WORKSHEET 5

Student Name: Arfan **UID: 22BCS12329**

Branch: CSE **Section/Group: NTPP 603B**

Semester: 6 **Date of Performance: 21/02/25**

Subject Name: AP Lab 2 **Subject Code: 22CSP-351**

1. **Aim**:
   a. Merge Sorted Array
   b. First Bad Version
   c. Kth Largest Element in an Array\
   d. Search a 2D Matrix II

2. **Source Code:**

a.
```
class Solution {

public void merge(int[] nums1, int m, int[] nums2, int n) {
    int i = m - 1;  // Pointer for the last valid element in nums1
    int j = n - 1;  // Pointer for the last element in nums2
    int k = m + n - 1;  // Pointer for the last position in nums1

    // Merge nums1 and nums2 from the back
    while (i >= 0 && j >= 0) {
        if (nums1[i] > nums2[j]) {
            nums1[k] = nums1[i];
            i--;
        } else {
            nums1[k] = nums2[j];
            j--;
        }
        k--;
    }

    // If there are remaining elements in nums2, copy them to nums1
    while (j >= 0) {
        nums1[k] = nums2[j];
        j--;
```

```
            k--;
        }

    }
}
```

b.

```
/* The isBadVersion API is defined in the parent class VersionControl.
      boolean isBadVersion(int version); */

public class Solution extends VersionControl {
    public int firstBadVersion(int n) {
        int low = 1;
        int high = n;

        while (low < high) {
            int mid = low + (high - low) / 2;  // Avoid overflow
            if (isBadVersion(mid)) {
                high = mid;  // The first bad version is at mid or before it
            } else {
                low = mid + 1;  // The first bad version is after mid
            }
        }

        // At the end of the loop, low == high, pointing to the first bad version
        return low;
    }
}
```

c.

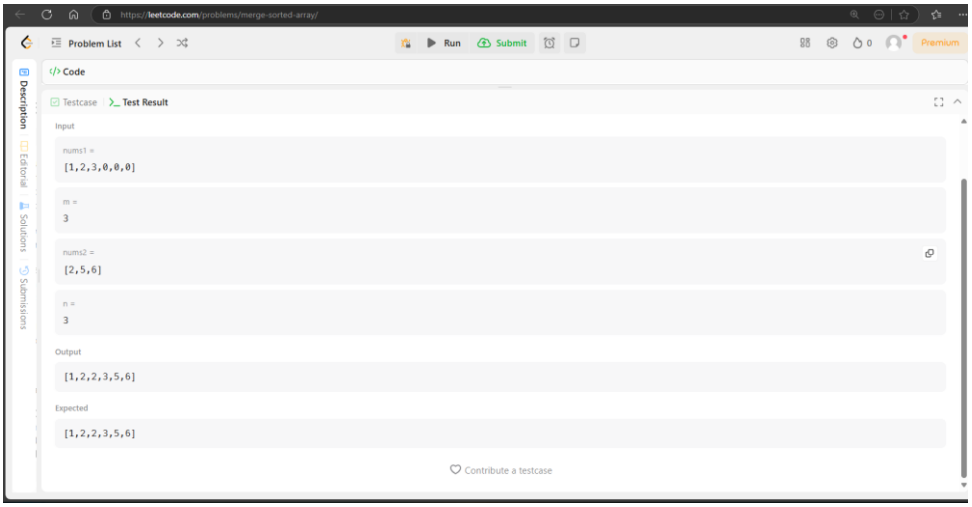
```
class Solution {

    public int findKthLargest(int[] nums, int k) {
        // Min-Heap to keep the top k elements
        PriorityQueue<Integer> minHeap = new PriorityQueue<>();

        // Process each element in the array
        for (int num : nums) {
            minHeap.offer(num); // Add the current element to the heap
            if (minHeap.size() > k) {
                minHeap.poll(); // Remove the smallest element to maintain size k
            }
        }

        // The root of the min-heap will be the kth largest element
        return minHeap.peek();
    }
}
```

d.

