Kick Start 2022 - Coding Practice with Kick Start Session #1

Analysis: H-index

View problem and solution walkthrough video

Test set 1

For a set of papers, let us define a function $score(x)$, which denotes the number of papers in the set with citations at least x. We can calculate the score for every integer $x \leq n$, where n is the number of papers in the set. For each set of papers, the answer is the maximum x such that $score(x) \geq x$. We can store the counts of each of the citation numbers in the input in an array, and calculate cumulative sum to find the score for any x.

We need to find the maximum x where $score(x) \geq x$, each time a new paper is added. We can find that using the approach mentioned above in $O(max(\mathbf{A}))$ time, where $max(\mathbf{A})$ is the maximum citation number in the set of papers. An observation here is that we can consider all citations $\geq \mathbf{N}$ as $\mathbf{N}$, which allows us to find H-index for any set of papers in $O(\mathbf{N})$. This leads to a total time complexity of $O(\mathbf{N}^2)$ per test case, which is sufficient for test set 1.

Test set 2

For test set 2, we initialize the H-index with 0, and after adding each paper, we need to update the current H-index. We can use a minimum priority queue data structure to store the citation numbers. As we go through each of the papers, we keep updating the current H-index. We also keep updating the priority queue so that it only contains numbers greater than the current H-index and remove the rest. For each new paper, we can follow these steps:

1. If the current citation number is bigger than the current answer, add it in the priority queue
2. Remove all the citation numbers from the priority queue which are not greater than the current answer
3. If the size of priority queue is not less than the current answer + 1, increment the current answer by 1 and return to step 2

A single operation of priority queue takes $O(\log \mathbf{N})$ time, and each number is added and removed at most once. This leads to a total time complexity of $O(\mathbf{N} \log \mathbf{N})$ per test case, which is sufficient for test set 2.

A solution with linear time complexity is also possible, if we store the citation numbers in an array or a hashtable to calculate the answer. This is left as an exercise for the readers.