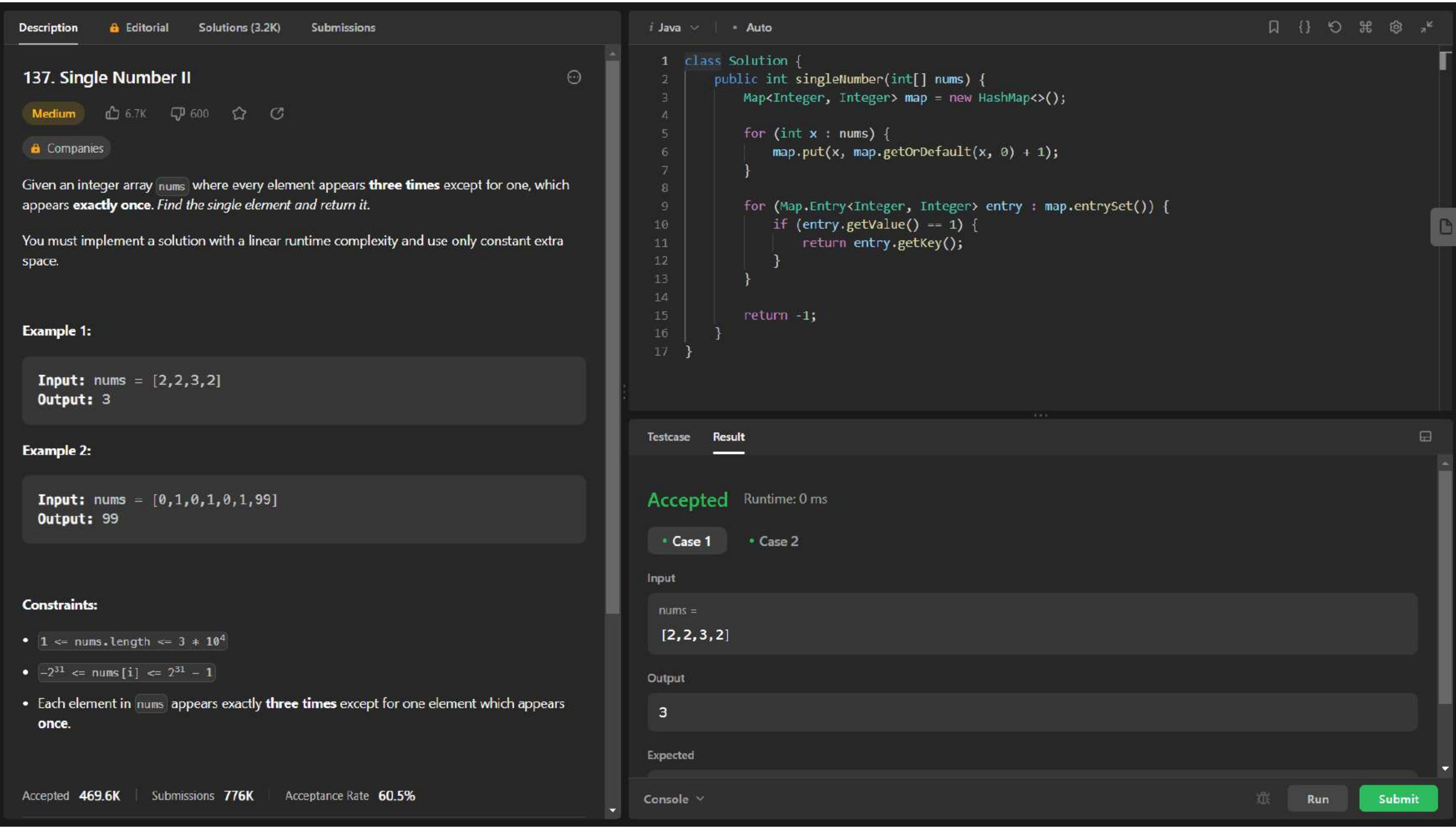


137. Single Number II

Medium 6.7K 600

Companies

Given an integer array `nums` where every element appears **three times** except for one, which appears **exactly once**. *Find the single element and return it.*

You must implement a solution with a linear runtime complexity and use only constant extra space.

Example 1:

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

Example 2:

```
Input: nums = [0,1,0,1,0,1,99]
Output: 99
```

Constraints:

- `1 <= nums.length <= 3 * 10^4`
- -2^{31} `<= nums[i] <=` $2^{31} - 1$
- Each element in `nums` appears exactly **three times** except for one element which appears **once**.

