Carson Sorenson

CS 5050

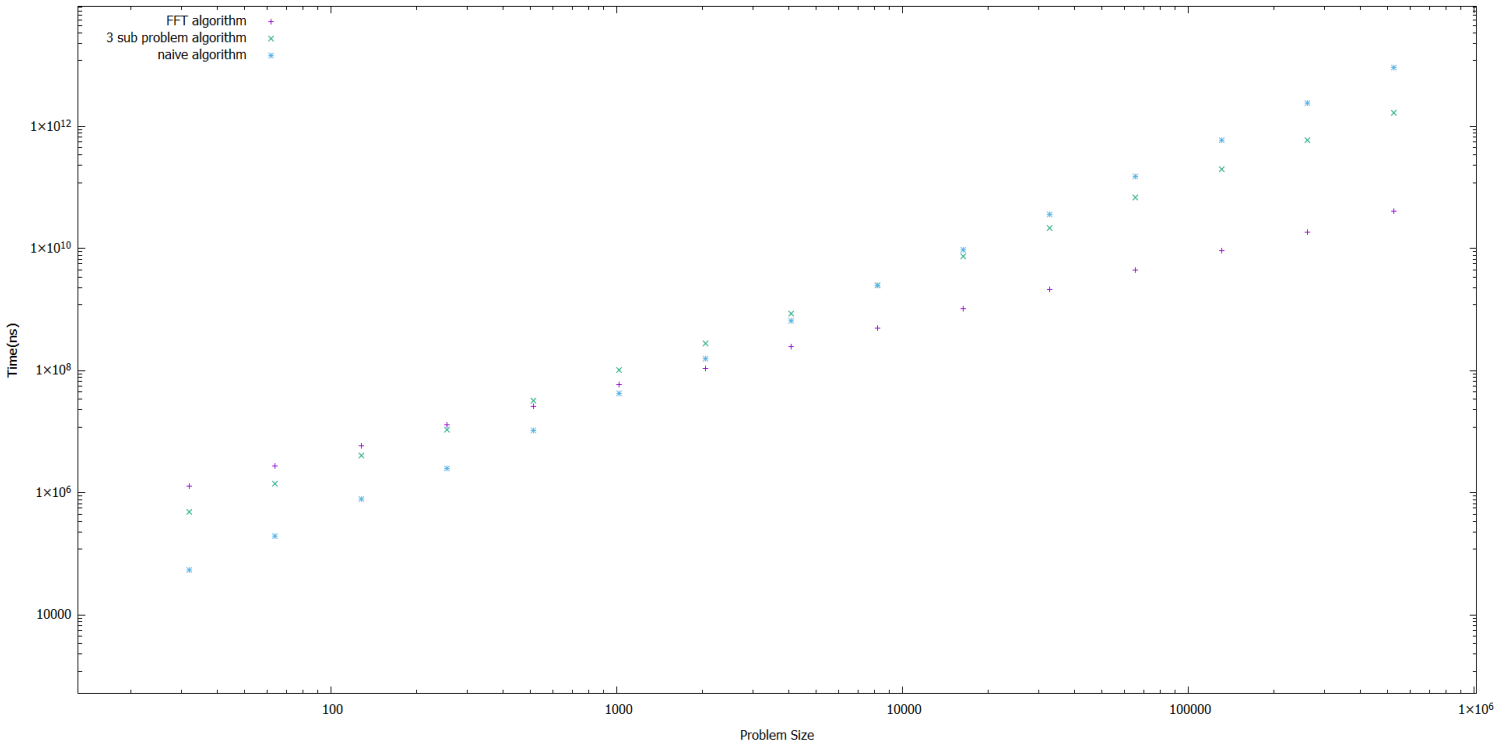
FFT Polynomial Multiplication

After timing each algorithm, the data was put into a .csv file called “data.csv”. I then calculated the equation for each of the fitted lines. After setting the Fast Fourier equation equal to the other equations and solving for n, I was able to find the value for which the FFT algorithm became faster than the other two algorithms. The value at which the FFT algorithm became faster than the 3-sub problem algorithm was with a problem size of 290, the value at which the FFT algorithm became faster than the naïve algorithm was with a problem size of 1312. This can also be seen by looking at the points in the graphs where the lines cross.

The Fast Fourier Transform algorithms graph looks linear because of its complexity. The complexity for the algorithm is O(n*log(n)). As n gets very large, the term log(n) becomes almost nonexistent compared to the n term, because of this the graph appears to be linear.

I set a limit on the runtime at 600 seconds or 10 minutes to see how big each algorithm could get before it exceeded the 10 minutes. The naïve, high school algorithm was able to reach a value of 2^18 or 524,288 until the algorithm could not run under 600 seconds. The three-sub problem algorithm was able to make it to 2^20 or 1,048,576 in just under 10 minutes. The FFT algorithm was much faster and was able to get up to 2^24 or 16,777,216 until it went over 10 minutes

Scatter Plot log-log graph



Linear fit log-log graph