









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  [Suggest Feedback](#)

Test Cases Passed
1115 / 1115

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


Points Scored 
4 / 4

Your Total Score: **23** 


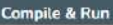

Time Taken
0.69

Solve Next


[A difference of values and indexes](#)[Max Diff Elements and Indexes](#)[Minimize the Heights I](#)

Java (21) 

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





 Custom Input  


Ctrl + Enter



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 


[Suggest Feedback](#)

Test Cases Passed
1121 / 1121

Attempts : Correct / Total
2 / 2

Accuracy : 100%

Time Taken
0.79


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


Solve Next

[Smallest Positive Missing](#) [Valid Pair Sum](#) [Optimal Array](#)

Java (21) Start Timer





```
1 import java.util.Arrays;
2 class Solution {
3     public static int kthSmallest(int[] arr, int k) {
4         Arrays.sort(arr);
5         return arr[k - 1];
6     }
7 }
```


 Custom Input Compile & Run Submit



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 

Suggest Feedback

Test Cases Passed

1120 / 1120


Attempts : Correct / Total

2 / 2

Accuracy : 100%

Time Taken

0.66

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

Solve Next

Maximum Index Jump Game Wine Buying and Selling

Java (21) Start Timer

```
1 class Solution {
2     static int minJumps(int[] arr) {
3         int n = arr.length;
4         if (arr[0] == 0) return -1;
5         int jumps = 0;
6         int farthest = 0;
7         int currentEnd = 0;
8         for (int i = 0; i < n - 1; i++) {
9             farthest = Math.max(farthest, i + arr[i]);
10            if (i == currentEnd) {
11                jumps++;
12                currentEnd = farthest;
13                if (currentEnd <= i) return -1;
14                if (currentEnd >= n - 1) return jumps;
15            }
16        }
17        return (currentEnd >= n - 1) ? jumps : -1;
18    }
19 }
```

Ctrl + Enter

Custom Input Compile & Run Submit

Problem List

Submit

Premium

Description

Accepted

Editorial

Solutions

Submissions

All Submissions

Accepted 59 / 59 testcases passed

vt2812 submitted at Feb 13, 2026 00:08

Editorial

Solution

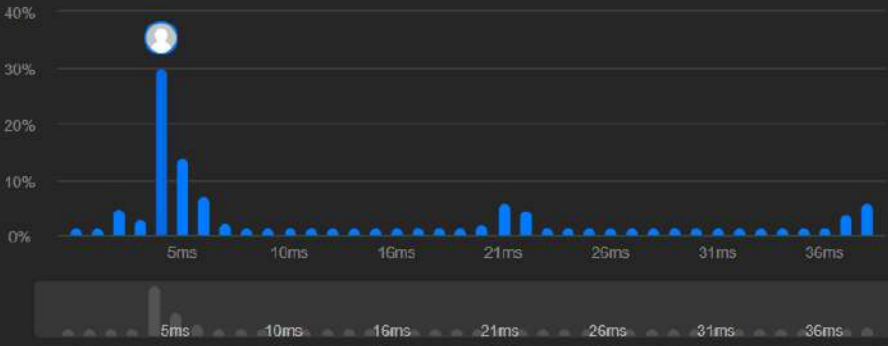
Runtime

4 ms | Beats 90.97%

Analyze Complexity

Memory

82.83 MB | Beats 68.40%



Runtime (ms)	Percentage (%)
5	35
6	15
7	8
21	5
22	5
36	5

Code | Java

```
1 class Solution {
2     public int findDuplicate(int[] nums) {
3         int tortoise = nums[0];
4         int hare = nums[0];
5         do {
```

Code

Java

Auto

```
1 class Solution {
2     public int findDuplicate(int[] nums) {
3         int tortoise = nums[0];
4         int hare = nums[0];
5         do {
6             tortoise = nums[tortoise];
7             hare = nums[nums[hare]];
8         } while (tortoise != hare);
9         tortoise = nums[0];
10        while (tortoise != hare) {
11            tortoise = nums[tortoise];
12            hare = nums[hare];
13        }
14        return hare;
15    }
16 }
```

Saved

Ln 13, Col 10

Testcase

Test Result

Accepted


Runtime: 0 ms

Case 1

Case 2

Case 3

Input



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

1111 / 1111

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

4 / 4

Your Total Score: 27

Time Taken

0.77

Solve Next






Median of 2 Sorted Arrays of Different Sizes

Nth Natural Number

Smallest Positive Integer that can not be represented as Sum

Java (21)

Start Timer



```
1- import java.util.Arrays;
2- class Solution {
3-     public void mergeArrays(int[] a, int[] b) {
4-         int n = a.length;
5-         int m = b.length;
6-         int len = n + m;
7-         int gap = (len / 2) + (len % 2);
8-         while (gap > 0) {
9-             int left = 0;
10-            int right = left + gap;
11-            while (right < len) {
12-                if (left < n && right < n) {
13-                    if (a[left] > a[right]) swap(a, a, left, right);
14-                }
15-                else if (left < n && right >= n) {
16-                    if (a[left] > b[right - n]) swap(a, b, left, right - n);
17-                }
18-                else {
19-                    if (b[left - n] > b[right - n]) swap(b, b, left - n, right - n);
20-                }
21-                left++;
22-                right++;
23-            }
24-            if (gap == 1) break;
25-            gap = (gap / 2) + (gap % 2);
26-        }
27-    }
28-    private void swap(int[] arr1, int[] arr2, int i, int j) {
29-        int temp = arr1[i];
30-        arr1[i] = arr2[j];
31-        arr2[j] = temp;
32-    }
33-}
```

