The performance of the sorting method implemented in this project is O(n(logn)). The reason we know this is that because for the n elements that we are adding, we insert the elements one at a time. Additionally, insertion happens at O(log(n)) since insertion gets called log(n) times at worst case and one call of the insertion helper function happens in constant time. Therefore, we can conclude the worst-case time complexity of inserting n elements O(n(log(n)).

However, the sorting performance could easily change as well. For example, if we were sorting a data structure that somehow took O(n) or even O(n^2) to compare between objects, this would increase our time complexity since comparisons would take longer and are used in the recursive insertion function. While the complexity of comparisons can change the complexity of the entire function, we do not have to worry about this for our examples since the comparison functions for numbers and the fraction class operate in O(1).