Question 2: Worst-case complexity Analysis

The complexity of the 'merge_sort' function provided is O(n log n) in the worst case. This is due to two main operations in the code. One is 'merge_sort' that splits the array into halves, and the conquer step, which is the 'merge' operation that combines two sorted subarrays into a single sorted array.

The array is recursively split into two halves until it is no longer possible to divide. This splitting process happens log n times because with each level of recursion the array is halved, which is a characteristic of logarithmic behavior. The merge function takes two sorted subarrays and merges them into a single sorted array. This function runs in O(n) time for each merge because it goes through all elements of the subarrays once. Since each level of recursion results in a merge, and there are log n levels of recursion, the merging operations collectively take O(n log n) time. Since both the divide and conquer steps are nested (the conquer step occurs after the divide step at each level of recursion), the total time complexity of the algorithm is O(n log n).

Question 3: Manual Application of the Algorithm

Given the array [8, 42, 25, 3, 3, 2, 27, 3], the manual application of the 'merge_sort' algorithm involves recursively dividing the array and then merging the sorted subarrays. The steps would look like this:

- Initially, the array is [8, 42, 25, 3, 3, 2, 27, 3].
- It is divided into [8, 42, 25, 3] and [3, 2, 27, 3].
- These are further divided into [8, 42], [25, 3], [3, 2], [27, 3].
- Then, into [8], [42], [25], [3], [3], [2], [27], [3].
- During merging, we get [8, 42], [3, 25], [2, 3], [3, 27].
- Further merging results in [3, 8, 25, 42] and [2, 3, 3, 27].
- Finally, the entire array is merged to [2, 3, 3, 3, 8, 25, 27, 42].

Question 4: Consistency with Complexity Analysis

The number of steps involved in the manual application of the algorithm is consistent with the complexity analysis:

- For an array of size 8, the divide step should occur log2(8) = 3 times. This is seen as the array is split into halves three times until single elements are left.
- For each of the log n levels of division, a merge operation is performed which requires linear time. We can see this as the subarrays are merged back together.
- The actual number of steps will depend on the specific implementation details and the initial order of elements. However, the fundamental operations that dictate the complexity (splitting and merging) are consistently reflected in the manual application steps.