

PROBLEM SHEET 2 Solution 1:

Search...

Get 90% Refund Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Output Window

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

Problem Solved Successfully ✓

Test Cases Passed: 1121 / 1121

Attempts: Correct / Total: 4 / 5

Accuracy: 80%

Time Taken: 0.59

You get marks only for the first correct submission if you solve the problem without...

```
1 class Solution {
2     public int kthSmallest(int[] arr, int k) {
3         int l = 0, r = arr.length - 1;
4
5         while (l <= r) {
6             int p = partition(arr, l, r);
7
8             if (p == k - 1) return arr[p];
9             else if (p > k - 1) r = p - 1;
10            else l = p + 1;
11        }
12        return -1;
13    }
14
15    private int partition(int[] a, int l, int r) {
16        int pivot = a[r], i = l;
17        for (int j = l; j < r; j++) {
18            if (a[j] <= pivot) {
19                int t = a[i]; a[i] = a[j]; a[j] = t;
20                i++;
21            }
22        }
23        int t = a[i]; a[i] = a[r]; a[r] = t;
24        return i;
25    }
26 }
27
28
```

Custom Input Compile & Run Submit

Solution 2:

Search...

Get 90% Refund Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Output Window

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

Problem Solved Successfully ✓

Test Cases Passed: 1115 / 1115

Attempts: Correct / Total: 1 / 1

Accuracy: 100%

Points Scored: 4 / 4

Time Taken: 0.78

Your Total Score: 23 ↑

Solve Next

```
1 import java.util.Arrays;
2
3 class Solution {
4     int getMinDiff(int[] arr, int k) {
5
6         int n = arr.length;
7         Arrays.sort(arr);
8
9         int ans = arr[n - 1] - arr[0];
10
11         int smallest = arr[0] + k;
12         int largest = arr[n - 1] - k;
13
14         for (int i = 0; i < n - 1; i++) {
15
16             int min = Math.min(smallest, arr[i + 1] - k);
17             int max = Math.max(largest, arr[i] + k);
18
19             if (min < 0) continue;
20
21             ans = Math.min(ans, max - min);
22         }
23
24         return ans;
25     }
26 }
27
28
```

Custom Input Compile & Run Submit

Solution 3:

Search...

Get 90% Refund Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Output Window

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

Problem Solved Successfully ✓

Test Cases Passed: 1120 / 1120

Attempts: Correct / Total: 4 / 4

Accuracy: 100%

Time Taken: 0.61

You get marks only for the first correct submission if you solve the problem without...

```
1 class Solution {
2     static int minJumps(int[] arr) {
3
4         int n = arr.length;
5
6         if (n == 1) return 0;
7
8         if (arr[0] == 0) return -1;
9
10        int maxReach = arr[0];
11        int steps = arr[0];
12        int jumps = 1;
13
14        for (int i = 1; i < n; i++) {
15
16            if (i == n - 1) return jumps;
17
18            maxReach = Math.max(maxReach, i + arr[i]);
19
20            steps--;
21
22            if (steps == 0) {
23                jumps++;
24
25                if (i >= maxReach) return -1;
26
27                steps = maxReach - i;
28            }
29        }
30    }
31 }
32
```

Custom Input Compile & Run Submit

Solution 4:

This screenshot shows a LeetCode submission for the problem "Find Duplicate Number". The submission is accepted, with a runtime of 4 ms and memory usage of 83.24 MB. The code implements a Floyd's Cycle-Finding algorithm (slow and fast pointers) to find the duplicate number in an array.

Runtime: 4 ms | Beats 90.89%
Memory: 83.24 MB | Beats 25.70%

Code:

```
1 class Solution {
2     public int findDuplicate(int[] nums) {
3
4         int slow = nums[0];
5         int fast = nums[0];
6
7         do {
8             slow = nums[slow];
9             fast = nums[nums[fast]];
10        } while (slow != fast);
11
12        slow = nums[0];
13        while (slow != fast) {
14
15        }
```

Testcase: Accepted Runtime: 0 ms
Case 1 Case 2 Case 3

Solution 5:

This screenshot shows a LeetCode submission for the problem "Merge Two Sorted Arrays". The submission is accepted, with a runtime of 7 ms and memory usage of 49.31 MB. The code implements a two-pointer approach to merge two sorted arrays.

Runtime: 7 ms | Beats 98.63%
Memory: 49.31 MB | Beats 21.33%

Code:

```
1 import java.util.*;
2
3 class Solution {
4     public int[][] merge(int[][] intervals) {
5
6         Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));
7
8         List<int[]> result = new ArrayList<>();
9
10        int[] current = intervals[0];
11        result.add(current);
12
13        for (int i = 1; i < intervals.length; i++) {
```

Testcase: Accepted Runtime: 1 ms
Case 1 Case 2 Case 3

Solution 6:

This screenshot shows a LeetCode submission for the problem "Merge Two Sorted Arrays". The submission is accepted, with a runtime of 7 ms and memory usage of 49.31 MB. The code implements a two-pointer approach to merge two sorted arrays.

