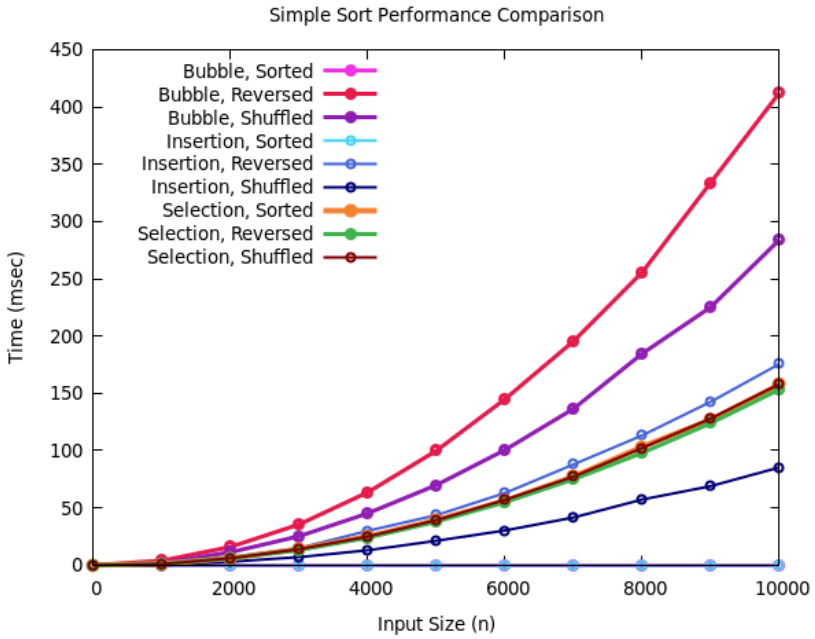
HW1 Writeup (Spring 2022)

Plot



Description

Looking at the plot above, one can derive that the quickest sort(s) occurred when using bubble and insertion algorithms on an already sorted array of integers. As both bubble and insertion sort have a condition to check if two current values are already sorted, it makes sense that starting with a list already in order would produce a fast execution. On the other hand, the worst performing instance was bubble sort in the cases where the array passed in was not already sorted. This too also makes sense because there are Boolean assignments that execute through each iteration of the nested loop structure.

Challenges

The main issue I faced was finding the proper range to iterate through given pseudocode for each sorting algorithm and a parameter of n to use as a loop termination. Before uncovering this issue, it seemed to only affect the reversed instance of each algorithm type. After careful consideration, I found that the range listed in the provided pseudocode was meant to be inclusive and I would have to add another iteration to each of my sorting algorithms to pass all test cases. The only other issue I found involved the background processes running on my computer. If I had other applications open when attempting to test the performance of my algorithms, and one happened to need more processing power at a given moment, the resulting graph depicted occasional spikes in sorting time.