Description | Editorial | Solutions (1.8K) | Submissions

703. Kth Largest Element in a Stream

Easy | 4.4K | 2.6K

Companies

Design a class to find the k^{th} largest element in a stream. Note that it is the k^{th} largest element in the sorted order, not the k^{th} distinct element.

Implement `KthLargest` class:

- `KthLargest(int k, int[] nums)` Initializes the object with the integer `k` and the stream of integers `nums`.
- `int add(int val)` Appends the integer `val` to the stream and returns the element representing the k^{th} largest element in the stream.

Example 1:

Input
```
["KthLargest", "add", "add", "add", "add", "add"]
[[3, [4, 5, 8, 2]], [3], [5], [10], [9], [4]]
```
Output
```
[null, 4, 5, 5, 8, 8]
```

Explanation
```
KthLargest kthLargest = new KthLargest(3, [4, 5, 8, 2]);
kthLargest.add(3);   // return 4
kthLargest.add(5);   // return 5
kthLargest.add(10);  // return 5
kthLargest.add(9);   // return 8
kthLargest.add(4);   // return 8
```

Constraints:

Java | Auto

```java
class KthLargest {

    PriorityQueue<Integer> minHeap;
    int k;

    public KthLargest(int k, int[] nums) {
        this.minHeap = new PriorityQueue<>();
        this.k = k;
        for(int num: nums){
            minHeap.add(num);
            if(minHeap.size() > k) {
                minHeap.poll();
            }

        }
    }

    public int add(int val) {
        minHeap.add(val);
        if (minHeap.size() > k) {
```

Testcase | Result

Accepted Runtime: 2 ms

Case 1

Input

```
["KthLargest","add","add","add","add","add"]
```

```
[[3,[4,5,8,2]],[3],[5],[10],[9],[4]]
```

Output

```
[null,4,5,5,8,8]
```

Console | Run | Submit

Description | Editorial | Solutions (11.3K) | Submissions

4. Median of Two Sorted Arrays

Hard 23.5K 2.6K

Companies

Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, return **the median** of the two sorted arrays.

The overall run time complexity should be `O(log (m+n))`.

Example 1:

```
Input: nums1 = [1,3], nums2 = [2]
Output: 2.00000
Explanation: merged array = [1,2,3] and median is 2.
```

Example 2:

```
Input: nums1 = [1,2], nums2 = [3,4]
Output: 2.50000
Explanation: merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5.
```

Constraints:

- `nums1.length == m`
- `nums2.length == n`
- `0 <= m <= 1000`
- `0 <= n <= 1000`

Java | Auto

```
            a[k]=nums1[i];
            i++;
        }
        else{
            a[k]=nums2[j];
            j++;
        }
        k++;
    }
    while(i<m){
        a[k]=nums1[i];
        i++;
        k++;
    }
    while(j<n){
        a[k]=nums2[j];
        j++;
        k++;
    }
    if ((m+n)%2!=0){
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

nums1 =
[1,3]

nums2 =
[2]

Output

Console | Run | Submit