
59     }
60 }
61
```

Time Complexity: $O(n \log n)$

Space Complexity: $O(1)$

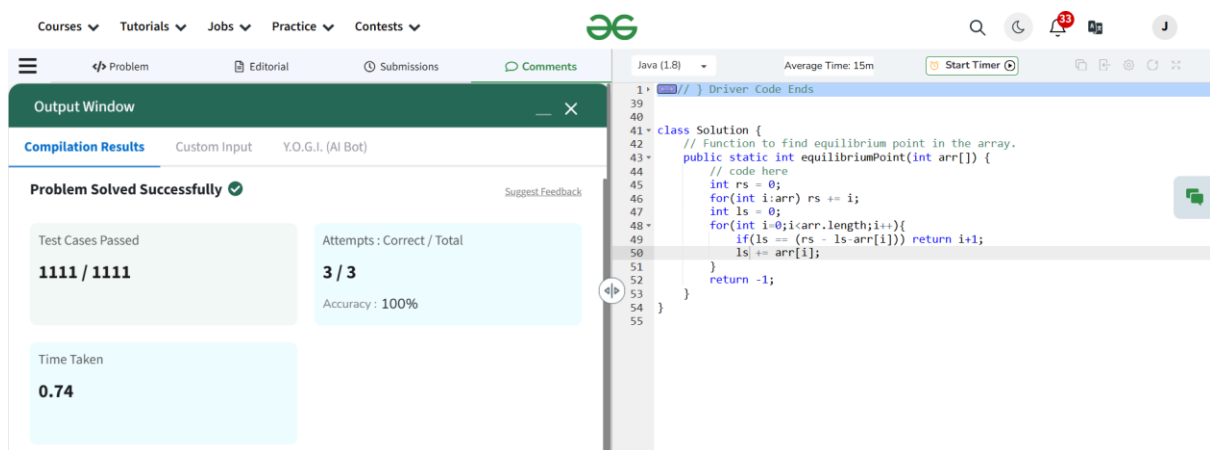
3. Parentheses Checker



Time Complexity: $O(n)$

Space Complexity: $O(n)$

4. Equilibrium Point



Time Complexity: $O(n)$

Space Complexity: $O(1)$

5.Binary Search

Courses ▾ Tutorials ▾ Jobs ▾ Practice ▾ Contests ▾

Java (1.8) Average Time: 20m Start Timer

1 // Driver Code Ends
32
33
34 // User function Template for Java
35
36 class Solution {
37 public int binarysearch(int[] arr, int k) {
38 // Code Here
39 int l=0,r=arr.length-1,mid=0;
40 for(;l<=r;){
41 mid = (l+r)/2;
42 if(arr[mid]>k){
43 r = mid-1;
44 }
45 if(arr[mid]<k){
46 l = mid+1;
47 }
48 if(arr[mid]==k){
49 return mid;
50 }
51 }
52 return -1;
53 }
54 }
55 }

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%

Points Scored ⓘ
2 / 2
Your Total Score: 62 ↑

Time Taken
0.75

Solve Next

Index of an Extra Element Array Search Sorted Array Search

Custom Input Compile & Run Submit

Time Complexity: $O(\log n)$

Space Complexity: $O(1)$

6.Next Greater Element

Courses ▾ Tutorials ▾ Jobs ▾ Practice ▾ Contests ▾

Java (1.8) Average Time: 20m Start Timer

1 // Driver Code Ends
36
37
38 class Solution {
39 // Function to find the next greater element for each element of the array.
40 public ArrayList<Integer> nextLargerElement(int[] arr) {
41 // code here
42 int n=arr.length;
43 ArrayList<Integer> list=new ArrayList<>(n);
44 for(int i=0;i<n;i++){
45 list.add(-1);
46 }
47
48 for(int i=0;i<n;i++){
49 for(int j=i+1;j<n;j++){
50 if(arr[j]>arr[i]){
51 list.set(i,arr[j]);
52 break;
53 }
54 }
55 }
56 return list;
57 }
58 }

Output Window

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

Problem Solved Successfully ✓ Suggest Feedback

Test Cases Passed
1110 / 1110

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

Points Scored ⓘ
4 / 4
Your Total Score: 66 ↑

Time Taken
1.47

Solve Next

Get Min from Stack Maximum Index
Next Greater Element in Circular Array

Custom Input Compile & Run Submit

Time Complexity: $O(n)$

Space Complexity: $O(n)$

7.Union of 2 arrays with Duplicates

The screenshot shows a coding platform interface with the following components:

- Top Navigation:** Courses, Tutorials, Jobs, Practice, Contests.
- Problem Header:** Java (1.8), Average Time: 10m, Start Timer.
- Output Window:** Compilation Results, Custom Input, Y.O.G.I. (AI Bot).
- Problem Status:** Problem Solved Successfully ✓.
- Test Cases:** 1111 / 1111.
- Attempts:** 1 / 1.
- Accuracy:** 100%.
- Points Scored:** 2 / 2.
- Time Taken:** 0.89.
- Solve Next:** Intersection of Arrays with Distinct, LCM of given array elements.
- Code Editor:** Java code for finding the union of two arrays using a HashSet.

```
1 // } Driver Code Ends
2
3
4 // User function Template for Java
5
6 class Solution {
7     public static int findUnion(int a[], int b[]) {
8         // code here
9         Set<Integer> set = new HashSet<Integer>();
10        int max = Math.max(a.length,b.length);
11
12
13
14
15        for(int i=0;i<max;i++){
16
17            if(i<a.length)
18                set.add(a[i]);
19
20            if(i<b.length)
21                set.add(b[i]);
22
23        }
24
25        return set.size();
26    }
27 }
```

Time Complexity: $O((n+m)\log(n+m))$

Space Complexity: $O(n+m)$