Question 2: Show O() by mathematical analysis - Show all work/algebraic summative derivations.

Given: Selection and Bubble Algorithms (See GitHub repository).

Selection Sort Approach: With typical BigO() analysis, we look to find the worst case computation scenario for the algorithm. For selection sort, we have two nested loops, one running n-1 times, and the inner loop running n - pos - i for each iteration of the outer loop.

Once again, let's let T(n) be the total summation of the operations. So T(n) = $\sum_{pos=0}^{n-2}[C1 + \sum_{i=pos+1}^{n-1} C2 + C3$, where C1 is the linear operations before loop, C2 is the operations made inside the loop, and C3, operations after the loop ends. Following the expansion of the summation series, we're left with
$T(n) = (n-1) \cdot \left[(C_1 + C_3) + C_2 \cdot \left(\frac{n}{2} + 1\right)\right] = (n-1) \cdot \left[(C_1 + C_3) + \frac{C_2 n}{2} + C_2\right] =$
$(n-1) \cdot (C_1 + C_3 + C_2) + (n-1) \cdot \frac{C_2 n}{2}$

Taking into account constants being removed, we're left with $n(n-1)/2$, which is O(n^2).

Bubble Sort Approach: The given bubble sort algorithm contains a do-while loop that goes until swap, a flag, is no longer true. Inside the loop, there's a for loop that iterates through the array. So, for each iteration of the do-while, the inner loop iterates n-1 times.

The T(n) summation is as such: $\sum_{j=1}^{k}[C1 + \sum_{i=0}^{n-2} C2 + C3$. Which when expanded, equals $T(n) = \sum_{j=1}^{k} [C_1 + C_2 \cdot (n-1) + C_3] = k \cdot (C_1 + C_3) + C_2 \cdot (n-1) \cdot k$.

Since we seek the worst case, where k = n, the dominant term is $C2 * n(n-1)$, which is $O(n^2)$.

Selection Sort Conclusion: We calculated the worst case O() timing to be $O(n^2)$, which also extends to the best and average case since selection sort always performs a full run through of the array to find the desired element in each iteration.

Bubble Sort Conclusion: We calculated the worst case O() timing to be $O(n^2)$, where the best case is O(n) (meaning the array is already sorted), and the average case to be $O(n^2)$ because the number of parses is dependent on the initial arrangement of elements.