Kth Largest Number in a Stream (medium)

We'll cover the following

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:

Java | Python3 | JS | C++

```
class KthLargestNumberInStream:
  def __init__(self, nums, k):
    # TODO: Write your code here
    self.k = k

  def add(self, num):
    # TODO: Write your code here
    return -1


def main():
  kthLargestNumber = KthLargestNumberInStream([3, 1, 5, 12, 2, 11], 4)
  print("4th largest number is: " + str(kthLargestNumber.add(6)))
  print("4th largest number is: " + str(kthLargestNumber.add(13)))
  print("4th largest number is: " + str(kthLargestNumber.add(4)))


main()


```

Solution

This problem follows the **Top 'K' Elements** pattern and shares similarities with Kth Smallest number (https://www.educative.io/collection/page/5668639101419520/5671464854355968/5696381570252800/).

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:

Java | Python3 | C++ | JS

```
from heapq import *


class KthLargestNumberInStream:
  minHeap = []

  def __init__(self, nums, k):
    self.k = k
    # add the numbers in the min heap
    for num in nums:
      self.add(num)

  def add(self, num):
    # add the new number in the min heap
    heappush(self.minHeap, num)

    # if heap has more than 'k' numbers, remove one number
    if len(self.minHeap) > self.k:
      heappop(self.minHeap)

    # return the 'Kth largest number
    return self.minHeap[0]


def main():

  kthLargestNumber = KthLargestNumberInStream([3, 1, 5, 12, 2, 11], 4)
  print("4th largest number is: " + str(kthLargestNumber.add(6)))
```

Time complexity

The time complexity of the `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.

Back

Frequency Sort (medium)

Next

'K' Closest Numbers (medium)