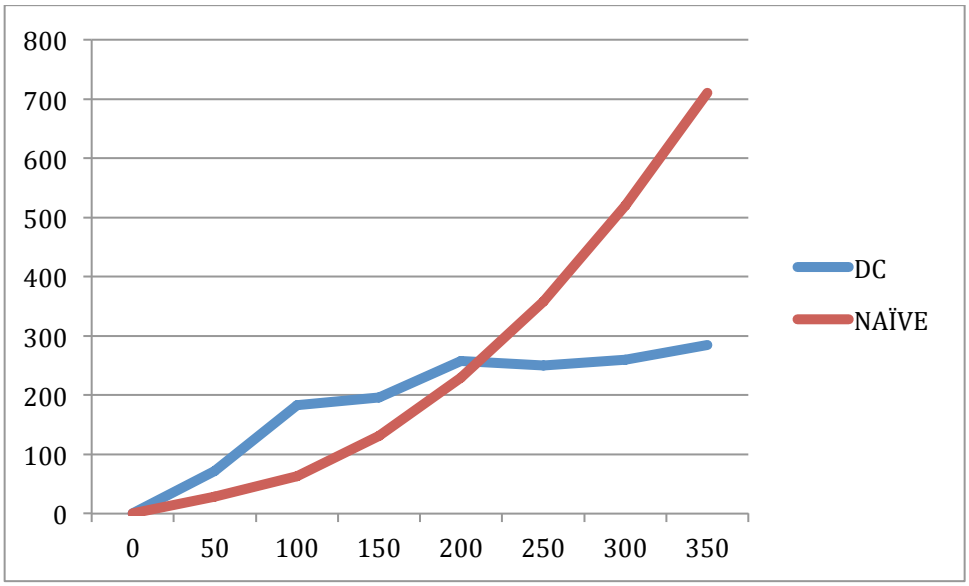
Grace Portelance
431602
ReadMe (lab 1)

Implementing the naïve algorithm was fairly simple. I used two for loops to iterate through my array of points, and brute force checked every distance.

Implementing the divide and conquer was much more difficult. I wrote the base cases, created my sub arrays, and checked those distances in my first method. I created a second method for combine() (which was arbitrary, but useful for me to keep the two sections separate in my head. I totally recognize there was no real need to make a second method). My most difficult problem was in a persistent null pointer exception in this method. I eventually realized that I was creating 'spaces' for n items when I was checking the distances, even though after each recursive call I would only have n/2. Once I modified my code to only run through the actual number of checks, my code was successful. I have not found any issues with it thus far.



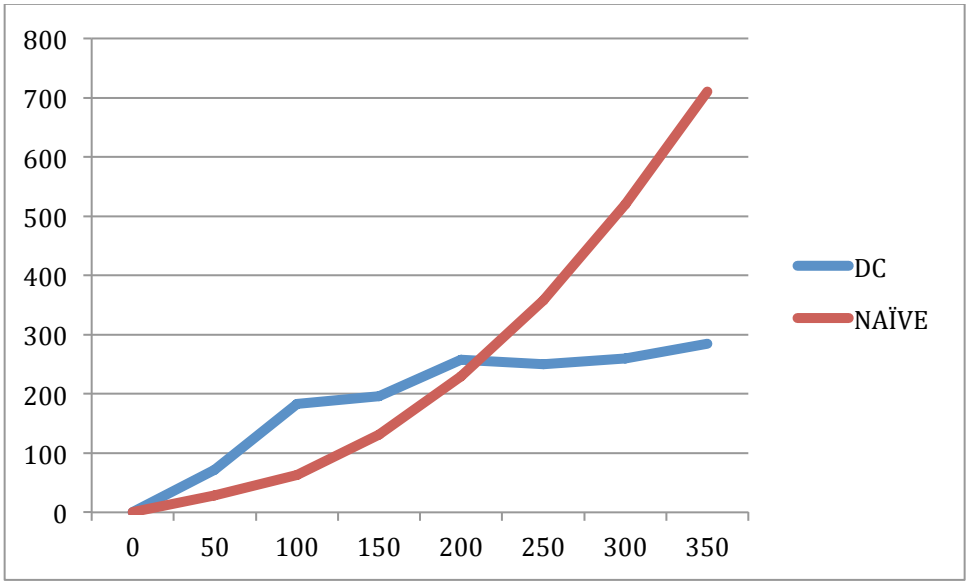
Run times for 100 sets of 50,000 points:

Different set of 50,000 points:
Average Time: 55.53
Minimum Time 38.0
Maximum Time 361.0
Same Set of 50,000 points
Average Time: 49.65
Minimum Time 37.0
Maximum Time 295.0

For both sets, the run time dropped drastically after the first few runs. Though the same set of points performed slightly better than the different set, it was by a potentially insignificant

amount as shown above. This indicates that the computer learns the more sets it runs, running the function more quickly.

To find the crossover points, I started guess and checking input sizes based on what I saw from past testing of my algorithm. I started at n=100, and ran 5000 trials of each. I found that at about n=185 the divide and conquer algorithm started winning the majority of the time. By n=200, the DC algorithm was winning every time.



As seen above, there is a crossover point around 200.
From this I can conclude that the DC algorithm is much more effective than the Naïve in running large sample sizes. While they were fairly similar before the crossover point, once they cross, the DC works much, much faster (as can be seen in the first graph)