

Kth Smallest | Practice | GeeksforGeeks

https://www.geeksforgeeks.org/problems/kth-smallest-element5635/1

Get 90% Refund

Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Kth Smallest

Difficulty: Medium Accuracy: 35.17% Submissions: 738K+ Points: 4 Average Time: 25m

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

Note: The `k`th 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.

Constraints:

$1 \leq \text{arr.size()} \leq 10^5$
 $1 \leq \text{arr}[i] \leq 10^5$
 $1 \leq k \leq \text{arr.size()}$

Try more examples

Java (21) Start Timer

```
1 class Solution {  
2     public int kthsmallest(int[] arr, int k) {  
3         // Code here  
4         Arrays.sort(arr);  
5  
6         return arr[k-1];  
7     }  
8 }  
9 }
```

Custom Input Compile & Run Submit

Kth Smallest | Practice | GeeksforGeeks

https://www.geeksforgeeks.org/problems/kth-smallest-element5635/1

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 3 / 3 Accuracy : 100%

Time Taken 0.73

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

Custom Input Compile & Run Submit

```
Java (21) Start Timer
1 class Solution {
2     public int kthsmallest(int[] arr, int k) {
3         // Code here
4         Arrays.sort(arr);
5
6         return arr[k-1];
7     }
8 }
9 }
```



Kth Smallest | Practice | GeeksforGeeks X Minimize the Heights II | Practice X +

https://www.geeksforgeeks.org/problems/minimize-the-heights3351/1

Get 90% Refund Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Minimize the Heights II

Difficulty: Medium Accuracy: 15.06% Submissions: 771K+ 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.

Output Window

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

Compilation Completed

Case 1

Input: k =

Java (21) Start Timer

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

Custom Input Compile & Run Submit

Kth Smallest | Practice | GeeksforGeeks X Minimize the Heights II | Practice X +

https://www.geeksforgeeks.org/problems/minimize-the-heights3351/1

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 Time Taken 0.66

Your Total Score: 23 ↑

Solve Next

A difference of values and indexes Max Diff Elements and Indexes

Minimize the Heights I

Stay Ahead With:

Custom Input Compile & Run Submit

Ctrl + Enter

```
Java (21) Start Timer
1 class Solution {
2     public int getMinDiff(int[] arr, int k) {
3         // code here
4         int n = arr.length;
5
6         Arrays.sort(arr);
7
8         int ans = arr[n - 1] - arr[0];
9
10        int smallest = arr[0] + k;
11        int largest = arr[n - 1] - k;
12
13        for (int i = 1; i < n; i++) {
14
15            int minHeight = Math.min(smallest, arr[i] - k);
16            int maxHeight = Math.max(largest, arr[i - 1] + k);
17
18            if (minHeight < 0) {
19                continue;
20            }
21
22            ans = Math.min(ans, maxHeight - minHeight);
23
24        }
25
26    }
27
28
29 }
```

Minimum Jumps | Practice | GeeksX

https://www.geeksforgeeks.org/problems/minimum-number-of-jumps-1587115620/1

Get 90% Refund Courses Tutorials Practice Jobs

Search... Problem Editorial Submissions Comments

Minimum Jumps

Difficulty: Medium Accuracy: 11.91% Submissions: 1.1M Points: 4

You are given an array `arr[]` of non-negative numbers. Each number tells you the **maximum number of steps** you can jump forward from that position.

For example:

- If `arr[i] = 3`, you can jump to index `i + 1, i + 2, or i + 3` from position `i`.
- If `arr[i] = 0`, you **cannot jump forward** from that position.

Your task is to find the **minimum number of jumps** needed to move from the **first** position in the array to the **last** position.

Note: Return `-1` if you can't reach the end of the array.

Examples :

Input: arr[] = [1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9]
Output: 3
Explanation: First jump from 1st element to 2nd element with value 3. From here we jump to 5th element with value 9, and from here we will jump to the last.

