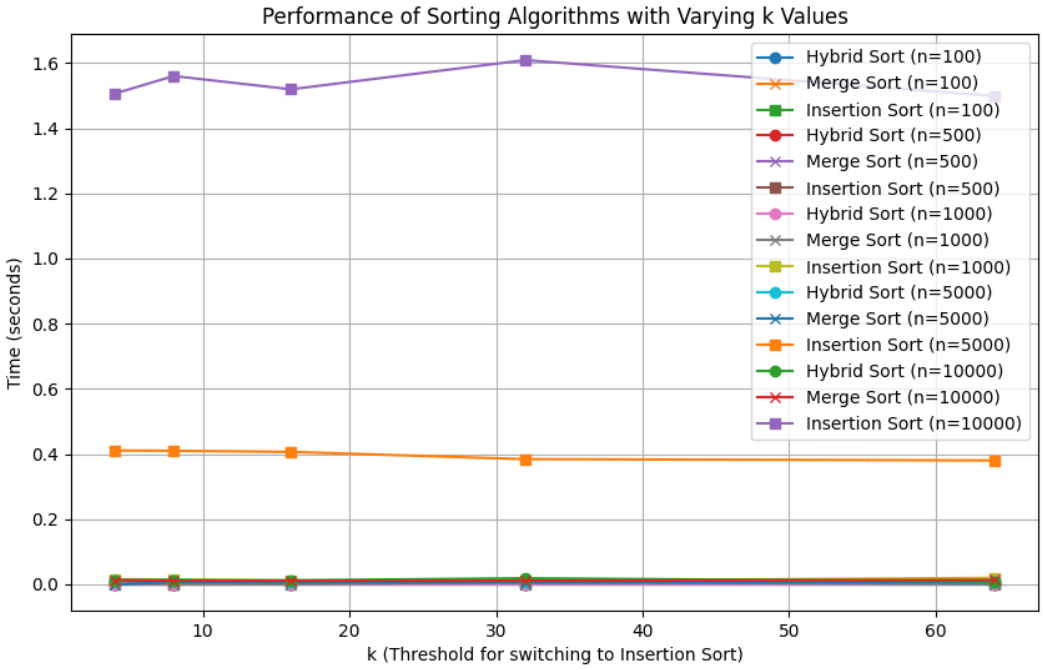
Hypothesis: I believe the k value will be like the one found in step one, as the crossover for what would be beneficial to include in the hybrid sort would overlap with where insertion sort started performing worse than merge sort. Therefore, it would be understandable to assume that value would be where the hybrid sort finds the best performance.

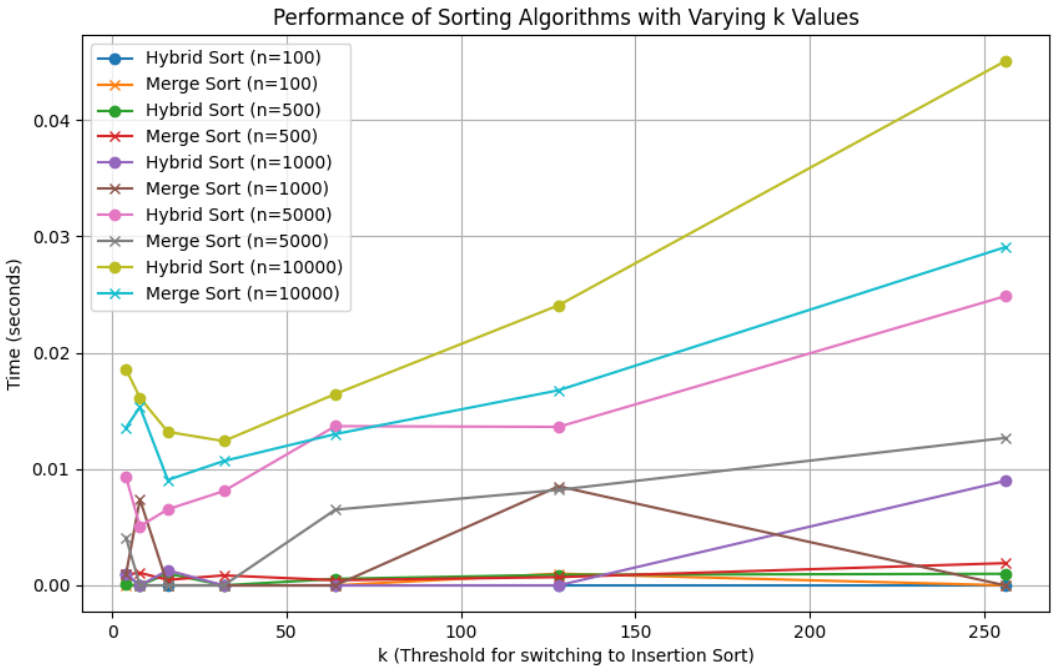
Methods: To test this, I had to first utilize my previous implementations of merge sort and insertion sort, however I had to include more code in the merge sort to account for when the array is smaller than the defined k value. Once I had implemented this in my merge sort, I was able to implement an overarching hybrid sort that was responsible for the structure of the entire sorting algorithm, taking the necessary parameters to determine when the insertion sort should take over for the merged arrays. To determine what value of k would be best, I had to make sure my algorithms tested several different k values, as well as array lengths to see the difference it had in the time complexity of the algorithms. Lastly, in order to visualize the results, I needed to use matplotlib to graph the results and understand how the data was behaving.

Results: I was able to generate a graph with the time it took for each separate sort based on the array length and the k value that was chosen. In my first graph, it is obvious that the larger insertion sorts were so much slower than the rest of the sorts it was virtually impossible to notice the differences between any of the others.



Comparison of All Sorts 1

In order to be able to visualize the differences in the k value, I decided to only graph the time complexity of the hybrid sort versus merge sort with the different k values and array lengths.In this graph, it looks like at around k=10, the hybrid sort starts to increase at much higher rates than that of the merge sort. This would suggest that a threshold around 10 would improve performance, but having a much higher value would decrease performance



Comparison of All Sorts 2

Discussion: I was surprised at just how low the k value threshold was that saw a benefit with the hybrid sort. Based on my previous data, I had suspected the k value could be as high as 100, however when dealing with true random arrays, insertion sort will slow down tremendously with higher array lengths. An unexpected challenge that I came across while performing this research is being able to ensure my sorting algorithms are working as expected, and don't have issues. After collecting the data, I was able to conclude that a threshold of k=10 would be best for the hybrid sort.

Conclusions: Under the conditions tested, a threshold of k=10 provides the fastest time complexity for hybrid sort, choosing to perform insertion sort on arrays less than size of 10.