Kth Smallest Number in M Sorted Lists (Medium)

Problem Statement

Given 'M' sorted arrays, find the K'th smallest number among all the arrays.

Example 1:

```
Input: L1=[2, 6, 8], L2=[3, 6, 7], L3=[1, 3, 4], K=5
Output: 4
Explanation: The 5th smallest number among all the arrays is 4, this can be verified from the m
erged
list of all the arrays: [1, 2, 3, 3, 4, 6, 6, 7, 8]
```

Example 2:

```
Input: L1=[5, 8, 9], L2=[1, 7], K=3
Output: 7
Explanation: The 3rd smallest number among all the arrays is 7.
```

Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```
const find_Kth_smallest = function(lists, k) {
  number = -1;
  // TODO: Write your code here
  return number;
};


console.log(`Kth smallest number is: ${find_Kth_smallest([[2, 6, 8], [3, 6, 7], [1, 3, 4]], 5)}`)

```

RUN SAVE RESET

Solution

This problem follows the **K-way merge** pattern and we can follow a similar approach as discussed in Merge K Sorted Lists.

We can start merging all the arrays, but instead of inserting numbers into a merged list, we will keep count to see how many elements have been inserted in the merged list. Once that count is equal to 'K', we have found our required number.

A big difference from Merge K Sorted Lists is that in this problem, the input is a list of arrays compared to LinkedLists. This means that when we want to push the next number in the heap we need to know what the index of the current number in the current array was. To handle this, we will need to keep track of the array and the element indices.

Code

Here is what our algorithm will look like:

Java | Python3 | C++ | JS

```
const Heap = require('./collections/heap'); //http://www.collectionsjs.com

function find_Kth_smallest(lists, k) {
  minHeap = new Heap([], null, ((a, b) => a[0] < b[0]));

  // put the 1st element of each list in the min heap
  for (let i = 0; i < lists.length; i++) {
    minHeap.push([lists[i][0], 0, lists[i]]);
  }

  // take the smallest(i.e., top) element form the min heap, if the running count is equal to k return th
  let numberCount = 0,
    number = 0;
  while (minHeap.length > 0) {
```

```
15    [number, i, list] = minHeap.pop();
16    numberCount += 1;
17    if (numberCount === k) {
18      break;
19    }
20    // if the array of the top element has more elements, add the next element to the heap
21    if (list.length > i + 1) {
22      minHeap.push([list[i + 1], i + 1, list]);
23    }
24  }
25  return number;
26 }
27
28
```

RUN SAVE RESET

Close

Output 4.461s

```
Kth smallest number is: 6
```

Time complexity

Since we'll be going through at most 'K' elements among all the arrays, and we will remove/add one element in the heap in each step, the time complexity of the above algorithm will be $O(K * logM)$ where 'M' is the total number of input arrays.

Space complexity

The space complexity will be $O(M)$ because, at any time, our min-heap will be storing one number from all the 'M' input arrays.

Similar Problems

Problem 1: Given 'M' sorted arrays, find the median number among all arrays.

Solution: This problem is similar to our parent problem with K=Median. So if there are 'N' total numbers in all the arrays we need to find the K'th minimum number where $K = N/2$.

Problem 2: Given a list of 'K' sorted arrays, merge them into one sorted list.

Solution: This problem is similar to Merge K Sorted Lists except that the input is a list of arrays compared to **LinkedLists**. To handle this, we can use a similar approach as discussed in our parent problem by keeping a track of the array and the element indices.

Back
Merge K Sorted Lists (medium)

Next
Kth Smallest Number in a Sorted Matri...

Mark as Completed