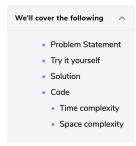


Grokking the Coding Interview: Patterns for **Coding Questions** 78% completed Q Search Course Pattern: In-place Reversal of a LinkedList Pattern: Tree Breadth First Search Pattern: Tree Depth First Search Pattern: Two Heaps Pattern: Subsets Pattern: Modified Binary Search Pattern: Bitwise XOR Pattern: Top 'K' Elements Introduction Top 'K' Numbers (easy) Kth Smallest Number (easy) 'K' Closest Points to the Origin (easy) Connect Ropes (easy) Top 'K' Frequent Numbers (medium) Frequency Sort (medium) Kth Largest Number in a Stream (medium) 'K' Closest Numbers (medium) Maximum Distinct Flements (medium) Sum of Elements (medium) Rearrange String (hard) Problem Challenge 1 Solution Review: Problem Challenge 1 Problem Challenge 2 Solution Review: Problem Challenge 2 Problem Challenge 3 Solution Review: Problem Challenge 3 Pattern: K-way merge Introduction Merge K Sorted Lists (medium) Kth Smallest Number in M Sorted Lists (Medium) Kth Smallest Number in a Sorted Matrix (Hard) Smallest Number Range (Hard) Problem Challenge 1 Solution Review: Problem

Challenge 1

Kth Largest Number in a Stream (medium)



Problem Statement

Design a class to efficiently find the Kth largest element in a stream of numbers.

The class should have the following two things:

- 1. The constructor of the class should accept an integer array containing initial numbers from the stream and an integer 'K'.
- 2. The class should expose a function add(int num) which will store the given number and return the Kth largest number.

Example 1:

```
Input: [3, 1, 5, 12, 2, 11], K = 4
1. Calling add(6) should return '5'.
2. Calling add(13) should return '6'.
2. Calling add(4) should still return '6'.
```

Try it yourself

Try solving this question here:

```
| class KthLargestNumberInStream {
| constructor(nums, k) {
| // TODO: Write your code here |
| this.k = k; |
| here |
| here
```

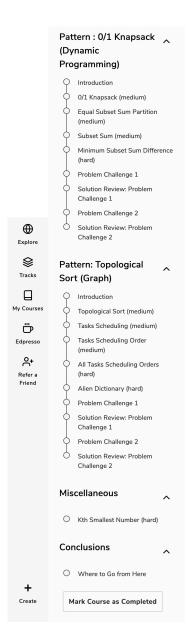
Solution

This problem follows the Top 'K' Elements pattern and shares similarities with Kth Smallest number.

We can follow the same approach as discussed in the 'Kth Smallest number' problem. However, we will use a **Min Heap** (instead of a **Max Heap**) as we need to find the Kth largest number.

Code

Here is what our algorithm will look like:



```
add(num) {
          this.minHeap.push(num);
          if (this.minHeap.length > this.k) {
           this.minHeap.pop();
          // return the 'Kth largest number
return this.minHeap.peek();
                                                                              SAVE
                                                                                        RESET
                                                                                                03
                                                                                            Close
                                                                                            4.134s
   4th largest number is: 5
   4th largest number is: 6
   4th largest number is: 6
Time complexity
The time complexity of the \  \, \operatorname{\sf add}() \, function will be O(logK) since we are inserting the new number in the
heap.
Space complexity
The space complexity will be O(K) for storing numbers in the heap.
 applying to them. See how ①
 ← Back
                                                                                            Next →
Frequency Sort (medium)
                                                                                 'K' Closest Numbers (medium)
                                                                                   ✓ Mark as Completed
```