Accepted 469.6K | Submissions 776K | Acceptance Rate 60.5%

```java
class Solution {
    public int singleNumber(int[] nums) {
        Map<Integer, Integer> map = new HashMap<>();

        for (int x : nums) {
            map.put(x, map.getOrDefault(x, 0) + 1);
        }

        for (Map.Entry<Integer, Integer> entry : map.entrySet()) {
            if (entry.getValue() == 1) {
                return entry.getKey();
            }
        }

        return -1;
    }
}
```

Accepted Runtime: 0 ms

Case 1 Case 2

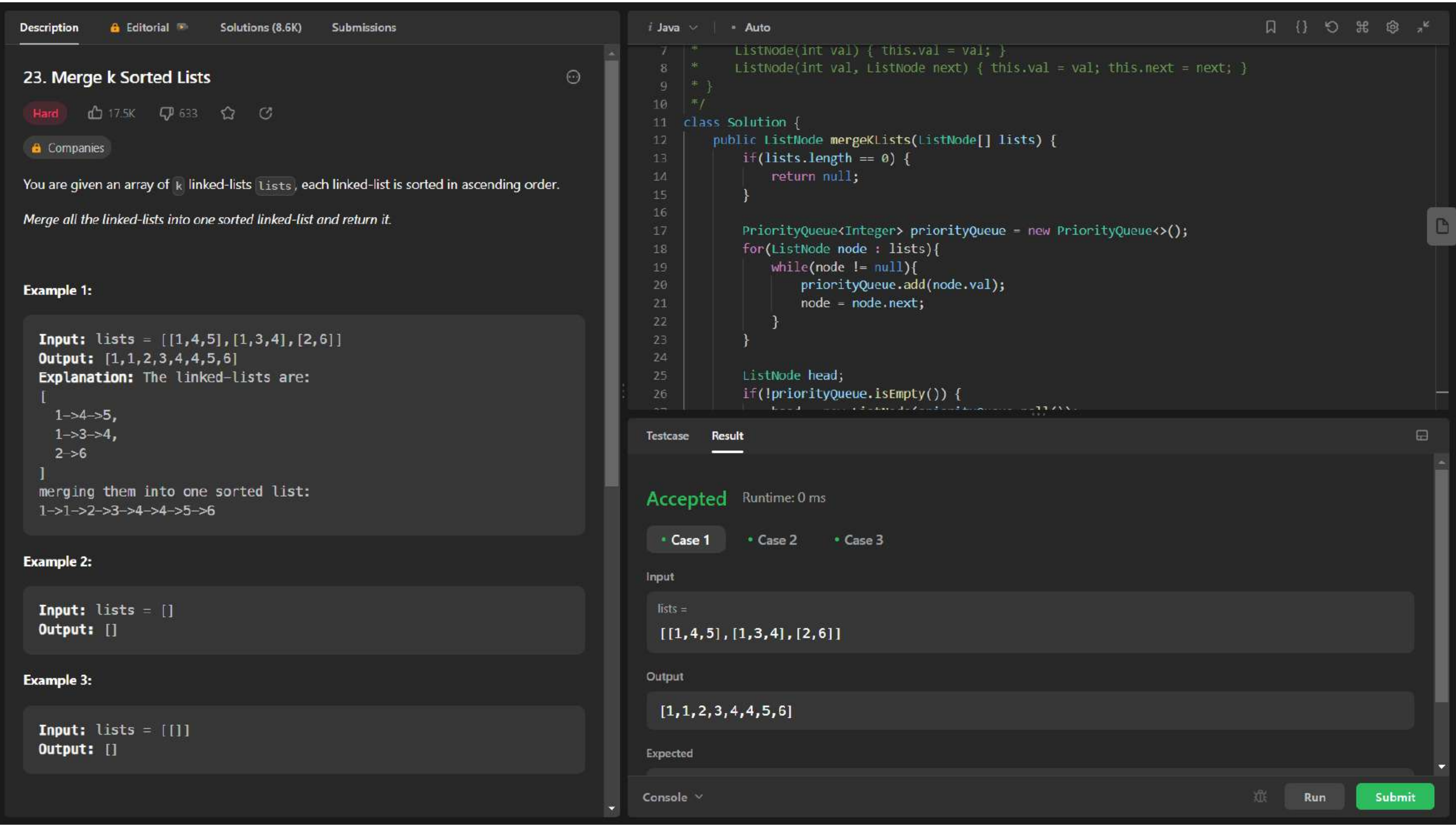
Input

```
nums =
[2,2,3,2]
```

Output

```
3
```

Expected



23. Merge k Sorted Lists

You are given an array of `k` linked-lists `lists`, each linked-list is sorted in ascending order.

Merge all the linked-lists into one sorted linked-list and return it.

Example 1:

```
Input: lists = [[1,4,5],[1,3,4],[2,6]]
Output: [1,1,2,3,4,4,5,6]
Explanation: The linked-lists are:
[
  1->4->5,
  1->3->4,
  2->6
]
merging them into one sorted list:
1->1->2->3->4->4->5->6
```

Example 2:

```
Input: lists = []
Output: []
```

Example 3:

```
Input: lists = [[]]
Output: []
```

```java
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode mergeKLists(ListNode[] lists) {
        if(lists.length == 0) {
            return null;
        }

        PriorityQueue<Integer> priorityQueue = new PriorityQueue<>();
        for(ListNode node : lists){
            while(node != null){
                priorityQueue.add(node.val);
                node = node.next;
            }
        }

        ListNode head;
        if(!priorityQueue.isEmpty()) {
```

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

lists =

```
[[1,4,5],[1,3,4],[2,6]]
```

Output

```
[1,1,2,3,4,4,5,6]
```

Expected