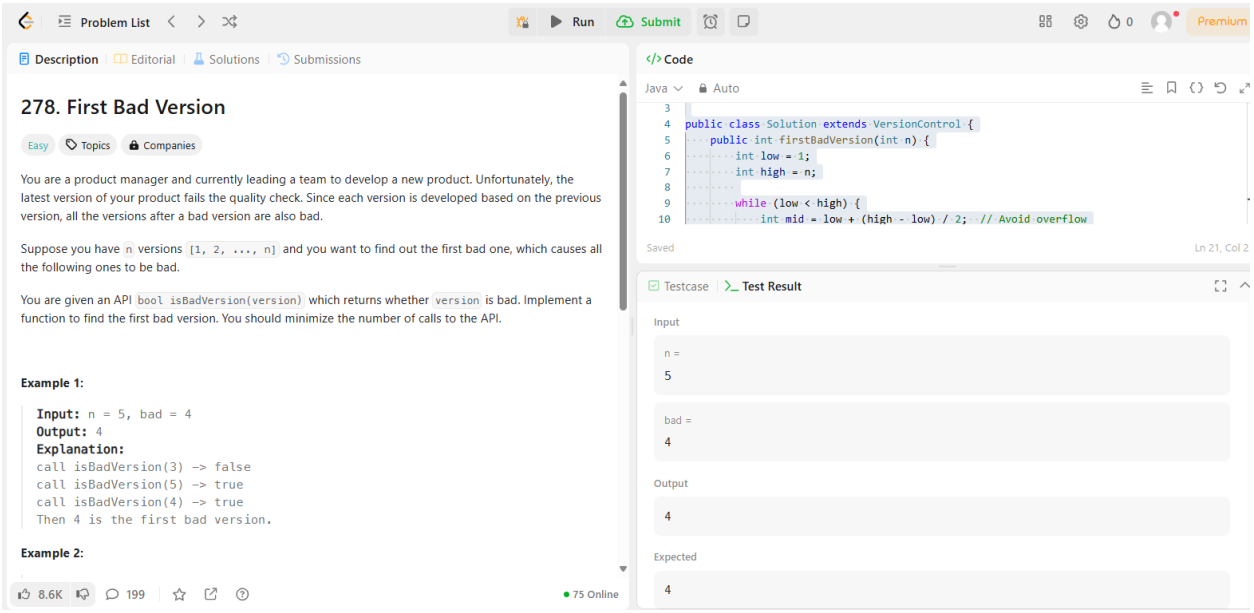
```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        if (matrix == null || matrix.length == 0 || matrix[0].length == 0) {
            return false;
        }

        int rows = matrix.length;
        int cols = matrix[0].length;

        int row = 0;          // Start at the top row
        int col = cols - 1;   // Start at the last column

        // Traverse the matrix
        while (row < rows && col >= 0) {
            if (matrix[row][col] == target) {
                return true;  // Target found
            } else if (matrix[row][col] > target) {
                col--;  // Move left
            } else {
                row++;  // Move down
            }
        }

        return false;  // Target not found
    }
}
```

3. Screenshot of Outputs:

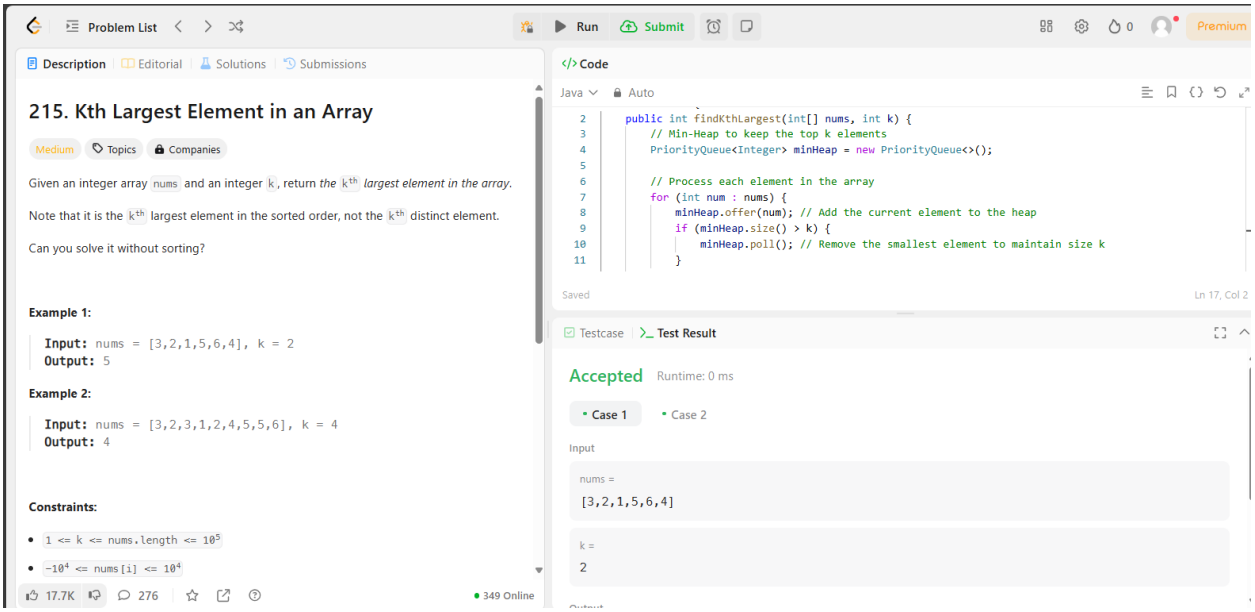
a.