Custom Input

Compile & Run

Submit

Problem List

Submit

Premium

Description

Accepted

Editorial

Solutions

Submissions

All Submissions

Accepted

172 / 172 testcases passed

Editorial

Solution

vi2812 submitted at Feb 13, 2026 00:12

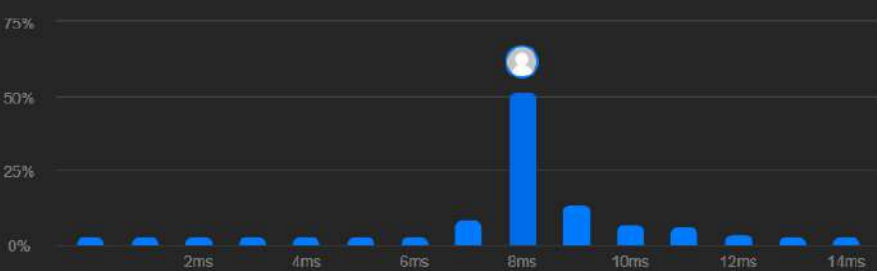
Runtime

8 ms | Beats 90.21%

Analyze Complexity

Memory

49.22 MB | Beats 37.89%



Runtime (ms)	Percentage
2	~2%
4	~2%
6	~2%
8	~60%
10	~15%
12	~5%
14	~2%

Code

Java

```
1 import java.util.Arrays;
2 import java.util.ArrayList;
3 import java.util.List;
4 class Solution {
5     public int[] merge(int[][] intervals) {
```

Code

Java

```
7         return intervals;
8     }
9     Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));
10    List<int[]> merged = new ArrayList<>();
11    int[] currentInterval = intervals[0];
12    merged.add(currentInterval);
13    for (int[] nextInterval : intervals) {
14        int currentEnd = currentInterval[1];
15        int nextStart = nextInterval[0];
16        int nextEnd = nextInterval[1];
17        if (nextStart <= currentEnd) {
18            currentInterval[1] = Math.max(currentEnd, nextEnd);
19        } else {
20            currentInterval = nextInterval;
21            merged.add(currentInterval);
22        }
23    }
24    return merged.toArray(new int[merged.size()][]);
25 }
26 }
```

Testcase

Test Result

Accepted


Runtime: 1 ms

Case 1

Case 2

Case 3

Input



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

1215 / 1215

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

2 / 2

Your Total Score: 29

Time Taken

3.25

Solve Next

Two Repeated Elements

Sorted and Rotated Minimum

Sorted Insert Position

Stay Ahead With:

Suggest Feedback

Java (21)


Start Timer

```
1 class Solution {
2     public List<Integer> commonElements(List<Integer> arr1, List<Integer> arr2,
3                                         List<Integer> arr3) {
4         ArrayList<Integer> ans = new ArrayList<>();
5         int i = 0, j = 0, k = 0;
6         while (i < arr1.size() && j < arr2.size() && k < arr3.size()) {
7             int a = arr1.get(i);
8             int b = arr2.get(j);
9             int c = arr3.get(k);
10            if (a == b && b == c) {
11                if (ans.size() == 0 || ans.get(ans.size() - 1) != a) {
12                    ans.add(a);
13                }
14                i++;
15                j++;
16                k++;
17            }
18            else if (a < b) {
19                i++;
20            }
21            else if (b < c) {
22                j++;
23            }
24            else {
25                k++;
26            }
27        }
28        return ans;
29    }
30 }
31
```





Custom Input


Compile & Run

Submit



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
1111 / 1111

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


Points Scored 
4 / 4
Your Total Score: 33 

Time Taken
0.49




Solve Next

Large Factorial Number following a pattern Rank The Permutations


Stay Ahead With:

Java (21) 

```
1 import java.util.ArrayList;
2 import java.util.Collections;
3 class Solution {
4     public static ArrayList<Integer> factorial(int n) {
5         ArrayList<Integer> res = new ArrayList<>();
6         res.add(1);
7         for (int x = 2; x <= n; x++) {
8             multiply(res, x);
9         }
10        Collections.reverse(res);
11        return res;
12    }
13    private static void multiply(ArrayList<Integer> res, int x) {
14        int carry = 0;
15        for (int i = 0; i < res.size(); i++) {
16            int prod = res.get(i) * x + carry;
17            res.set(i, prod % 10);
18            carry = prod / 10;
19        }
20        while (carry != 0) {
21            res.add(carry % 10);
22            carry = carry / 10;
23        }
24    }
25 }
```





 Custom Input  

Ctrl + Enter



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 

Suggest Feedback


Test Cases Passed

1114 / 1114


Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored 

1 / 1

Your Total Score: 34 

Time Taken

0.63

Solve Next

Counting elements in two arrays






Union of 2 Sorted Arrays

Left most and right most index

Start Head With:


Java (21)

Start Timer

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


Ctrl + Enter



Custom Input

Compile & Run

Submit



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

1111 / 1111

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

4 / 4

Your Total Score: 38

Time Taken

0.15

Solve Next

Sort Elements by Decreasing Frequency

Zero Sum Subarrays

Triplets with Smaller Sum

Get 90% Refund

Java (21)

Start Timer

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

Ctrl + Enter

Custom Input

Compile & Run

Submit

[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

[Suggest Feedback](#)

Test Cases Passed

1111 / 1111

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

8 / 8

Your Total Score: 46

Time Taken

0.23**Solve Next**[Longest Arithmetic Subsequence](#)[Rod Cutting](#)[Jump Game](#)**Stay Ahead With:**

Java (21)

[Start Timer](#)

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

[Custom Input](#)[Compile & Run](#)[Submit](#)