Documentation

The h-index problem requires finding the highest integer (h) for which a researcher has published at least (h) papers, each cited at least (h) times. In other words, the h-index measures productivity and citation impact by identifying the maximum number of documents with a citation count that meets or exceeds a threshold. Given an array of integers where each element represents the number of citations for a particular paper, our task is to determine the researcher's h-index based on this criterion. The h-index is widely used in academia to evaluate a researcher's influence, making this a meaningful metric for assessing academic contributions.

To solve the h-index problem, we start by sorting the citation list in descending order. Sorting the list helps arrange the papers in a way that allows us to evaluate the potential h-index by examining each paper sequentially from the most cited to the least cited. Once sorted, it becomes easier to find the highest possible h-index by simply counting how many papers meet the citation threshold of (h). This approach leverages the ordered list to check the citation requirements straightforwardly, thus simplifying the logic required to find the h-index.

After sorting, we iterate through each paper's citation count to see if the count is at least as high as its position in the list (1-based index). For example, if the citation count of the first paper (most cited) is greater than or equal to one, then it satisfies the condition for an h-index of one. If the second paper's citation count is greater than or equal to two, it satisfies the condition for an h-index of two, and so on. We continue this process until we encounter a paper where the citation count is less than its position in the sorted list, indicating that no further h-index values can be valid beyond this point. This incremental approach helps us determine the maximum h-index effectively.

The solution benefits from the efficiency of breaking the iteration as soon as the h-index condition fails. Once we identify the largest value of (h) that satisfies the h-index criteria, we have reached the researcher's maximum h-index. This approach ensures that we avoid unnecessary calculations, as the descending order enables early termination when no further papers can meet the citation threshold. Therefore, this solution is optimal, especially for cases where the number of papers is large, as it minimizes the number of comparisons needed to compute the h-index.

The complexity of this solution is primarily due to the sorting step, which requires (O(n log n)) time, where (n) is the number of papers. Iterating through the sorted list to evaluate the h-index has a linear time complexity of (O(n)), making the total time complexity dominated by the sorting operation. Space complexity is minimal, typically (O(1)) if the sort operation is performed in place, which makes this solution efficient in terms of both time and space. This balance between sorting and linear iteration provides a practical and performant method for calculating the h-index.

Overall, the h-index problem is solved by leveraging sorted order, which allows for an intuitive and efficient way to count papers that meet or exceed their respective citation thresholds. The method reflects a common approach to problems that require threshold-based counting, where ordering the data simplifies the logic required to find an optimal solution. This solution not only meets the problem's requirements but also serves as a useful template for similar problems that involve finding thresholds within ordered datasets.