Name: Imadh Ajaz Bandey
UTA ID: 1002228309
Hands-On 4

Problem : Fibonacci:

Call stack:

fib (5) → fib (4) → fib (3) → fib (2) → fib (1) → fib (0)
→ fib (1)
→ fib (2) → fib (1) → fib (0)
→ fib (3) → fib (2) → fib (1) → fib (0)
→ fib (1)

→ Problem 1:

Q2.) Time Complexity

Time Complexity for number of levels is

$O(\log M)$

For the merging of array is $O(M)$

Total time complexity $= O(\log M) \cdot O(M)$

$= O(M \log M)$

Consider all rows and columns

$T(K, N) = O((K * N) * \log (K * N))$

Q3) We can improve the algorithm by implementing min-heap with optimized priority queues and libraries like heapq. This allows to store only necessary data and minimize memory overhead. Multi-threading can also be used as one of the alternatives.

→ Problem 2:

Q2.) Time complexity:

If array is sorted:

O(N) → for input

O(N) → for removing duplicates

Combined : O(N)

If array is unsorted:

O(N) → for input

O(N log N) → for sorting

$O(N) \rightarrow$ for removing duplicates

$\therefore$ Combined : $O(N \log N)$

Q3.) Approach is already efficient. If the array is sorted, it takes time complexity of $O(N)$.