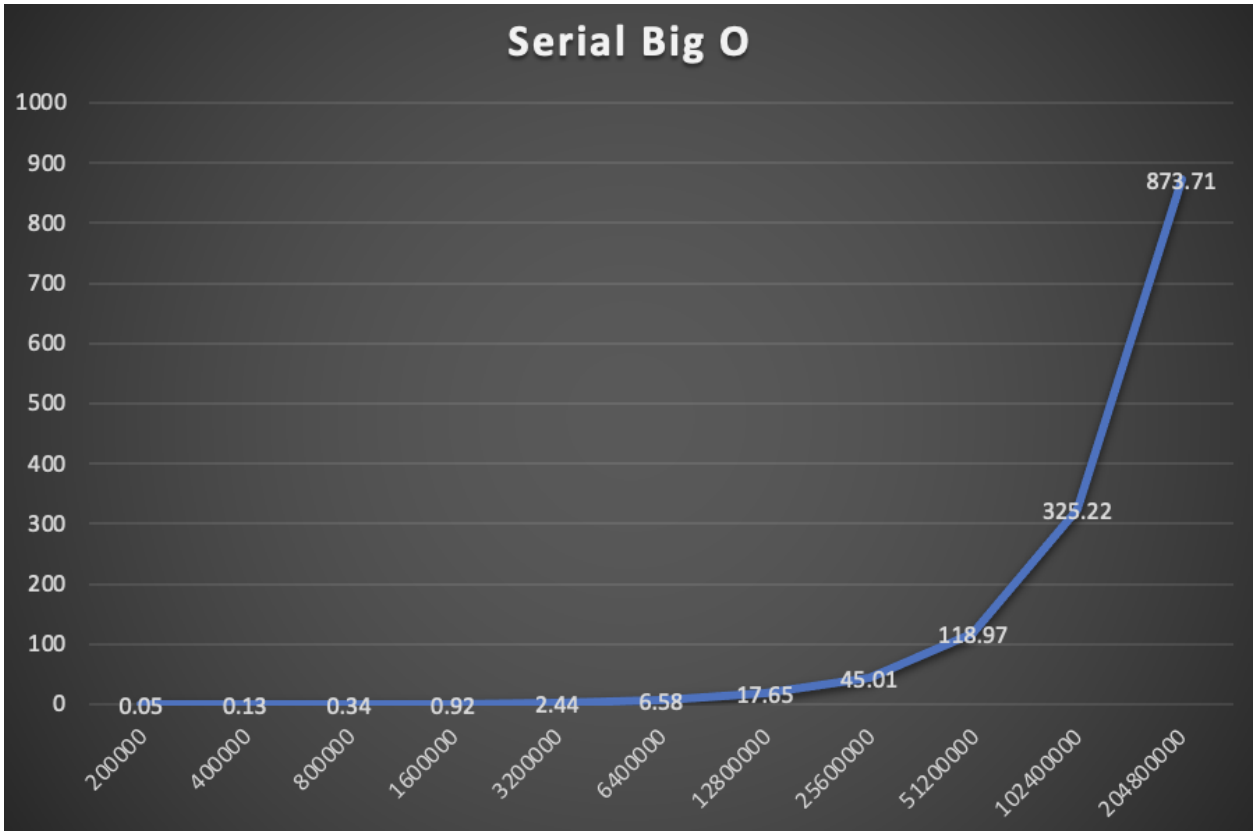
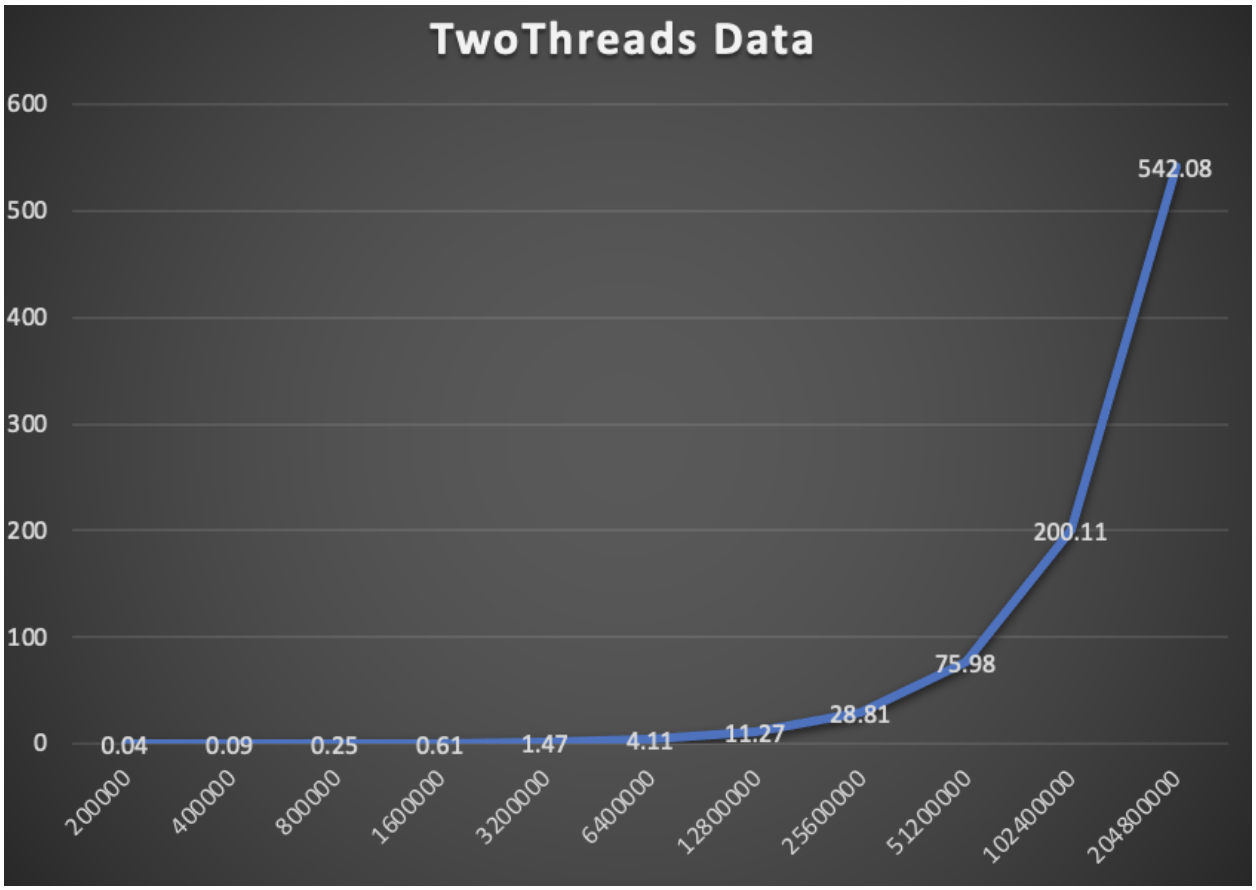
I would expect my Serial Algorithm to have a quadratic order of growth because my serial algorithm consists of a nested for loop, and the actual order of growth is quadratic, as can be seen in the graph below. The X-axis is the end parameter, and the Y-axis is time in seconds.