Input: arr = [1, 4, 3, 2, 6, 7]
Output: 2
Explanation: First we jump from the 1st to 2nd element and then jump to the

```
Java (21) Start Timer
1+ class Solution {
2+     public int minJumps(int[] arr) {
3+         // code here
4+         int n = arr.length;
5+
6+         if (n == 1) {
7+             return 0;
8+         }
9+
10+        if (arr[0] == 0) {
11+            return -1;
12+        }
13+
14+        int jumps = 1;
15+        int maxReach = arr[0];
16+        int steps = arr[0];
17+
18+        for (int i = 1; i < n; i++) {
19+
20+            if (i == n - 1) {
21+                return jumps;
22+            }
23+
24+            maxReach = Math.max(maxReach, i + arr[i]);
25+            steps--;
26+
27+            if (steps == 0) {
28+                jumps++;
29+
30+                if (i >= maxReach) {
31+                    return -1;
32+                }
33+
34+                steps = maxReach - i;
35+
36+
37+        }
38+
39+    }
40+
41+}
```

Custom Input Compile & Run Submit

Minimum Jumps | Practice | GeeksX

https://www.geeksforgeeks.org/problems/minimum-number-of-jumps-1587115620/1

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
0.62

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+     public int minJumps(int[] arr) {  
3+         // code here  
4+         int n = arr.length;  
5+  
6+         if (n == 1) {  
7+             return 0;  
8+         }  
9+  
10+        if (arr[0] == 0) {  
11+            return -1;  
12+        }  
13+  
14+        int jumps = 1;  
15+        int maxReach = arr[0];  
16+        int steps = arr[0];  
17+  
18+        for (int i = 1; i < n; i++) {  
19+  
20+            if (i == n - 1) {  
21+                return jumps;  
22+            }  
23+  
24+            maxReach = Math.max(maxReach, i + arr[i]);  
25+            steps--;  
26+  
27+            if (steps == 0) {  
28+                jumps++;  
29+  
30+                if (i >= maxReach) {  
31+                    return -1;  
32+                }  
33+  
34+                steps = maxReach - i;  
35+            }  
36+        }  
37+    }
```

Custom Input Compile & Run Submit

Ctrl + Enter

LeetCode - The World's Leading ... | Search - LeetCode | Find the Duplicate Number - Le ... +

leetcode.com/problems/find-the-duplicate-number/ Ask Google

Problem List Description Editorial Solutions Submissions Premium

287. Find the Duplicate Number

Medium Topics Companies

Given an array of integers `nums` containing $n + 1$ integers where each integer is in the range $[1, n]$ inclusive.

There is only **one repeated number** in `nums`, return *this repeated number*.

You must solve the problem **without** modifying the array `nums` and using only constant extra space.

Example 1:

Input: `nums = [1,3,4,2,2]`
Output: 2

Example 2:

Input: `nums = [3,1,3,4,2]`
Output: 3

Example 3:

Input: `nums = [3,3,3,3,3]`
Output: 3

25.3K 492 183 Online 16°C Mostly cloudy ENG IN 19:33 03-02-2026

Code

Java Auto

```
1 class Solution {
2     public int findDuplicate(int[] nums) {
3         int slow = nums[0];
4         int fast = nums[0];
5
6         do {
7             slow = nums[slow];
8             fast = nums[nums[fast]];
9         } while (slow != fast);
10
11         slow = nums[0];
12
13         while (slow != fast) {
14             slow = nums[slow];
15             fast = nums[fast];
16         }
17
18         return slow;
19 }
```

Ln 18, Col 21

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =
[1,3,4,2,2]

LeetCode - The World's Leading x | Search - LeetCode x | Find the Duplicate Number - Le x +

leetcode.com/problems/find-the-duplicate-number/submissions/1906769545/ Ask Google

Problem List < > ☒ Submit ✖ ✖ Premium

Description | Accepted x | Editorial | Solutions | Submissions

All Submissions

Accepted 59 / 59 testcases passed

Adyasha27 submitted at Feb 03, 2026 19:33

Runtime 4 ms | Beats 90.72% Analyze Complexity

Memory 83.29 MB | Beats 25.55%

Runtime distribution chart showing a peak at 5ms.

Code Java Auto

