

Grokking the Coding Interview: Patterns for Coding Questions

78% completed



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 Elements (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)

We'll cover the following

- Problem Statement
- Try it yourself
- Solution
- Code
 - Time complexity
 - Space complexity

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:

- The constructor of the class should accept an integer array containing initial numbers from the stream and an integer 'K'.
- 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:

JavaPython3JS C++

```
1 class KthLargestNumberInStream {
2   constructor(nums, k) {
3     // TODO: Write your code here
4     this.k = k;
5   }
6
7   add(num) {
8     // TODO: Write your code here
9     return -1;
10  }
11 };
12
13
14 kthLargestNumber = new KthLargestNumberInStream([3, 1, 5, 12, 2, 11], 4);
15 console.log(`4th largest number is: ${kthLargestNumber.add(6)}`);
16 console.log(`4th largest number is: ${kthLargestNumber.add(13)}`);
17 console.log(`4th largest number is: ${kthLargestNumber.add(4)}`);
18
```

RUNSAVERESET↺

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:

JavaPython3C++JS

```
1 const Heap = require('..collections/heap'); //http://www.collectionsjs.com
2
3
4 class KthLargestNumberInStream {
5   constructor(nums, k) {
6     this.minHeap = new Heap([], null, ((a, b) => b - a));
7     this.k = k;
8
9     // add the numbers in the min heap
10    nums.forEach((num) => {
11      this.add(num);
12    });
13  }
14
```

Explore

Tracks

My Courses

Edpresso

Refer a Friend

Create

Pattern : 0/1 Knapsack (Dynamic Programming)

Introduction

0/1 Knapsack (medium)

Equal Subset Sum Partition (medium)

Subset Sum (medium)

Minimum Subset Sum Difference (hard)

Problem Challenge 1

Solution Review: Problem Challenge 1

Problem Challenge 2

Solution Review: Problem Challenge 2

Pattern: Topological Sort (Graph)

Introduction

Topological Sort (medium)

Tasks Scheduling (medium)

Tasks Scheduling Order (medium)

All Tasks Scheduling Orders (hard)

Alien Dictionary (hard)

Problem Challenge 1

Solution Review: Problem Challenge 1

Problem Challenge 2

Solution Review: Problem Challenge 2

Miscellaneous

Kth Smallest Number (hard)

Conclusions

Where to Go from Here

Mark Course as Completed

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

```
add(num) {  
  // add the new number in the min heap  
  this.minHeap.push(num);  
  
  // if heap has more than 'k' numbers, remove one number  
  if (this.minHeap.length > this.k) {  
    this.minHeap.pop();  
  }  
  
  // return the 'Kth largest number  
  return this.minHeap.peek();  
}
```

RUN

SAVE

RESET

Close

Output4.134s

4th largest number is: 5
4th largest number is: 6
4th largest number is: 6

Time complexity

The time complexity of the `add()` function will be $O(\log K)$ 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.

Interviewing soon? We've partnered with Hired so that companies apply to you instead of you applying to them. [See how](#)

Back

Next

Frequency Sort (medium)

'K' Closest Numbers (medium)

Mark as Completed

Report an Issue

Ask a Question