

DSA Coding Practice-4

1.Kth Smallest Element:

```
class Solution {
    public static int kthSmallest(int[] arr, int k) {
        return quickSelect(arr, 0, arr.length - 1, k - 1);
    }
    private static int quickSelect(int[] arr, int low, int high, int k) {
        if (low <= high) {
            int pivotIndex = partition(arr, low, high);
            if (pivotIndex == k) {
                return arr[pivotIndex];
            }
            else if (pivotIndex > k) {
                return quickSelect(arr, low, pivotIndex - 1, k);
            }
            else {
                return quickSelect(arr, pivotIndex + 1, high, k);
            }
        }
        return Integer.MAX_VALUE;
    }
    private static int partition(int[] arr, int low, int high) {
        int pivot = arr[high];
        int i = low - 1;
        for (int j = low; j < high; j++) {
            if (arr[j] < pivot) {
                i++;
                swap(arr, i, j);
            }
        }
        swap(arr, i + 1, high);
        return i + 1;
    }
    private static void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
}
```

Courses ▾ Tutorials ▾ Jobs ▾ Practice ▾ Contests ▾

Java (1.8) Average Time: 25m Start Timer

Output Window

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

Problem Solved Successfully ✓

Test Cases Passed: 1110 / 1110

Attempts: Correct / Total: 1 / 1

Accuracy: 100%

Points Scored: 4 / 4

Time Taken: 0.25

Your Total Score: 46 ↑

Solve Next

```
1 // Driver Code Ends
41
42
43 class Solution {
44     public static int kthSmallest(int[] arr, int k) {
45         return quickSelect(arr, 0, arr.length - 1, k - 1);
46     }
47     private static int quickSelect(int[] arr, int low, int high, int k) {
48         if (low <= high) {
49             int pivotIndex = partition(arr, low, high);
50             if (pivotIndex == k) {
51                 return arr[pivotIndex];
52             }
53             else if (pivotIndex > k) {
54                 return quickSelect(arr, low, pivotIndex - 1, k);
55             }
56             else {
57                 return quickSelect(arr, pivotIndex + 1, high, k);
58             }
59         }
60         return Integer.MAX_VALUE;
61     }
62     private static int partition(int[] arr, int low, int high) {
63         int pivot = arr[high];
64         int i = low - 1;
65         for (int j = low; j < high; j++) {
66             if (arr[j] < pivot) {
67                 i++;
68                 swap(arr, i, j);
69             }
70         }
71         swap(arr, i, high);
72         return i;
73     }
74     private static void swap(int[] arr, int i, int j) {
75         int temp = arr[i];
76         arr[i] = arr[j];
77         arr[j] = temp;
78     }
79 }
```

2.Minimize the Heights II:

Courses ▾ Tutorials ▾ Jobs ▾ Practice ▾ Contests ▾

Java (1.8) Average Time: 25m Start Timer

Output Window

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

Problem Solved Successfully ✓

Test Cases Passed: 1115 / 1115

Attempts: Correct / Total: 1 / 1

Accuracy: 100%

Points Scored: 4 / 4

Time Taken: 0.74

Your Total Score: 50 ↑

Solve Next

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