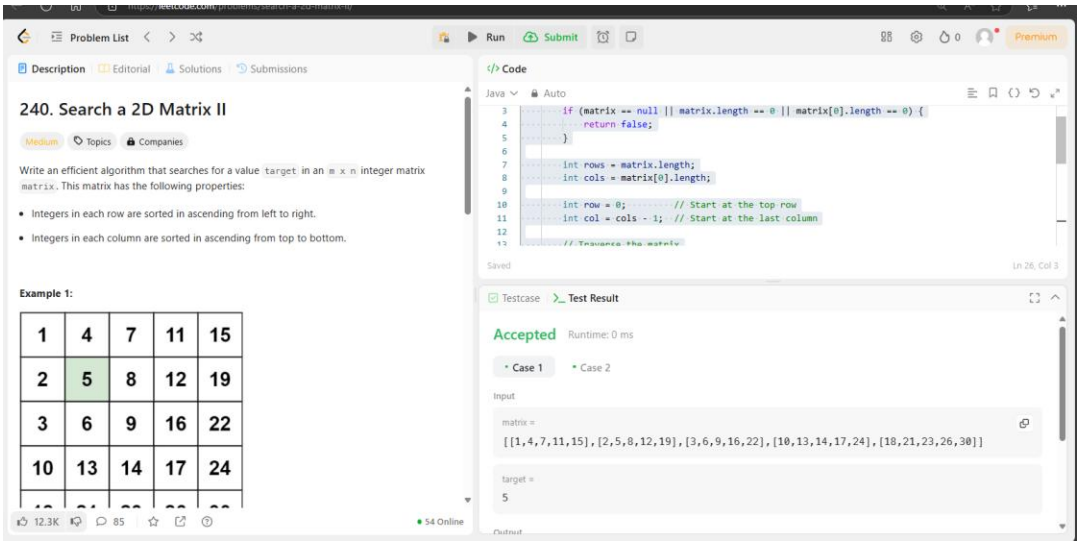
b.


Problem List
Run
Submit
Premium
Description
Editorial
Solutions
Submissions
278. First Bad Version
Easy
Topics
Companies
You are a product manager and currently leading a team to develop a new product. Unfortunately, the latest version of your product fails the quality check. Since each version is developed based on the previous version, all the versions after a bad version are also bad.
Suppose you have n versions [1, 2, ..., n] and you want to find out the first bad one, which causes all the following ones to be bad.
You are given an API bool isBadVersion(version) which returns whether version is bad. Implement a function to find the first bad version. You should minimize the number of calls to the API.
Example 1:
Input: n = 5, bad = 4
Output: 4
Explanation:
call isBadVersion(3) -> false
call isBadVersion(5) -> true
call isBadVersion(4) -> true
Then 4 is the first bad version.
Example 2:
8.6K
199
75 Online
Code
Java
Auto
public class Solution extends VersionControl {
public int firstBadVersion(int n) {
int low = 1;
int high = n;
while (low < high) {
int mid = low + (high - low) / 2; // Avoid overflow
Saved
Ln 21, Col 2
Testcase
Test Result
Input
n =
5
bad =
4
Output
4
Expected
4

c.


Problem List
Run
Submit
Premium
Description
Editorial
Solutions
Submissions
215. Kth Largest Element in an Array
Medium
Topics
Companies
Given an integer array nums and an integer k, return the kth largest element in the array.
Note that it is the kth largest element in the sorted order, not the kth distinct element.
Can you solve it without sorting?
Example 1:
Input: nums = [3,2,1,5,6,4], k = 2
Output: 5
Example 2:
Input: nums = [3,2,3,1,2,4,5,5,6], k = 4
Output: 4
Constraints:
1 <= k <= nums.length <= 10^5
-10^4 <= nums[i] <= 10^4
17.7K
276
349 Online
Code
Java
Auto
public int findKthLargest(int[] nums, int k) {
// Min-Heap to keep the top k elements
PriorityQueue<Integer> minHeap = new PriorityQueue<>();
// Process each element in the array
for (int num : nums) {
minHeap.offer(num); // Add the current element to the heap
if (minHeap.size() > k) {
minHeap.poll(); // Remove the smallest element to maintain size k
}
Saved
Ln 17, Col 2
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
nums =
[3,2,1,5,6,4]
k =
2

d.


Problem List
Run
Submit
Premium
Description
Editorial
Solutions
Submissions
240. Search a 2D Matrix II
Medium
Topics
Companies
Write an efficient algorithm that searches for a value target in an m x n integer matrix matrix. This matrix has the following properties:
Integers in each row are sorted in ascending from left to right.
Integers in each column are sorted in ascending from top to bottom.
Example 1:
1 4 7 11 15
2 5 8 12 19
3 6 9 16 22
10 13 14 17 24
Code
Java
Auto
if (matrix == null || matrix.length == 0 || matrix[0].length == 0) {
return false;
}
int rows = matrix.length;
int cols = matrix[0].length;
int row = 0; // Start at the top row
int col = cols - 1; // Start at the last column
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
matrix =
[[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]]
target =
5
12.3K
85
54 Online