a. Hypothesis

I expect that the hybrid sort will be slightly better than quicksort and insertion sort for small N values and then even out with quicksort for larger N values. I also expect that a K value of 25 or within 20 to 30 will give the best results. A N of 25 was the cutoff where quicksort became better than insertion sort in my previous experiment so I expect that 25 would be the best K for this experiment. I do not expect the difference of hybrid sort to be significant from the other two sorting algorithms, because the difference between insertion sort and quicksort is not very large at small values.

b. Methods

I will be using python to implement insertion sort, quicksort, and hybrid sort which uses both. I am using sources found on the internet for efficient implementations of the sorting algorithms to get the best efficiency and accuracy among the tests. In order to find the best K value to use for hybrid sort I am testing hybrid sort under a range of N size lists. I am also testing each K value for each N value to find the best K value for each N sized list. I tested each N size and K value ten times to get a good average. After I found the best N for each sized N, I chose the K value that had the best time the most out of all the tests. Once I figure out the best K, I can try tests of each sorting algorithm for random sized lists of size 1 to 50 (same as the previous experiment to get comparable results). I used an average again, running each sort for each algorithm 10 times to get a better accuracy. Once I had the average times for all sizes of N, I learned a 3rd order polynomial regressor again to fit a line to the times. I then graphed these points and lines to have a graph to analyze.

c. Results

Below (figure 1) shows the results from testing different values of K. I plotted a pie chart to see the distribution of the best K values for the different size N lists being sorted. Each slice represents a K value that is shown outside the chart. The percentages represent the percent of the best times that the K value was the best. As you can see the best two values was 12 and 13 and I ultimately chose to use 12 as K in the experiment.

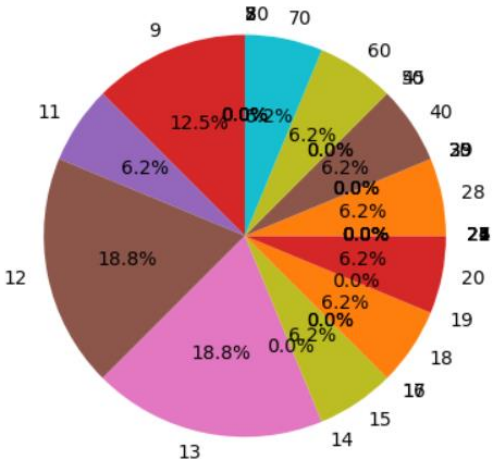

Figure 1

Shown below in figure 2 is the results from testing all three sorting algorithms. Dots correspond to times for each N for different sorting algorithms. The three dashed lines are the learned functions for each sorting algorithms times. The y axis is the time in seconds for the sorting times and the x axis is each N value tested.

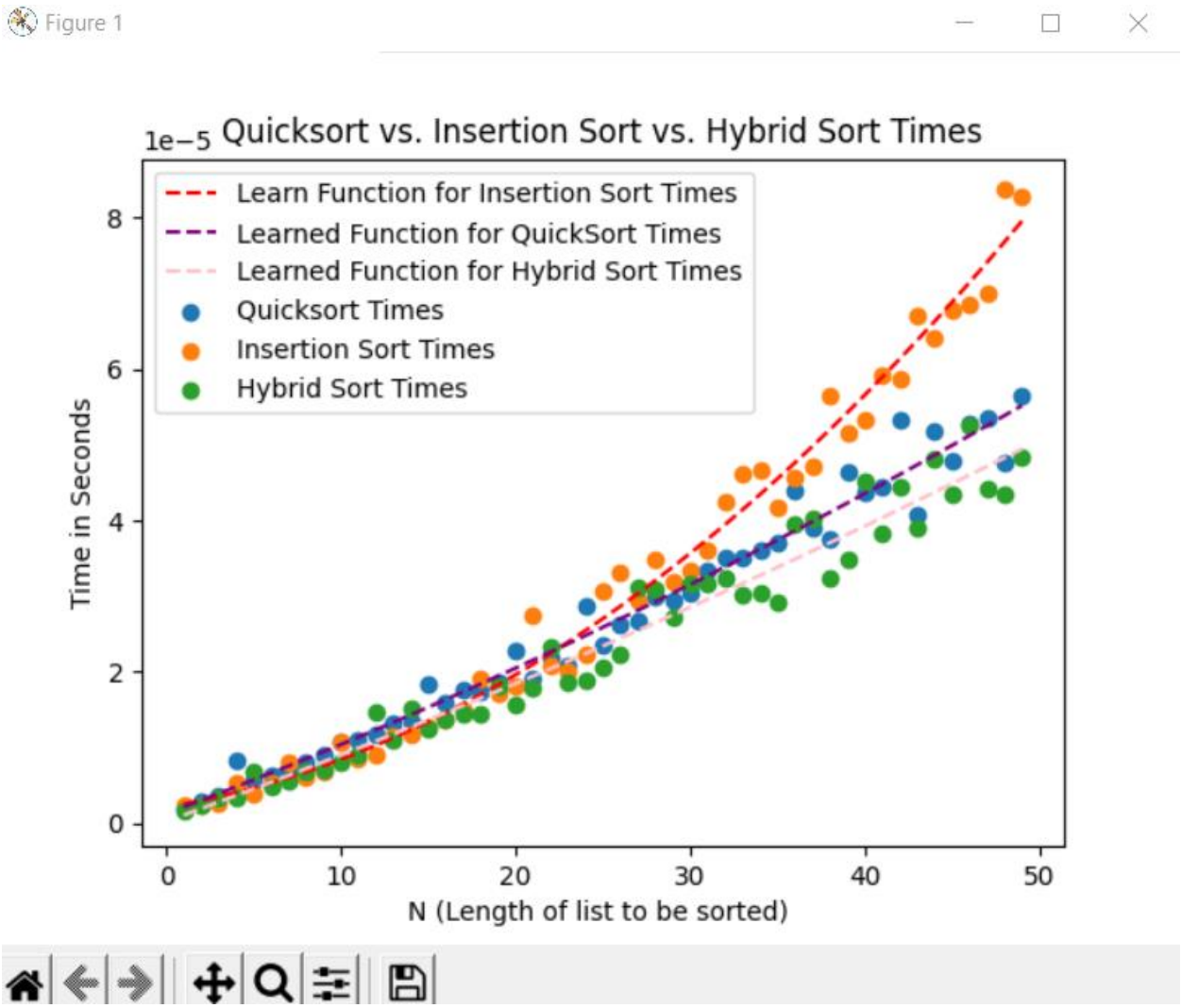

Figure 2

d. Discussion

I was first surprised that the K value that was best was 12 which I thought would have been 25. I expect that my prediction was incorrect because hybrid sorting continually uses insertion sort during recursion so continually using insertion sort should only happen at smaller K values. I was also surprised that as N got larger, hybrid sort separated from quicksort and became better at larger values. I thought it would have evened out, but because hybrid is a more efficient algorithm, it becomes better with larger N values. I expect that choosing the best K is not always clear when doing hybrid sorting for real world datasets because it does range based on the size of N. I do think that using 12 for K would be a good way to get consistently efficient sorting times. It seemed to be pretty indistinguishable for what was most efficient for smaller values and I expect it can differ on different tests and K sizes.

e. Conclusion

From this experiment I found that Hybrid sorting is indistinguishable from insertion sort in efficiency with N <= 18 and below but better than insertion sort and quicksort for N > 18.