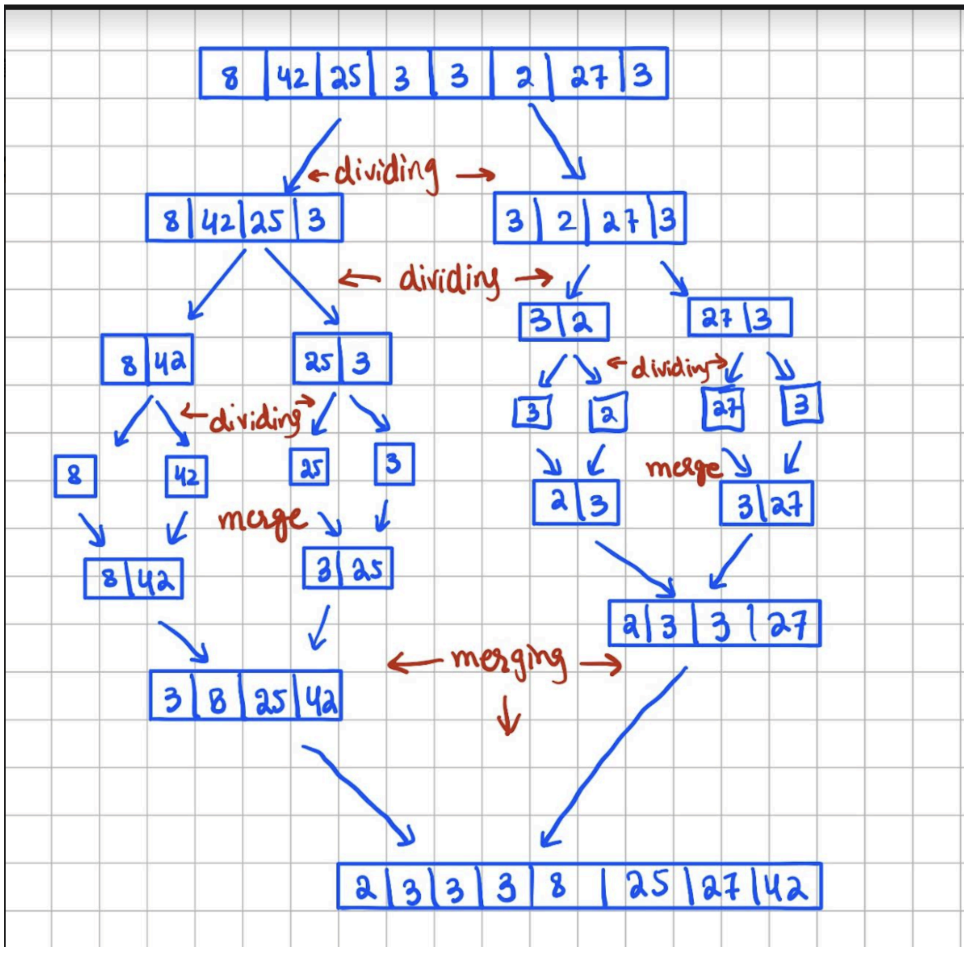
Exercise 1:

2) The array is divided recursively into halves in the merge sort function until each subarray has only one element. This division process takes O(log n) time for n number of elements in the array. The merge function combines all the sorted subarrays which takes O(n) time. Thus, combining the division and merging steps the overall worst-case time complexity of the merge sort algorithm is O(n log n).

Source:
https://softwareengineering.stackexchange.com/questions/297160/why-is-mergesort-olog-n

3)



In the visual representation it shows how we are splitting the array into halves at each step until there is only one element is left in the array and then after that, we start merging subarrays back together in a sorted way. At the end, we get the sorted array.

4) Yes, the time complexity is the same as calculated in part two because for each recursion call the array is split into halves until there is only one element left, and the merging operation for each step takes linear time to merge the sorted subarrays. Thus, the complexity analysis and the number of steps the merge sort method takes consistently confirm the predicted O(n log n) time complexity.