2. The merge() function has a time complexity of O(n). The function has linear complexity because it iterates over the elements once. The recursive calls to merge_sort() constantly divide the array making its complexity O(log n). This makes the overall algorithm's complexity O(nlogn)

3. Initial array: [8, 42, 25, 3, 3, 2, 27, 3]

Divide phase: The array gets divided after each call of merge_sort()

- [8, 42, 25, 3] [3, 2, 27, 3]
- [8, 42] [25, 3] [3, 2] [27, 3]
- [8] [42] [25] [3] [3] [2] [27] [3]

Merge phase: Merging of sub-arrays

- [8] and [42]: [8, 42]
- [25] and [3]: [3, 25]
- [3] and [2]: [2, 3]
- [27] and [3]: [3, 27]
- [8, 42] and [3, 25]: [3, 8, 25, 42]
- [3, 2] and [27, 3]: [2, 3, 3, 27]
- [3, 8, 25, 42] and [2, 3, 3, 27]: **[2, 3, 3, 3, 8, 25, 27, 42]** ← **Final sorted array**

4. The number of steps in the Divide phase (caused by calls to mergesort()) are 3. It matches the complexity analysis of O(logn). $Log_2 8 = 3$. The merge phase has O(n) complexity so it should have 8 steps. However, there are 7 steps because the 8th step already gives us a sorted array.