```
1 class Solution {
2     public int findDuplicate(int[] nums) {
3         int slow = nums[0];
4         int fast = nums[0];
5
6         do {
7             slow = nums[slow];
8             fast = nums[nums[fast]];
9         } while (slow != fast);
10
11         slow = nums[0];
12
13         while (slow != fast) {
14             slow = nums[slow];
15             fast = nums[fast];
16         }
17
18         return slow;
19 }
```

Saved Upgrade to Cloud Saving Ln 18, Col 21

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums = [1,3,4,2,2]

16°C Mostly cloudy ENG IN 19:33 03-02-2026

Experiment List 26-01-2026 To 1-0 X Merge Without Extra Space | Pract X +

https://www.geeksforgeeks.org/problems/merge-two-sorted-arrays-1587115620/1

Get 90% Refund Courses Tutorials Practice Jobs

Search... Problem Editorial Submissions Comments

Merge Without Extra Space

Difficulty: Medium Accuracy: 32.01% Submissions: 327K+ Points: 4 Average Time: 20m

Given two sorted arrays **a[]** and **b[]** of size **n** and **m** respectively, the task is to merge them in sorted order without using any **extra space**. Modify **a[]** so that it contains the first **n** elements and modify **b[]** so that it contains the last **m** elements.

Examples:

Input: a[] = [2, 4, 7, 10], b[] = [2, 3]
Output: a[] = [2, 2, 3, 4], b[] = [7, 10]
Explanation: After merging the two non-decreasing arrays, we get, [2, 2, 3, 4, 7, 10]

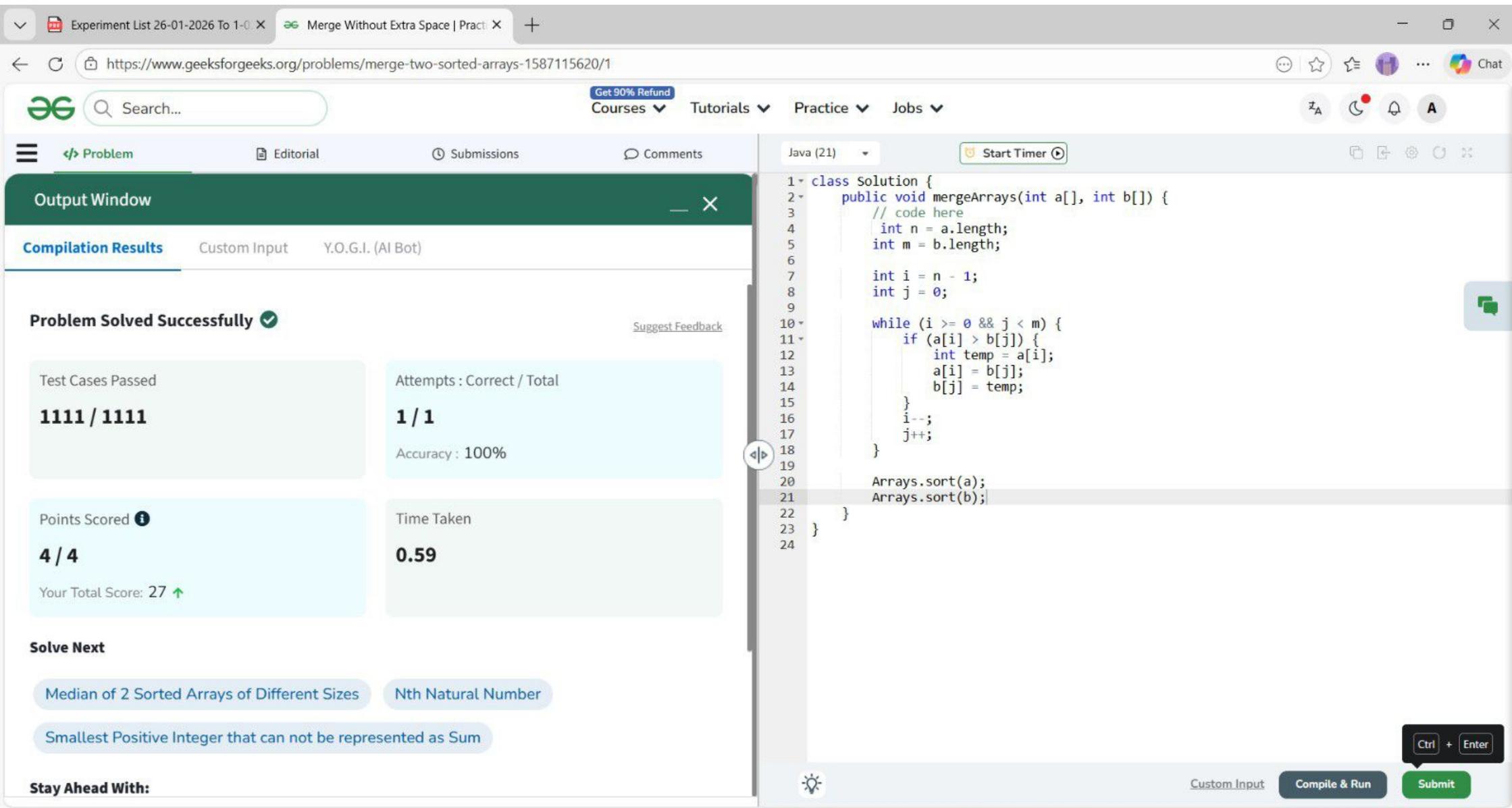
Input: a[] = [1, 5, 9, 10, 15, 20], b[] = [2, 3, 8, 13]
Output: a[] = [1, 2, 3, 5, 8, 9], b[] = [10, 13, 15, 20]
Explanation: After merging two sorted arrays we get [1, 2, 3, 5, 8, 9, 10, 13, 15, 20].

Input: a[] = [0, 1], b[] = [2, 3]
Output: a[] = [0, 1], b[] = [2, 3]
Explanation: After merging two sorted arrays we get [0, 1, 2, 3].

Java (21) Start Timer

