Name: Diane Li
PID: A15773774

Alpha: Selection Sort

- This sort made the least number of movements each time but a very high number of comparisons
- Additionally, as I doubled the data set size, the runtime also quadrupled, even for best case with InOrder list properties

Beta: Quick Sort

- This sort had an incredibly fast runtime on inputs of smaller sizes, making relatively many comparisons and fewer movements
- It's runtime also seemed to scale by a factor of O(nlogn)

Gamma: Bubble Sort

- With random numbers, as the data set changed from 8000 to 16000, the total time changed from 4 to 8, doubling with the data size and showing linear growth
- With InOrder list elements, the total time doubled as data set size doubled, showing best case O(n) time
- With ReverseOrder elements, the total time quadrupled as data size doubled, showing worst case O(n^2) time

Delta: Merge sort

- This sort had an incredibly fast runtime on a worst case input of large sizes. It also made relatively few comparisons and more movements
- In the best case, it's runtime scaled by a factor of less than O(n^2), concluding O(nlogn)

Epsilon: Insertion Sort

- With random numbers, as the data set doubled from 8000 to 16000 to 32000 to 64000 to 128000, the total time changed from 2 to 2 to 6 to 13 to 20, doubling with the data size and showing linear growth
- Epsilon showed double the total time, ⅓ the movements, and 1.5 times the comparisons as delta
- This sort took a dramatically long time, even in its best case InOrder scenario

Zeta: Check Sort

- With random data and n=10, this was the only sort with total time greater than 0 (94) and a great number of comparisons and movements
- With random data and n=20, I waited a few minutes and saw no results
- With random data and n=11, the total time was 260 and n=9 gave total time 7