Project 2 – Collinear Points

By Nicholas Donahue

Estimated Running Time

**Brute –** The brute class of this project uses a quadruple nested for loop to check each point within that loop against every other point. This is obviously an extremely inefficient and overall slow method (hence the name, ‘brute’-force) and is thus reflected in the run time.

**Estimated Runtime:** f(x) ~ N4

**Fast –** The fast class of this project utilizes a much more efficient algorithm to determine collinear points. Due to this increased efficiency, the runtime is thus increased. This is similar to a merge sort runtime as it utilizes a double nested for loop to check each point in the first loop to every point thereafter in the second loop. This, combined with the sorting time of the program (limited to O(nlogn) due to Collections.sort()) I can estimate the following runtime.

**Estimated Runtime:** f(x) ~ N2logN

Real-World Execution Time Data







*Graph on next page…*

*Enlarged to show detail…*

Estimation for N = 1,000,000

**Brute –** I believe the runtime would be around 1.004 \* 1012 seconds, or a staggering 31, 837 years!

**Fast –** I believe the runtime would be around 899,800 seconds, or 10.41 days!

The difference between these algorithms is extremely clear, Fast is incredibly faster than Brute die to its very reduced time complexity within the code.