Hypothesis

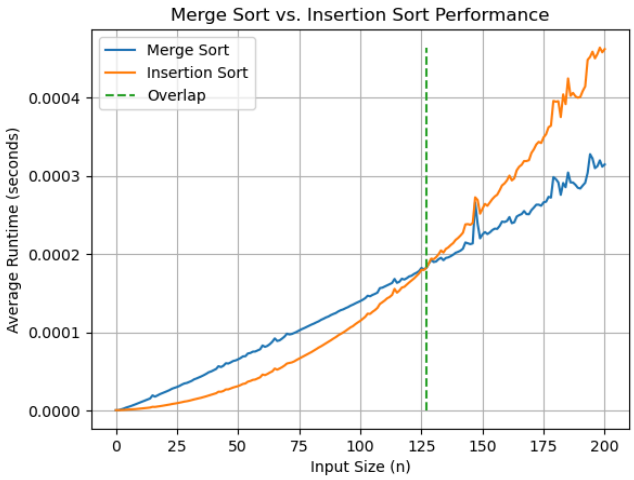
If Insertion sort is faster than Merge sort, then the length of the list that is being sorted will be less than or equal to 40. I hypothesize this as n=40 seems like it is low enough for Insertion sort to sort the list really quickly, but is large enough for Merge sort to start becoming more effective dealing with bigger lists.

Methods

Using Python I first utilized the Merge sort and Insertion sort algorithms created by GeeksforGeeks and set the random seed to 42. I created two empty lists to contain the average times for each sorting algorithm, and another list containing each n integer from 0 to 200. For each n in the list, I created a random list of n values containing random numbers from 0 to 100,000, and copied the list so I would have two identical lists, one for each sorting algorithm. I would then run each sorting algorithm on the data, measuring the time it took to complete each algorithm using timeit. I repeated this process 2,000 times for each n, storing the average runtime for each algorithm for each n in their corresponding lists. Finally, I created a graph showcasing the average runtimes for each algorithm based on the input size, with an extra line showcasing for what n Insertion sort starts taking longer than Merge Sort.

Source Code: https://github.com/Degancol/Colby-Degan-HW3-Q5-Submission/tree/main

Results



As seen in the graph above, Merge sort appears to follow its expected runtime of $\Theta(n\log(n))$ and Insertion sort appears to follow its expected runtime of $\Theta(n^2)$. The overlap line, which showcases the n that Insertion sort first starts taking longer than Merge sort, is marked at n=127.

Discussion

The data performed as I expected, with both sorting algorithms following their expected runtimes. I was surprised by how long it took for the algorithms to cross however, as I had hypothesized it would occur around 80 iterations sooner. The largest challenge for collecting the data was deciding how many times the data should be randomized and the sorting algorithms to be run to obtain accurate data for average sorting runtimes. I decided on 2,000 because it seemed like it would be large enough to obtain a decent sample size, as well as low enough for the code to not run for a long period of time.

Conclusions

Under the conditions tested, Insertion sort produces a faster algorithm for n less than around 127, while Merge sort is faster for n above around 127.

Sources

GeeksforGeeks. “Merge Sort.” GeeksforGeeks, www.geeksforgeeks.org/dsa/merge-sort/.

GeeksforGeeks. “Insertion Sort Algorithm.” GeeksforGeeks, www.geeksforgeeks.org/dsa/insertion-sort-algorithm/.