Kth Smallest Number in M Sorted Lists (Medium)

We'll cover the following ∧

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

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 me
rged
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++ |

```python
1   def find_Kth_smallest(lists, k):
2       number = -1
3       # TODO: Write your code here
4       return number
5
6
7   def main():
8       print("Kth smallest number is: " +
9             str(find_Kth_smallest([[2, 6, 8], [3, 6, 7], [1, 3, 4]], 5)))
10
11
12  main()
13
```

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** JS

```python
from heapq import *


def find_Kth_smallest(lists, k):
    minHeap = []

    # put the 1st element of each list in the min heap
    for i in range(len(lists)):
        heappush(minHeap, (lists[i][0], 0, lists[i]))

    # take the smallest(top) element form the min heap, if the running count is equal to k
    #return the number
    numberCount, number = 0, 0
    while minHeap:
        number, i, list = heappop(minHeap)
        numberCount += 1
        if numberCount == k:
            break
        # if the array of the top element has more elements, add the next element to the heap
        if len(list) > i+1:
            heappush(minHeap, (list[i+1], i+1, list))

    return number


def main():
    print("Kth smallest number is: " +
          str(find_Kth_smallest([[2, 6, 8], [3, 6, 7], [1, 3, 4]], 5)))


main()
```

Run   Save   Reset   ⌞⌝

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

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.

① Report an Issue   ？ Ask a Question