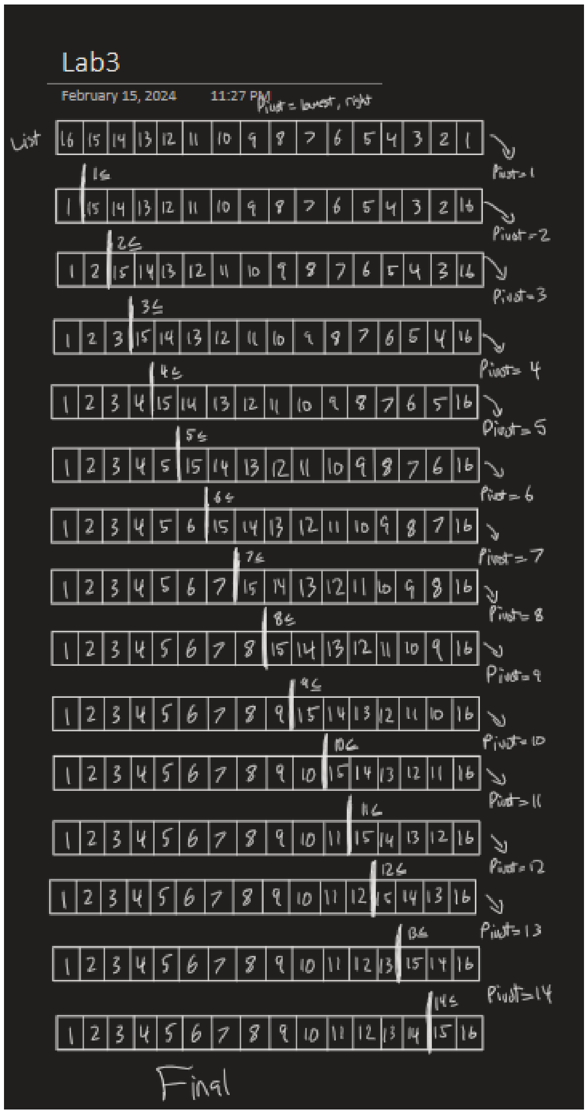
Exercise #4:

Question 1:

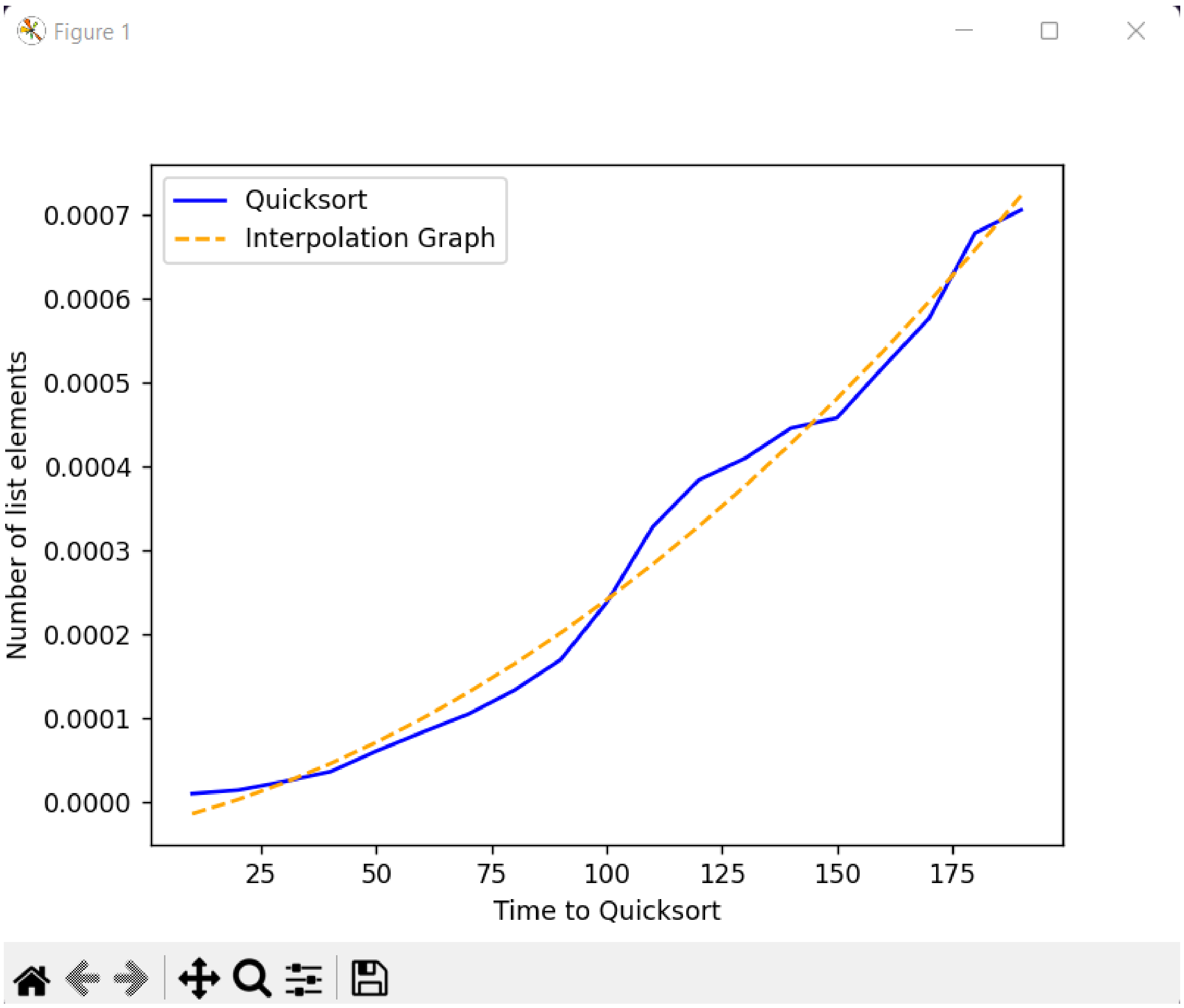
The formula for the worst-case complexity for the quicksort is (n(n-1)/2) -1. This goes through the array or list of n elements where the list is in reversed order, meaning the larget number is at the beginning and the smallest is at the end, or left side. It goes n through the first time it iterates and n-1 through the second time, and since it compares each element twice, we have to divide it by 2 and then it keeps going through it and adds 2 each time it goes through so n-2, n-3. It evaluates to (n(n-1)/2) -1 at the end. This later on evaluates to O(n^2) case complexity which is the worst.

Question 2:



Every time, the list is already sorted on the left but on the right side of the pivot point, it still needs to be sorted and it chooses the right side (lowest value) and it compares it to every other element on the right side to see where it should be.

Question 4:



The quicksort matches our complexity analysis and the graph of the times for the worst-case scenario, O(n^2). It also clearly shows how the more elements there are, the longer time quicksort takes. This graph shows the absolute worst case because the pivot was always the highest number or lowest number in the list of elements, and therefore, it concludes that when the pivot number is the highest number or lowest number in the list, the graph of it's complexity will be a quadratic that is much higher compared to the average or best-case scenarios.