**Details about Time Complexity**

**Longest Increasing Subsequence (LIC)**

The time complexity of the longest increasing subsequence (LIS) algorithm implemented using dynamic programming can be analyzed as follows:

**Best Case**:

* + In the best-case scenario, the input array is already sorted in increasing order, or it consists of all identical elements.
  + In this case, when processing each element **i**, the inner loop will not execute (since there are no previous elements to compare with).
  + Therefore, the time complexity would be **O(n)**, where **n** is the number of elements in the input array.

**Worst Case**:

* + In the worst-case scenario, the input array is in strictly decreasing order.
  + This means that for each element **i**, the inner loop will iterate over all previous elements **j** (0 ≤ j < i) to find the longest increasing subsequence ending at element **i**.
  + As a result, the time complexity would be **O(n^2)**, where **n** is the number of elements in the input array.

**Average Case**:

* + The average-case time complexity is also **O(n^2)**.
  + This is because, on average, we expect to perform approximately **n/2** comparisons for each element.
  + Considering that there are **n** elements in the array, the overall time complexity remains **O(n^2)**.

In summary, the LIS algorithm has a quadratic time complexity in both the worst case and the average case, making it suitable for relatively small input sizes.