```
1 class Solution {  
2     public void mergeArrays(int a[], int b[]) {  
3         // code here  
4         int n = a.length;  
5         int m = b.length;  
6  
7         int i = n - 1;  
8         int j = 0;  
9  
10        while (i >= 0 && j < m) {  
11            if (a[i] > b[j]) {  
12                int temp = a[i];  
13                a[i] = b[j];  
14                b[j] = temp;  
15            }  
16            i--;  
17            j++;  
18        }  
19  
20        Arrays.sort(a);  
21        Arrays.sort(b);  
22    }  
23}  
24}
```

Custom Input Compile & Run Submit



LeetCode - The World's Leading Search - LeetCode Merge Intervals - LeetCode Find the Duplicate Number - LeetCode

leetcode.com/problems/merge-intervals/ Ask Google Premium

Problem List Description Editorial Solutions Submissions

56. Merge Intervals

Medium Topics Companies

Given an array of intervals where `intervals[i] = [starti, endi]`, merge all overlapping intervals, and return an array of the non-overlapping intervals that cover all the intervals in the input.

Example 1:

Input: intervals = [[1,3],[2,6],[8,10],[15,18]]
Output: [[1,6],[8,10],[15,18]]
Explanation: Since intervals [1,3] and [2,6] overlap, merge them into [1,6].

Example 2:

Input: intervals = [[1,4],[4,5]]
Output: [[1,5]]
Explanation: Intervals [1,4] and [4,5] are considered overlapping.

Example 3:

Input: intervals = [[4,7],[1,4]]
Output: [[1,7]]
Explanation: Intervals [1,4] and [4,7] are considered overlapping.

Constraints:

24.4K 264 322 Online

Code

```
Java Auto
1 class Solution {
2     public int[][] merge(int[][] intervals) {
3         int n = intervals.length;
4
5         Arrays.sort(intervals, (x, y) -> x[0] - y[0]);
6
7         ArrayList<int[]> answer = new ArrayList<>();
8
9         int currentStart = intervals[0][0];
10        int currentEnd = intervals[0][1];
11
12        for (int i = 1; i < n; i++) {
```

Ln 26, Col 3

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

intervals = [[1,3],[2,6],[8,10],[15,18]]

Output

[[1,6],[8,10],[15,18]]

LeetCode - The World's Leading Search - LeetCode Merge Intervals - LeetCode Find the Duplicate Number - LeetCode

leetcode.com/problems/merge-intervals/submissions/1906785661/ Ask Google Premium

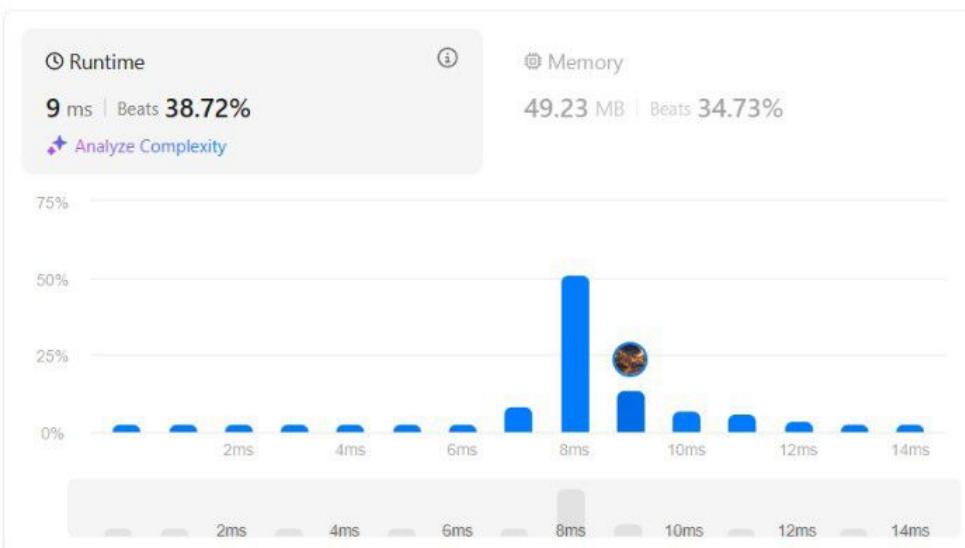
Problem List | Description | Accepted | Editorial | Solutions | Submissions

All Submissions

Accepted 172 / 172 testcases passed Adyasha27 submitted at Feb 03, 2026 19:47

Runtime 9 ms | Beats 38.72% | Analyze Complexity

Memory 49.23 MB | Beats 34.73%



Code

Java Auto

