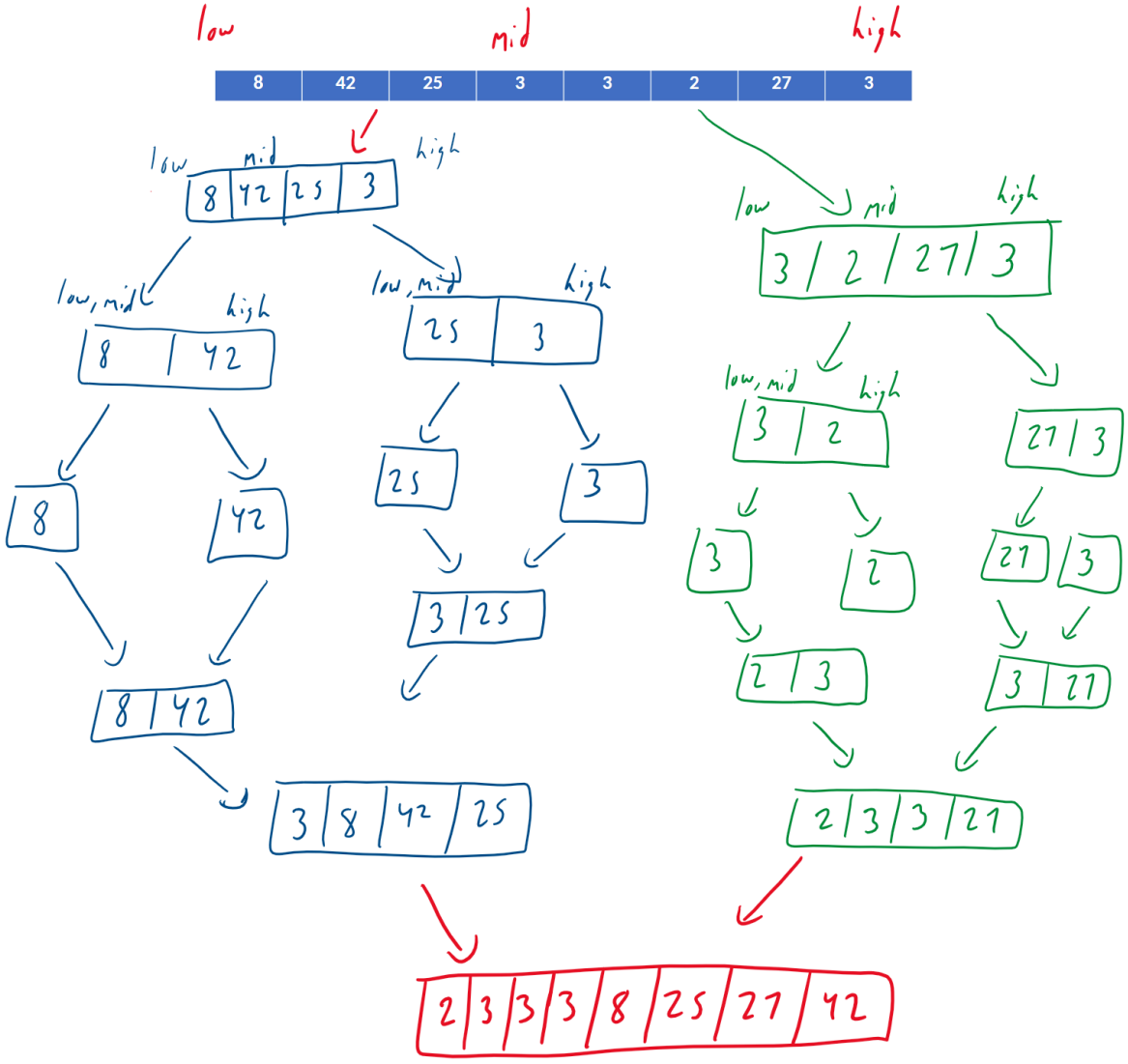
Exercise One:

Question 2: Argue that the overall algorithm has a worst-case complexity of O(nlogn). Note, your description must specifically refer to the code you wrote, i.e., not just generically talk about mergesort. [0.4 pts]

The primary reasoning for the nlogn complexity is the use of a divide and conquer technique. Because the algorithm divides the total array into smaller, more manageable arrays. It allow us to avoid large sections of if/else code statements (like those in lines 10-16 of ex1.py). This division of objects, but then the subsequent parse of each value, causes the algorithm to have a worst-case complexity of O(nlogn).

Question 3:



As you can see, the algorithm splits the list into a series of subproblems, and then proceeds to merge that sequence of arrays. By merging two smaller sorted arrays, it cuts down on the complexity of the overall algorithm.

4.)

Yes, the complexity is consistent with what we predicted. The merge's have a complexity of n. This is because they end up looking over each item of the array. While the division into smaller arrays allows for the problem to be represented in a log(n) complexity.