Problem 703: Kth Largest Element in a Stream

Problem Information

Difficulty: Easy
Acceptance Rate: 60.41%
Paid Only: No
Tags: Tree, Design, Binary Search Tree, Heap (Priority Queue), Binary Tree, Data Stream

Problem Description

You are part of a university admissions office and need to keep track of the `kth` highest test score from applicants in real-time. This helps to determine cut-off marks for interviews and admissions dynamically as new applicants submit their scores.

You are tasked to implement a class which, for a given integer `k`, maintains a stream of test scores and continuously returns the `k`th highest test score **after** a new score has been submitted. More specifically, we are looking for the `k`th highest score in the sorted list of all scores.

Implement the `KthLargest` class:

* `KthLargest(int k, int[] nums)` Initializes the object with the integer `k` and the stream of test scores `nums`. * `int add(int val)` Adds a new test score `val` to the stream and returns the element representing the `kth` largest element in the pool of test scores so far.

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

Example 2:

Input: ["KthLargest", "add", "add", "add", "add"] [[4, [7, 7, 7, 7, 8, 3]], [2], [10], [9], [9]]

Output: [null, 7, 7, 7, 8]

Explanation:

KthLargest kthLargest = new KthLargest(4, [7, 7, 7, 7, 8, 3]); kthLargest.add(2); // return 7 kthLargest.add(10); // return 7 kthLargest.add(9); // return 7 kthLargest.add(9); // return 8

Constraints:

* `0 <= nums.length <= 104` * `1 <= k <= nums.length + 1` * `-104 <= nums[i] <= 104` * `-104 <= val <= 104` * At most `104` calls will be made to `add`.

Code Snippets

C++:

```
class KthLargest {
public:
KthLargest(int k, vector<int>& nums) {

}

int add(int val) {

}
};

/**
* Your KthLargest object will be instantiated and called as such:
* KthLargest* obj = new KthLargest(k, nums);
* int param_1 = obj->add(val);
*/
```

Java:

```
class KthLargest {

public KthLargest(int k, int[] nums) {

}

public int add(int val) {

}
}

/**
* Your KthLargest object will be instantiated and called as such:
* KthLargest obj = new KthLargest(k, nums);
* int param_1 = obj.add(val);
*/
```

Python3:

```
class KthLargest:

def __init__(self, k: int, nums: List[int]):


def add(self, val: int) -> int:



# Your KthLargest object will be instantiated and called as such:
# obj = KthLargest(k, nums)
# param_1 = obj.add(val)
```