```
1 class Solution {
2     public int[][] merge(int[][] intervals) {
3         int n = intervals.length;
4
5         Arrays.sort(intervals, (x, y) -> x[0] - y[0]);
6
7         ArrayList<int[]> answer = new ArrayList<>();
8
9         int currentStart = intervals[0][0];
10        int currentEnd = intervals[0][1];
11
12        for (int i = 1; i < n; i++) {
13
14            int nextStart = intervals[i][0];
15            int nextEnd = intervals[i][1];
16
17            if (nextStart <= currentEnd) {
18                if (nextEnd > currentEnd) {
19                    currentEnd = nextEnd;
20                }
21            } else {
22                answer.add(new int[]{currentStart, currentEnd});
23                currentStart = nextStart;
24                currentEnd = nextEnd;
25            }
26        }
27
28        answer.add(new int[]{currentStart, currentEnd});
29    }
}
```

Saved Ln 26, Col 3

Testcase | Test Result

Code | Java

```
1 class Solution {
2     public int[][] merge(int[][] intervals) {
3         int n = intervals.length;
4
5         Arrays.sort(intervals, (x, y) -> x[0] - y[0]);
```

16°C Mostly cloudy

Search

19:47 03-02-2026

Experiment List 26-01-2026 To 1-0 X Merge Without Extra Space | Pract X Common in 3 Sorted Arrays | Pract X +

https://www.geeksforgeeks.org/problems/common-elements1132/1

Get 90% Refund Courses Tutorials Practice Jobs

Search... Problem Editorial Submissions Comments Java (21) Start Timer

Common in 3 Sorted Arrays

Difficulty: Easy Accuracy: 22.16% Submissions: 440K+ Points: 2

Given three sorted arrays in **non-decreasing** order, print all common elements in **non-decreasing** order across these arrays. If there are no such elements return an empty array. In this case, the output will be -1.

Note: can you handle the duplicates without using any additional Data Structure?

Examples :

Input: arr1 = [1, 5, 10, 20, 40, 80] , arr2 = [6, 7, 20, 80, 100] , arr3 = [3, 4, 15, 20, 30, 70, 80, 120]
Output: [20, 80]
Explanation: 20 and 80 are the only common elements in arr1, arr2 and arr3.

Input: arr1 = [1, 2, 3, 4, 5] , arr2 = [6, 7] , arr3 = [8,9,10]
Output: [-1]
Explanation: There are no common elements in arr1, arr2 and arr3.

Input: arr1 = [1, 1, 1, 2, 2, 2], arr2 = [1, 1, 2, 2, 2], arr3 = [1, 1, 1, 1, 2, 2, 2, 2]
Output: [1, 2]
Explanation: We do not need to consider duplicates

```
// Code Here
List<Integer> result = new ArrayList<>();

int i = 0;
int j = 0;
int k = 0;

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 (result.size() == 0 || result.get(result.size() - 1) != a) {
            result.add(a);
        }
        i++;
        j++;
        k++;
    } else if (a < b) {
        i++;
    } else if (b < c) {
        j++;
    } else {
        k++;
    }
}

return result;
```

Custom Input Compile & Run Submit

Experiment List 26-01-2026 To 1-0 X Merge Without Extra Space | Pract X Common in 3 Sorted Arrays | Pract X +

https://www.geeksforgeeks.org/problems/common-elements1132/1

Get 90% Refund Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments Java (21) Start Timer

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 1 / 1 Accuracy: 100%

Points Scored 2 / 2 Time Taken 3.33

Your Total Score: 29 ↑

Stay Ahead With:

 Build 21 Projects in 21 Days

Build real-world ML, Deep Learning & Gen AI projects

Register Now →

```
// Code Here
List<Integer> result = new ArrayList<>();

int i = 0;
int j = 0;
int k = 0;

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 (result.size() == 0 || result.get(result.size() - 1) != a) {
            result.add(a);
        }

