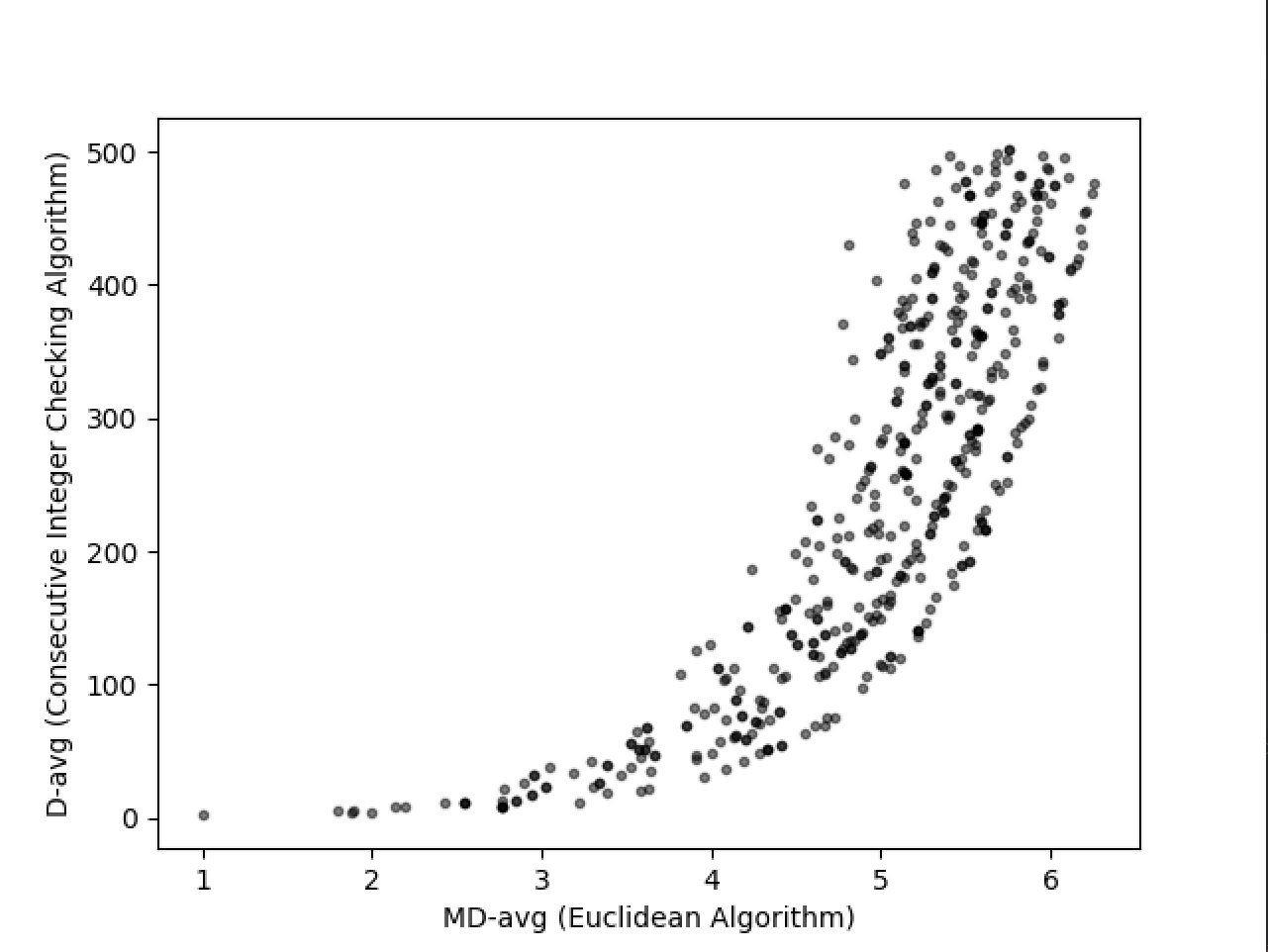
Michael Carr, Bryant Pinto

CS415

Project 1

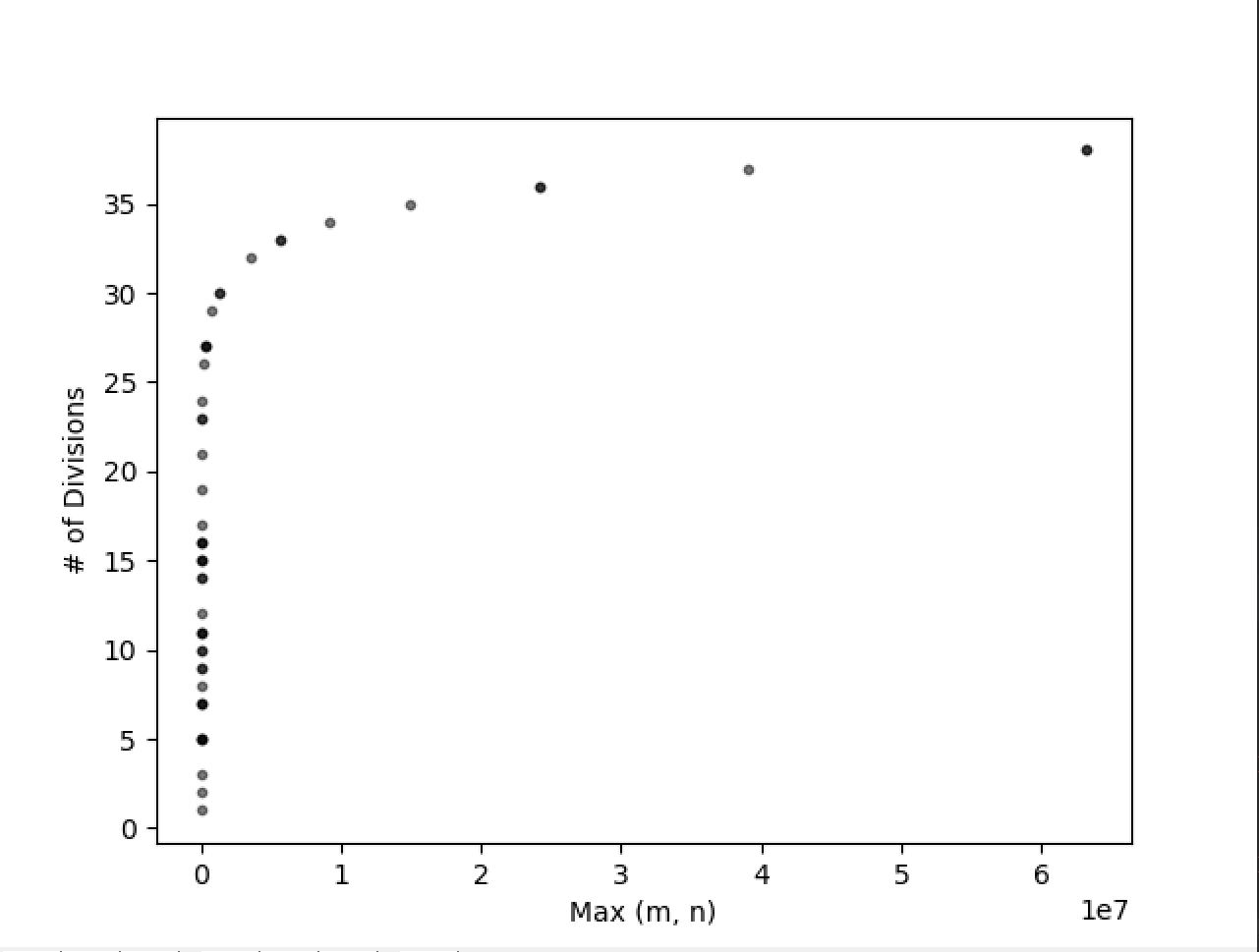
September 29, 2019

Task 1.



With the first task, the number n is 500 random values from 1 to 1000. Euclid’s algorithm requires only one basic operation which if m is greater than or equal to n the run time is in *O(log(n)) in the average-case.* The consecutive integer checking algorithm has at most 3 comparisons which will result in the running time to be in *O(n) in the average-case*. Small values of m and n yield the GCD faster than larger values on both algorithms, to an exponential number of operations. The consecutive integer checking algorithm will require a lot more basic operations versus Euclid’s algorithm.

Task 2.



## DOUBLE CHECK ##

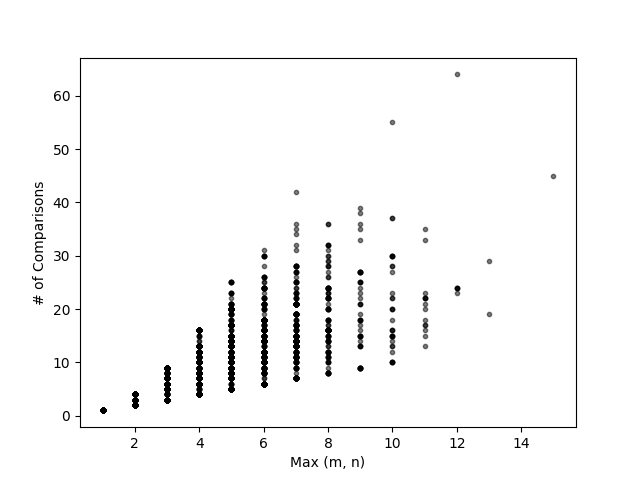
The second task demonstrates a correlation between the largest input value of M or N and the number of modulo divisions taken to compute the GCD. These values of M and N are used as values for the Fibonacci sequence, giving the values of the Mth and Nth term of that sequence. From there, the Mth and Nth term are used for the computation of their GCD. Smaller values of Max(M, N) have a large range of their number of divisions, but as Max(M, N) increases the number of modulo divisions increase.

The upper bound of k is O() as for the number of additions the algorithm has to make for any integer k.

The *worst-case* efficiency class of the algorithm for computing its GCD is in O() because we use the consecutive integer checking algorithm to compute the GCD of the values produced by F(k).

Using values from the Fibonacci sequence is a more time consuming process for computing their GCD as compared to using random value. As shown with the Fibonacci’s efficiency class versus the consecutive integer checking algorithm and the euclidean algorithm’s efficiency classes, they prove that Fibonacci is the worst in the case of computing values for GCD.

For smaller values of k the algorithm grows exponentially with many divisions and maintains that property as the value of k increases, until eventually being consistent at a steady amount of divisions being performed per GCD calculation.

Task 3.

The third task shows a linear correlation between the maximum size of either M or N on the x-axis, and the number of total comparisons between the two lists on the y-axis. With smaller lists the number of comparisons are equal to the number of comparisons to be made and increases as the list gets larger.

The input used were 5,000 random integers from the range of 2 and 50,000. A list of prime numbers were generated for each M and N (whose product resulted in each value of M and N), then the total number of comparisons made by both lists were graphed with the correlation of the list with the largest size.