Smallest Number Range (Hard)

We'll cover the following

Problem Statement

Given 'M' sorted arrays, find the smallest range that includes at least one number from each of the 'M' lists.

Example 1:

```
Input: L1=[1, 5, 8], L2=[4, 12], L3=[7, 8, 10]
Output: [4, 7]
Explanation: The range [4, 7] includes 5 from L1, 4 from L2 and 7 from L3.
```

Example 2:

```
Input: L1=[1, 9], L2=[4, 12], L3=[7, 10, 16]
Output: [9, 12]
Explanation: The range [9, 12] includes 9 from L1, 12 from L2 and 10 from L3.
```

Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```python
def find_smallest_range(lists):
  # TODO: Write your code here
  return [-1, -1]


def main():
  print("Smallest range is: " +
        str(find_smallest_range([[1, 5, 8], [4, 12], [7, 8, 10]])))


main()
```

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 by inserting the first number from all the arrays in a min-heap. We will keep track of the largest number that we have inserted in the heap (let's call it `currentMaxNumber`).

In a loop, we'll take the smallest (top) element from the min-heap and `currentMaxNumber` has the largest element that we inserted in the heap. If these two numbers give us a smaller range, we'll update our range. Finally, if the array of the top element has more elements, we'll insert the next element to the heap.

We can finish searching the minimum range as soon as an array is completed or, in other terms, the heap has less than 'M' elements.

Code

Here is what our algorithm will look like:

Java | Python3 | C++ | JS

```python
from heapq import *
import math


def find_smallest_range(lists):
  minHeap = []
  rangeStart, rangeEnd = 0, math.inf
  currentMaxNumber = -math.inf

  # put the 1st element of each array in the max heap
  for arr in lists:
    heappush(minHeap, (arr[0], 0, arr))
    currentMaxNumber = max(currentMaxNumber, arr[0])

  # take the smallest(top) element form the min heap, if it gives us smaller range, update the ranges
  # if the array of the top element has more elements, insert the next element in the heap
  while len(minHeap) == len(lists):
    num, i, arr = heappop(minHeap)
    if rangeEnd - rangeStart > currentMaxNumber - num:
      rangeStart = num
      rangeEnd = currentMaxNumber

    if len(arr) > i+1:
      # insert the next element in the heap
      heappush(minHeap, (arr[i+1], i+1, arr))
      currentMaxNumber = max(currentMaxNumber, arr[i+1])

  return [rangeStart, rangeEnd]


def main():
  print("Smallest range is: " +
        str(find_smallest_range([[1, 5, 8], [4, 12], [7, 8, 10]])))


main()

```

Run | Save | Reset

Time complexity

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

Space complexity

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