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

return result;
```

Custom Input Compile & Run Submit Ctrl + Enter

3 15°C Mostly cloudy

Search

20:00 03-02-2026

https://www.geeksforgeeks.org/problems/factorials-of-large-numbers2508/1

Get 90% Refund Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Factorials of large numbers

Difficulty: Medium Accuracy: 36.57% Submissions: 177K+ Points: 4 Average Time: 20m

Given an integer n , find its factorial. Return a list of integers denoting the digits that make up the factorial of n .

Examples:

Input: $n = 5$
Output: [1, 2, 0]
Explanation: $5! = 1*2*3*4*5 = 120$

Input: $n = 10$
Output: [3, 6, 2, 8, 8, 0, 0]
Explanation: $10! = 1*2*3*4*5*6*7*8*9*10 = 3628800$

Input: $n = 1$
Output: [1]
Explanation: $1! = 1$

Constraints:
 $1 \leq n \leq 10^3$

Java (21) Start Timer

```
1 // User function Template for Java
2
3 class Solution {
4     public static ArrayList<Integer> factorial(int n) {
5         // code here
6         ArrayList<Integer> result = new ArrayList<>();
7         result.add(1);
8
9         for (int i = 2; i <= n; i++) {
10
11             int carry = 0;
12
13             for (int j = 0; j < result.size(); j++) {
14
15                 int value = result.get(j) * i + carry;
16                 result.set(j, value % 10);
17                 carry = value / 10;
18             }
19
20             while (carry > 0) {
21                 result.add(carry % 10);
22                 carry = carry / 10;
23             }
24
25             Collections.reverse(result);
26             return result;
27         }
28     }
29 }
```

Custom Input Compile & Run Submit

Factorials of large numbers | Pract X Merge Without Extra Space | Pract X Common in 3 Sorted Arrays | Pract X +

https://www.geeksforgeeks.org/problems/factorials-of-large-numbers2508/1

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 4 / 4 Time Taken 0.55

Your Total Score: 33 ↑

Solve Next

Large Factorial Number following a pattern Rank The Permutations

Stay Ahead With:

Custom Input Compile & Run Submit Ctrl + Enter

```
Java (21) Start Timer
1 // User function Template for Java
2
3 class Solution {
4     public static ArrayList<Integer> factorial(int n) {
5         // code here
6         ArrayList<Integer> result = new ArrayList<>();
7         result.add(1);
8
9         for (int i = 2; i <= n; i++) {
10             int carry = 0;
11
12             for (int j = 0; j < result.size(); j++) {
13                 int value = result.get(j) * i + carry;
14                 result.set(j, value % 10);
15                 carry = value / 10;
16             }
17
18             while (carry > 0) {
19                 result.add(carry % 10);
20                 carry = carry / 10;
21             }
22
23         }
24
25         Collections.reverse(result);
26         return result;
27     }
28 }
29 }
```

Factorials of large numbers | X Merge Without Extra Space | X Common in 3 Sorted Arrays | X Array Subset | Practice | GeekX Experiment List 26-01-2026 T X Experiment List 26-01-2026 T X +

https://www.geeksforgeeks.org/problems/array-subset-of-another-array2317/1

Get 90% Refund Courses Tutorials Practice Jobs

Problem Editorial Submissions Comments

Array Subset

Difficulty: Basic Accuracy: 44.05% Submissions: 518K+ Points: 1 Average Time: 20m

Given two arrays **a[]** and **b[]**, your task is to determine whether **b[]** is a subset of **a[]**.

Examples:

Input: a[] = [11, 7, 1, 13, 21, 3, 7, 3], b[] = [11, 3, 7, 1, 7]
Output: true
Explanation: b[] is a subset of a[]

Input: a[] = [1, 2, 3, 4, 4, 5, 6], b[] = [1, 2, 4]
Output: true
Explanation: b[] is a subset of a[]

Input: a[] = [10, 5, 2, 23, 19], b[] = [19, 5, 3]
Output: false
Explanation: b[] is not a subset of a[]

Constraints:

$1 \leq a.size(), b.size() \leq 10^5$
 $1 \leq a[i], b[j] \leq 10^6$

Two more examples Custom Input Compile & Run Submit

