Introduction:

The Longest Increasing Subsequence (LIS) problem involves finding the longest subsequence of a given sequence that is in increasing order. In this write-up, we will discuss an optimized solution to find the LIS using the LIS algorithm with binary search. This algorithm reduces the time complexity to O(n log n), where n is the length of the input array.

Algorithm:

1. Initialize an array 'tails' to store the smallest tail element of all increasing subsequences found so far.

2. Initialize an array 'prevIndices' to store the index of the previous element in the LIS for each element.

3. Initialize a variable 'length' to track the length of the longest increasing subsequence.

4. Iterate through each element in the input array:

   a. Perform a binary search on the 'tails' array to find the index to update or insert the current number.

   b. If the index is equal to the current 'length', update 'tails[length]' with the current index and increment 'length'.

   c. Otherwise, update 'tails[index]' with the current index.

   d. Set 'prevIndices[currentIndex]' to 'tails[index-1]' if 'index' is greater than 0, otherwise set it to -1.

5. Reconstruct the longest increasing subsequence:

   a. Initialize an empty list 'longestIncreasingSubsequence'.

   b. Set 'index' to 'tails[length - 1]'.

   c. While 'index' is not equal to -1, do the following:

      i. Add the element at 'index' from the input array to the beginning of 'longestIncreasingSubsequence'.

      ii. Update 'index' to 'prevIndices[index]'.

6. Return the 'longestIncreasingSubsequence' as the result.