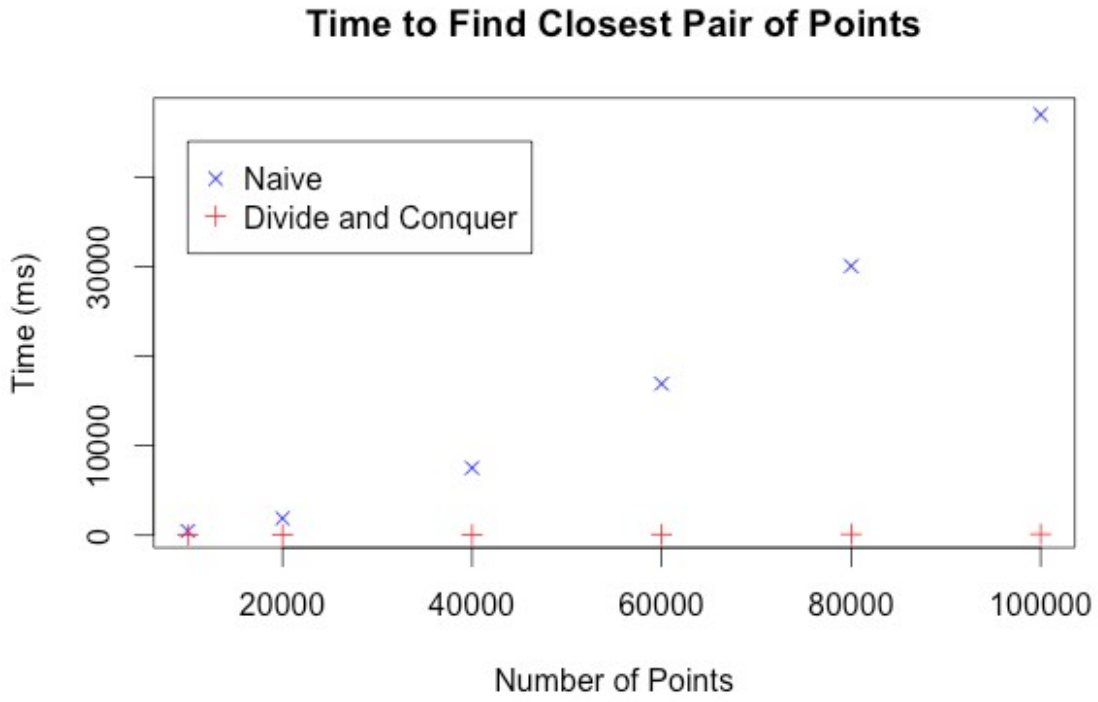
My FindClosestPair() divide and conquer method works by implementing a recursive Divide and Conquer algorithm, saving the closest points and closest distance via static instance variables. An unusual feature of my implementation is that the “centerPoints” array of points within the “combine” step is created as the same size as the array of points passed into the initial method call. This is done in order to find and track the number of points whose horizontal distance is less than the minimum distance within a single iteration statement.



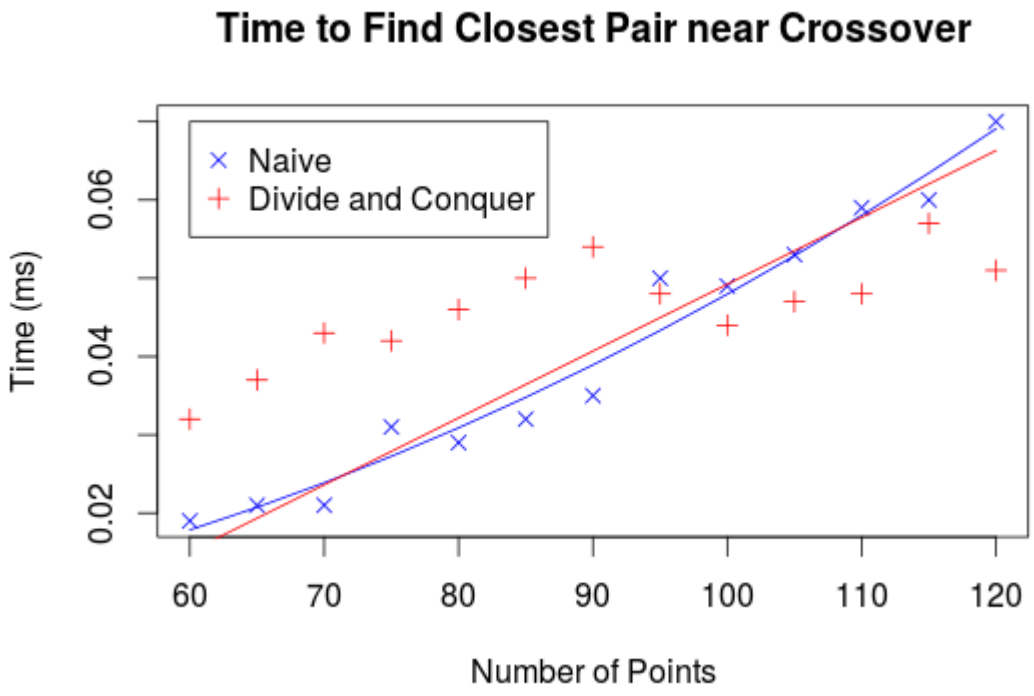
Divide and Conquer Trials:

Trial 1: 100 sets of of 50,000 Randomly Generated Points, different sets of points:
Maximum Time: 69 ms
Minimum Time: 20 ms
Average Time: 23.59 ± 1.45 ms (95% Confidence Interval)

Trial 2: 100 sets of of 50,000 Randomly Generated Points, same set of points:
Maximum Time: 69 ms
Minimum Time: 20 ms
Average Time: 24.17 ± 1.55 ms (95% Confidence Interval)

Finding the Crossover Point:

In order to find the crossover point, where the DC algorithm implementation becomes faster than the Naive algorithm, I altered the "Main" to run 4,000 trials. I then used guess-and-check methods to find the approximate crossover, and then analyzed 5ms increments around the approximate value. From our notes in class, I supposed that the Naive method ran in 2nd degree polynomial time. For simplicity, I assumed that the DC algorithm was in linear time (in truth, it runs in nlogn time, as we examined in class, but over this short scale, linear time should be a close enough approximation)



The formulae for these fitted lines are:
Naive:

$Time_{Naive} = .003022 - 5.604 \times 10^{-5}(Points) + 5.055 \times 10^{-6}(Points)^2$, Adjusted $R^2 = 0.962$

DC:

$Time_{DC} = -.0362 + 8.538 \times 10^{-4}(Points)$, Adjusted $R^2 = 0.9553$

Setting these Times equal to each other, and solving for *Points*, we see there are two possible solutions, $Points \approx 72$ and $Points \approx 108$. Given the data we have, the most logical crossover point seems to be at $Points \approx 108$.

Sources:

I used a few different references in order to to help me complete my coding and analysis for this project.

For help with the coding and algorithm:

Cormen, Thomas H. *Introduction to Algorithms*. 3rd ed. Cambridge, Mass.: MIT, 2009. Print.

Notes from the Course Website

Professor Buhler and the TA's via Piazza

For Help with Statistical Analysis:

Tamhane, Ajit C., and Dorothy D. Dunlop. Statistics and Data Analysis: From Elementary to Intermediate. Upper Saddle River, NJ: Prentice Hall, 2000. Print.

http://stattrek.com – various pages.

Various resources for R knowledge

Wolfram Alpha for intersection calculation