```
Java (21) Start Timer
1
2 class Solution {
3     public boolean isSubset(int a[], int b[]) {
4         // Your code here
5         HashMap<Integer, Integer> map = new HashMap<>();
6
7         for (int i = 0; i < a.length; i++) {
8             map.put(a[i], map.getOrDefault(a[i], 0) + 1);
9         }
10
11        for (int i = 0; i < b.length; i++) {
12            if (!map.containsKey(b[i]) || map.get(b[i]) == 0) {
13                return false;
14            }
15            map.put(b[i], map.get(b[i]) - 1);
16        }
17
18        return true;
19    }
20}
21
22
23
```

Trending videos Golden retriever... Search

2026 03-02-2026 ENG IN

The screenshot shows a LeetCode problem-solving interface. At the top, there's a navigation bar with links for 'Courses', 'Tutorials', 'Practice', and 'Jobs'. On the far right are user profile icons. Below the navigation is a toolbar with buttons for 'Problem', 'Editorial', 'Submissions', 'Comments', and a search bar.

The main area displays the 'Output Window' for a solved problem. It includes sections for 'Compilation Results', 'Custom Input', and 'Y.O.G.I. (AI Bot)'. A message 'Problem Solved Successfully' with a checkmark is shown, along with 'Test Cases Passed' (1114 / 1114), 'Attempts : Correct / Total' (1 / 3), 'Accuracy : 33%', and 'Time Taken' (0.55). Points scored are listed as 1 / 1, and the total score is 34. There's also a 'Suggest Feedback' link.

A large code editor window on the right shows Java code for a solution to the 'Is Subsequence' problem. The code uses a HashMap to count occurrences of each character in array 'a' and then iterates through array 'b' to check if it contains all characters of 'a' in the same order.

At the bottom, there are buttons for 'Custom Input', 'Compile & Run', and 'Submit'. A 'Stay Ahead With:' section lists recommended problems: 'Counting elements in two arrays', 'Union of 2 Sorted Arrays', and 'Left most and right most index'. A keyboard shortcut 'Ctrl + Enter' is also shown.

Factorials of large numbers | X | Merge Without Extra Space | X | Common in 3 Sorted Arrays | X | Array Subset | Practice | GeekX | Triplet Sum in Array | Practice | Experiment List 26-01-2026 | +

https://www.geeksforgeeks.org/problems/triplet-sum-in-array-1587115621/1

Get 90% Refund
Courses ▾ Tutorials ▾ Practice ▾ Jobs ▾

Search...

Problem Editorial Submissions Comments

Triplet Sum in Array

Difficulty: Medium Accuracy: 35.0% Submissions: 361K+ Points: 4 Average Time: 15m

Given an array **arr[]** and an integer **target**, determine if there exists a triplet in the array whose sum equals the given **target**.

Return **true** if such a triplet exists, otherwise, return **false**.

Examples:

Input: arr[] = [1, 4, 45, 6, 10, 8], target = 13
Output: true
Explanation: The triplet {1, 4, 8} sums up to 13.

Input: arr[] = [1, 2, 4, 3, 6, 7], target = 10
Output: true
Explanation: The triplets {1, 3, 6} and {1, 2, 7} both sum to 10.

Input: arr[] = [40, 20, 10, 3, 6, 7], target = 24
Output: false
Explanation: No triplet in the array sums to 24.

Constraints:
 $3 \leq \text{arr.size()} \leq 5 \times 10^3$

Java (21) Start Timer

```
1 class Solution {
2     public boolean hasTripletSum(int arr[], int target) {
3         // code Here
4         int n = arr.length;
5
6         Arrays.sort(arr);
7
8         for (int i = 0; i < n - 2; i++) {
9
10            int left = i + 1;
11            int right = n - 1;
12
13            while (left < right) {
14
15                int sum = arr[i] + arr[left] + arr[right];
16
17                if (sum == target) {
18                    return true;
19                }
20                else if (sum < target) {
21                    left++;
22                }
23                else {
24                    right--;
25                }
26            }
27
28        }
29
30    }
31
32 }
33 }
```

Custom Input

Factorials of large numbers | X Merge Without Extra Space | X Common in 3 Sorted Arrays | X Array Subset | Practice | GeekX Triplet Sum in Array | Practice | Experiment List 26-01-2026 | +

https://www.geeksforgeeks.org/problems/triplet-sum-in-array-1587115621/1

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 4 / 4 Your Total Score: 38 ↑

Solve Next

Sort Elements by Decreasing Frequency Zero Sum Subarrays

Triplets with Smaller Sum

Stay Ahead With:

Custom Input Compile & Run Submit

Ctrl + Enter