Runtime: 7 ms | Beats 98.63%
Memory: 49.31 MB | Beats 21.33%

Code:

```
1 import java.util.*;
2
3 class Solution {
4     public int[][] merge(int[][] intervals) {
5
6         Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));
7
8         List<int[]> result = new ArrayList<>();
9
10        int[] current = intervals[0];
11        result.add(current);
12
13        for (int i = 1; i < intervals.length; i++) {
```

Testcase: Accepted Runtime: 1 ms
Case 1 Case 2 Case 3

Solution 7:

Output Window

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

Problem Solved Successfully ✓

Test Cases Passed
1215 / 1215

Attempts : Correct / Total
1 / 1

Accuracy : 100%

Points Scored
2 / 2

Your Total Score: 29 ↑

Time Taken
3.82

Solve Next

```
1 import java.util.*;
2
3 public class Solution {
4     public List<Integer> commonElements(List<Integer> arr1, List<Integer> arr2, List<Integer> arr3) {
5         List<Integer> result = new ArrayList<>();
6         int i = 0, j = 0, k = 0;
7         int n1 = arr1.size(), n2 = arr2.size(), n3 = arr3.size();
8
9         while (i < n1 && j < n2 && k < n3) {
10             if (arr1.get(i).equals(arr2.get(j)) && arr2.get(j).equals(arr3.get(k))) {
11                 if (result.isEmpty() || !result.get(result.size() - 1).equals(arr1.get(i))) {
12                     result.add(arr1.get(i));
13                 }
14                 i++;
15                 j++;
16                 k++;
17             } else if (arr1.get(i) < arr2.get(j)) {
18                 i++;
19             } else if (arr2.get(j) < arr3.get(k)) {
20                 j++;
21             } else {
22                 k++;
23             }
24         }
25
26         return result;
27     }
28 }
```

Solution 8:

Output Window

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

Problem Solved Successfully ✓

Test Cases Passed
1111 / 1111

Attempts : Correct / Total
1 / 1

Accuracy : 100%

Points Scored
4 / 4

Your Total Score: 33 ↑

Time Taken
0.62

Solve Next

```
1 import java.util.*;
2
3 public class Solution {
4     public ArrayList<Integer> factorial(int n) {
5         ArrayList<Integer> result = new ArrayList<>();
6         result.add(1);
7
8         for (int i = 2; i <= n; i++) {
9             int carry = 0;
10             for (int j = result.size() - 1; j >= 0; j--) {
11                 int prod = result.get(j) * i + carry;
12                 result.set(j, prod % 10);
13                 carry = prod / 10;
14             }
15             while (carry > 0) {
16                 result.add(0, carry % 10);
17                 carry /= 10;
18             }
19         }
20
21         return result;
22     }
23 }
```

Solution 9:

Output Window

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

Problem Solved Successfully ✓

Test Cases Passed
1114 / 1114

Attempts : Correct / Total
1 / 1

Accuracy : 100%

Points Scored
1 / 1

Your Total Score: 34 ↑

Time Taken
0.57

Solve Next

```
1 import java.util.*;
2
3 public class Solution {
4     public boolean isSubset(int a[], int b[]) {
5         Map<Integer, Integer> countA = new HashMap<>();
6
7         for (int num : a) {
8             countA.put(num, countA.getOrDefault(num, 0) + 1);
9         }
10
11         for (int num : b) {
12             if (!countA.containsKey(num) || countA.get(num) == 0) {
13                 return false;
14             }
15             countA.put(num, countA.get(num) - 1);
16         }
17
18         return true;
19     }
20 }
```

Solution 10:

Output Window

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

Problem Solved Successfully ✓

Test Cases Passed: 1111 / 1111

Attempts: Correct / Total: 1 / 1

Accuracy: 100%

Points Scored: 4 / 4

Time Taken: 0.17

Your Total Score: 38 ↑

Solve Next

```
1 import java.util.*;
2
3 public class Solution {
4     public boolean hasTripletSum(int[] arr, int target) {
5         int n = arr.length;
6         Arrays.sort(arr);
7
8         for (int i = 0; i < n - 2; i++) {
9             int left = i + 1;
10            int right = n - 1;
11
12            while (left < right) {
13                int sum = arr[i] + arr[left] + arr[right];
14                if (sum == target) {
15                    return true;
16                } else if (sum < target) {
17                    left++;
18                } else {
19                    right--;
20                }
21            }
22        }
23        return false;
24    }
25 }
26 }
```

Custom Input Compile & Run Submit

Solution 11:

Output Window

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

Problem Solved Successfully ✓

Test Cases Passed: 1111 / 1111

Attempts: Correct / Total: 1 / 1

Accuracy: 100%

Points Scored: 8 / 8

Time Taken: 0.24

Your Total Score: 46 ↑

Solve Next

```
1 class Solution {
2
3     // Method name matches: maxWater
4     public static long maxWater(int[] arr) {
5
6         int n = arr.length;
7         int left = 0, right = n - 1;
8         int leftMax = 0, rightMax = 0;
9         long water = 0;
10
11         while (left <= right) {
12             if (arr[left] <= arr[right]) {
13                 if (arr[left] >= leftMax)
14                     leftMax = arr[left];
15                 else
16                     water += leftMax - arr[left];
17                 left++;
18             } else {
19                 if (arr[right] >= rightMax)
20                     rightMax = arr[right];
21                 else
22                     water += rightMax - arr[right];
23                 right--;
24             }
25         }
26         return water;
27     }
28 }
```

Custom Input Compile & Run Submit