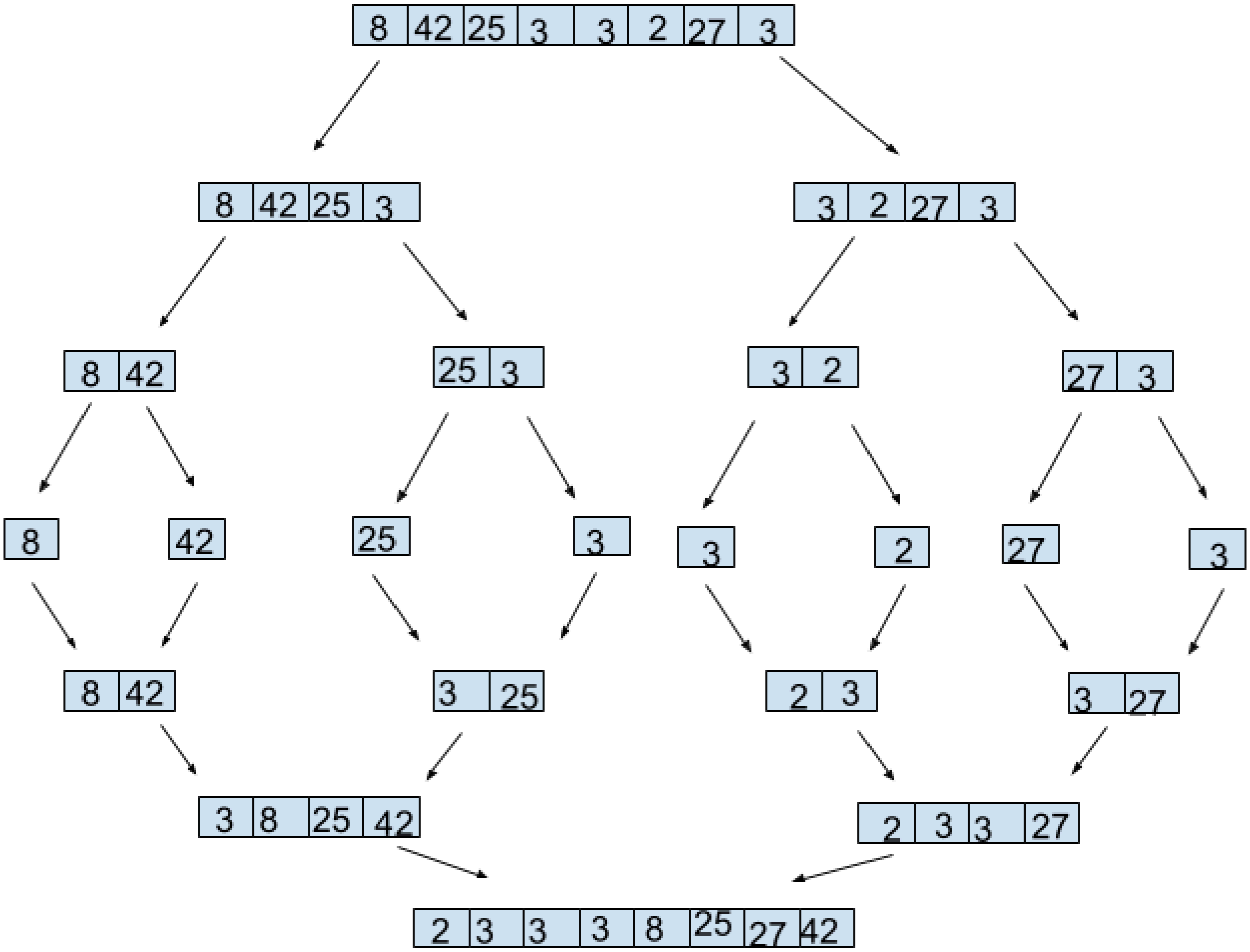
Exercise 1.2

The worst-case time complexity of the algorithm is O(n log n) because the array is recursively divided in half until each subarray has 1 element, which gives a depth of recursion of log n levels. This is indicated by the recursive calls merge_sort(arr, low, mid) and merge_sort(arr, mid + 1, high) in the code. At each level of recursion, merge() is called. The merge() function runs a loop that touches every element in the subarray once. Therefore, merging two halves takes O(n) time as every element is compared and copied exactly once. So we multiple these together:
O(n) x O(log n) = O(n log n).

Exercise 1.3



Exercise 1.4

Yes, the number of steps is consistent with the complexity analysis. The array was split log(n) times, which was 3 times for 8 elements(1 set of 8 elements was split into 2 sets of 4, then 4 sets of 2 and then 8 sets of 1). At each level, every element was compared and copied once, leading to O(n) work per level. Thus, the overall complexity is consistent with O(n log n).