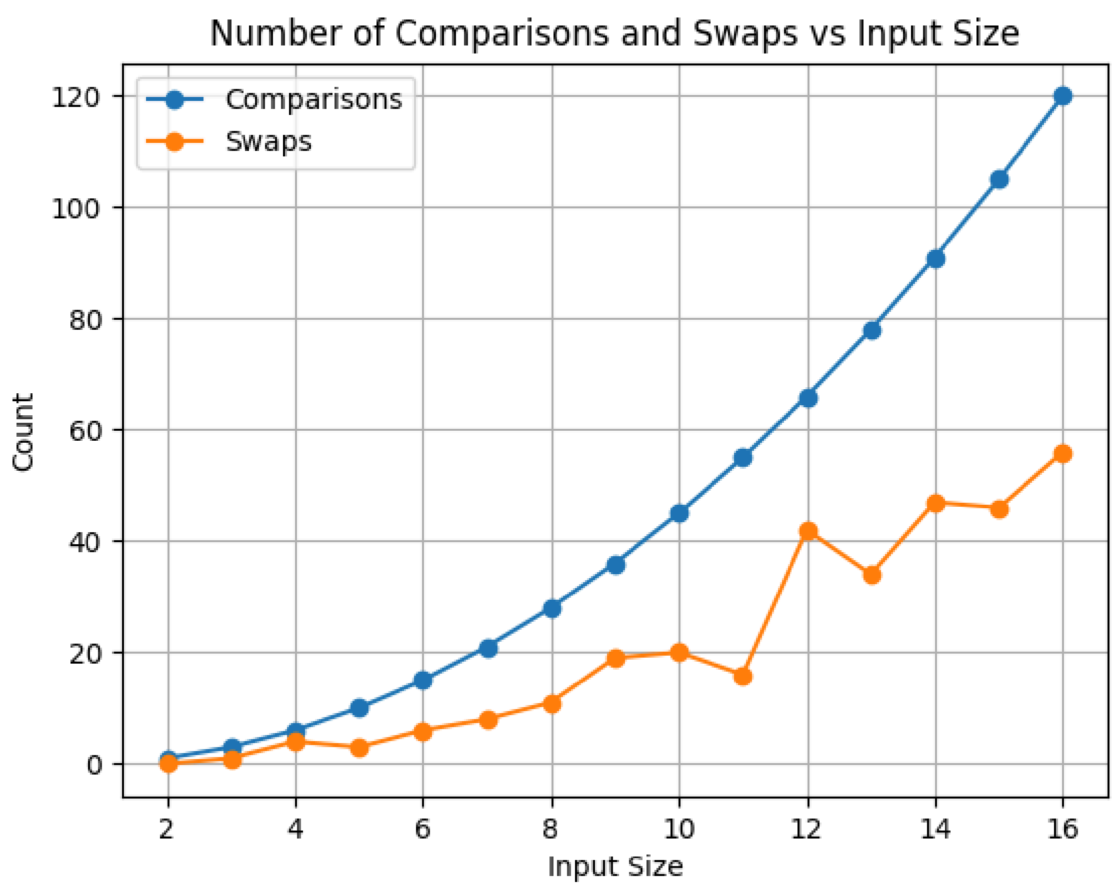
1. <u>Comparisons:</u>

Comparisons are done in bubble sort by comparing two adjacent pairs of elements together. So, for an array of n elements, n-1 comparisons are made on the first pass because the last element is already in the correct position. For the second pass, n-2 comparisons are made because the last two elements are already in the correct position, and so on. So by computing the sum of the arithmetic series, an array of n elements would have n(n-1) / 2 comparisons. Since comparisons in Bubble Sort are the same for the best, average and worst cases, the number of Comparisons in Bubble Sort is always **n(n-1) / 2.**

<u>Swaps:</u>

The number of swaps is dependent on the arrangement of elements in the array. The worst case being the number of swaps = number of comparisons if the array is in reverse order. The best case would be there would be no swaps at all (already sorted). So, the average number of swaps is **n(n-1) / 4.** By dividing the number of comparisons in half.

4.



Yes, the results plotted seem to match the complexity analysis discussed in question 1. Comparisons are the same regardless if they're the best, worst, or average case as there's always n(n-1) / 2 comparisons, as shown on the graph above. For the swaps, the swaps were dependent on how the elements were arranged, and the graph displayed mostly average-cases, which is around n(n-1) / 4 comparisons. There were a few almost worst-case scenarios as shown in input size 4 when the number of swaps almost equaled the number of comparisons.