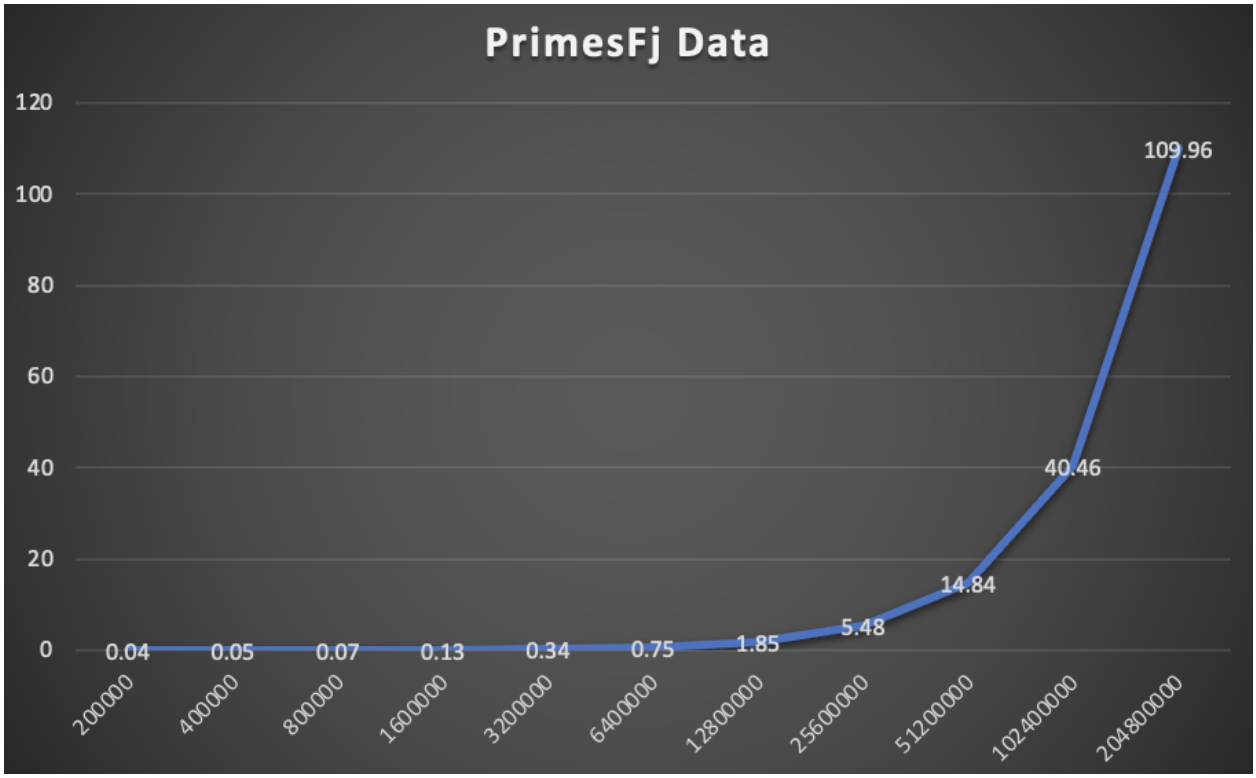


The theoretical best performance for TwoThreadPrimes is a 200 percent increase in performance. Below is a graph showing the values I used as the end parameter on the X-axis and how much time it took in seconds to run the algorithm on the Y-axis.

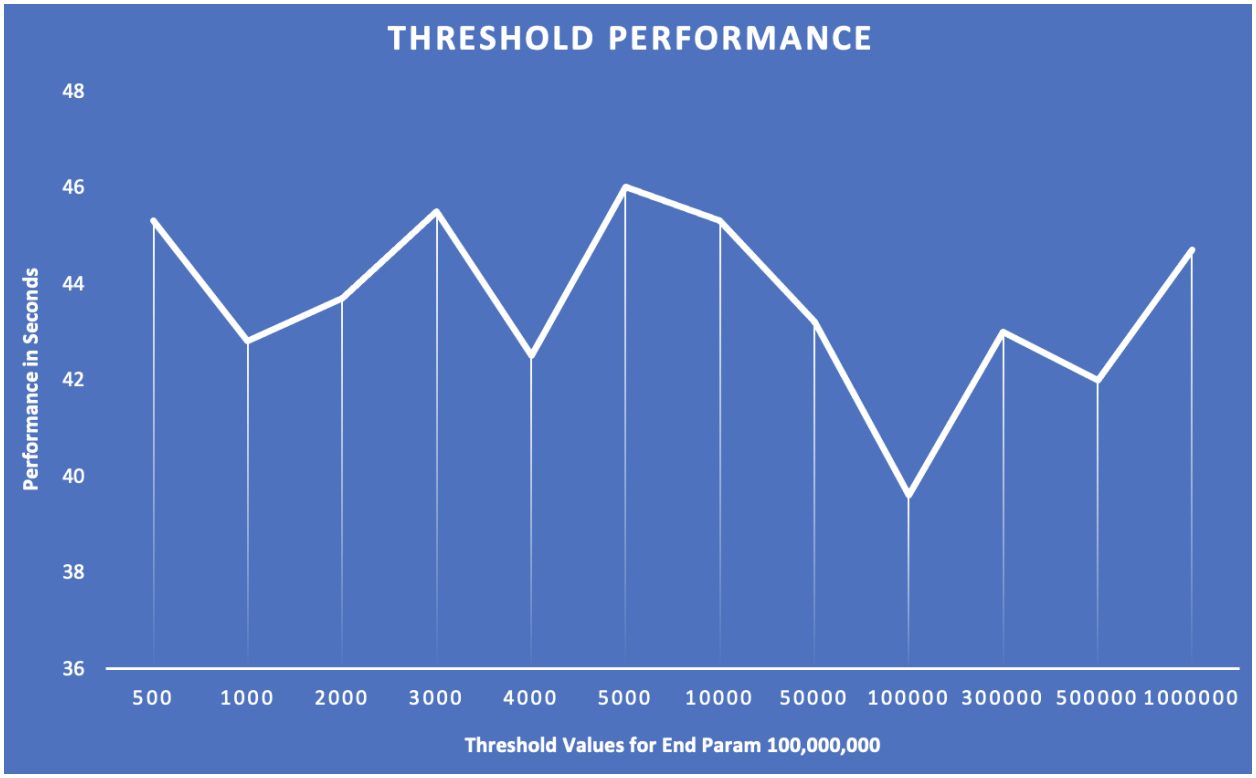


According to the data I collected, there is an actual 161 percent increase in performance, and the difference between the actual and theoretical performance can be attributed to the work imbalance. The order of growth remains quadratic.

My laptop has 8 cores (according to java 16) therefore the theoretical best performance on my machine would be an 800 percent increase. Below is a graph of my observations as it relates to performance.



From the data I have collected there is a 795 percent increase in performance. Observations from changing the threshold value with the end parameter set to 100 million are graphed below.



The threshold value of 100,000 returned the best results. The order of growth remains quadratic.

The order of growth between the three algorithms remains unchanged. The reason is that parallelization is not meant to change the order of growth but instead break down the order of growth work over n cores or threads.