```
Java (21) Start Timer
1 class Solution {
2     public boolean hasTripletSum(int arr[], int target) {
3         // code Here
4         int n = arr.length;
5
6         Arrays.sort(arr);
7
8         for (int i = 0; i < n - 2; i++) {
9
10            int left = i + 1;
11            int right = n - 1;
12
13            while (left < right) {
14
15                int sum = arr[i] + arr[left] + arr[right];
16
17                if (sum == target) {
18                    return true;
19                }
20                else if (sum < target) {
21                    left++;
22                }
23                else {
24                    right--;
25                }
26            }
27
28        }
29
30    }
31
32 }
33 }
```

15°C Mostly cloudy

Search

Windows Start

HP Chrome WhatsApp

Custom Input Compile & Run Submit

Ctrl + Enter

20:33 03-02-2026

https://www.geeksforgeeks.org/problems/trapping-rain-water-1587115621/1

Get 90% Refund Courses Tutorials Practice Jobs

Search... Problem Editorial Submissions Comments

Trapping Rain Water

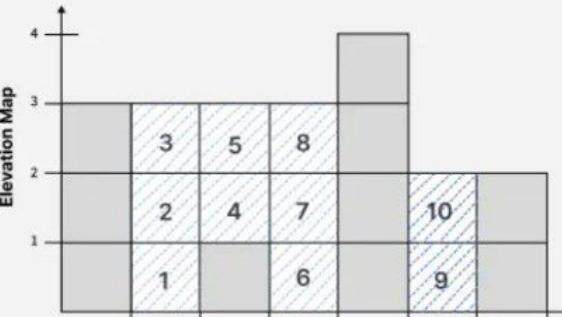
Difficulty: Hard Accuracy: 33.14% Submissions: 497K+ Points: 8 Average Time: 20m

Given an array `arr[]` with non-negative integers representing the height of blocks. If the width of each block is 1, compute how much water can be trapped between the blocks during the rainy season.

Examples:

Input: arr[] = [3, 0, 1, 0, 4, 0 2]
Output: 10
Explanation: Total water trapped = $0 + 3 + 2 + 3 + 0 + 2 + 0 = 10$ units.

Elevation Map



Building Water

```
Java (21) Start Timer
1 class Solution {
2     public int maxWater(int arr[]) {
3         // code here
4         int n = arr.length;
5
6         int[] leftMax = new int[n];
7         int[] rightMax = new int[n];
8
9         leftMax[0] = arr[0];
10        for (int i = 1; i < n; i++) {
11            leftMax[i] = Math.max(leftMax[i - 1], arr[i]);
12        }
13
14        rightMax[n - 1] = arr[n - 1];
15        for (int i = n - 2; i >= 0; i--) {
16            rightMax[i] = Math.max(rightMax[i + 1], arr[i]);
17        }
18
19        int water = 0;
20
21        for (int i = 0; i < n; i++) {
22            water += Math.min(leftMax[i], rightMax[i]) - arr[i];
23        }
24
25    }
26
27
28 }
```

Custom Input Compile & Run Submit

Factorials of large numbers | X Merge Without Extra Space | X Common in 3 Sorted Arrays | X Array Subset | Practice | GeekyXenon | X Triplet Sum in Array | Practice X Trapping Rain Water | Practice X + - X

https://www.geeksforgeeks.org/problems/trapping-rain-water-1587115621/1

Get 90% Refund Courses Tutorials Practice Jobs

Search... Problem Editorial Submissions Comments Output Window

Java (21) Start Timer

```
1 class Solution {  
2     public int maxWater(int arr[]) {  
3         // code here  
4         int n = arr.length;  
5  
6         int[] leftMax = new int[n];  
7         int[] rightMax = new int[n];  
8  
9         leftMax[0] = arr[0];  
10        for (int i = 1; i < n; i++) {  
11            leftMax[i] = Math.max(leftMax[i - 1], arr[i]);  
12        }  
13  
14        rightMax[n - 1] = arr[n - 1];  
15        for (int i = n - 2; i >= 0; i--) {  
16            rightMax[i] = Math.max(rightMax[i + 1], arr[i]);  
17        }  
18  
19        int water = 0;  
20  
21        for (int i = 0; i < n; i++) {  
22            water += Math.min(leftMax[i], rightMax[i]) - arr[i];  
23        }  
24  
25    }  
26  
27 }  
28 }
```

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 Time Taken 0.26

Your Total Score: 46 ↑

Solve Next

Longest Arithmetic Subsequence Rod Cutting Jump Game

Stay Ahead With:

Custom Input Compile & Run Submit Ctrl + Enter

15°C Mostly cloudy

Search

2037 03-02-2026