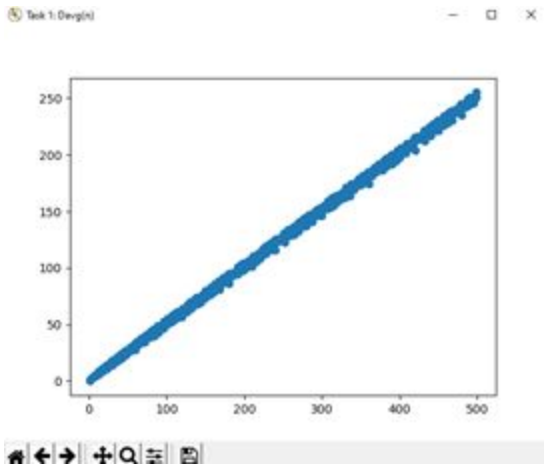
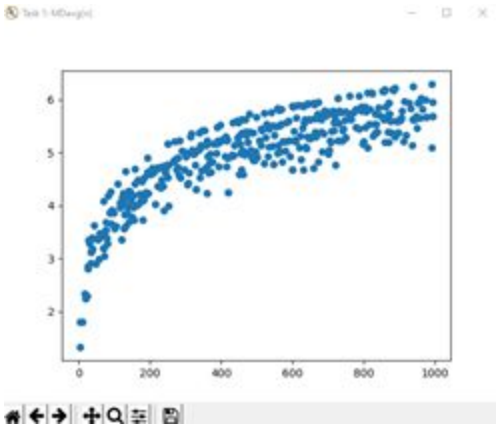
Authors: Spencer Greco, Nicky Valdecnas
Task 1:

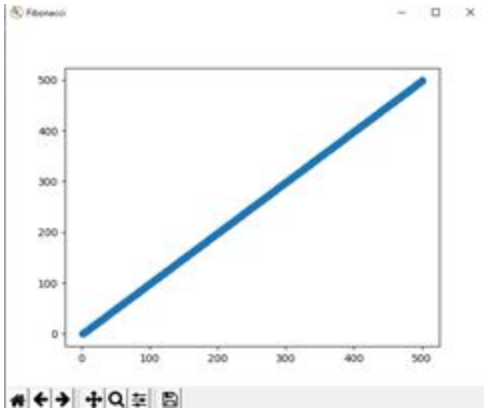


Graph of Average Normal divisions
Algorithms average-case efficiency: Θ(n)
Number Range(**n**): 1-500 incremented by 1.
The average number of decisions seems to be increasing in a linear fashion.



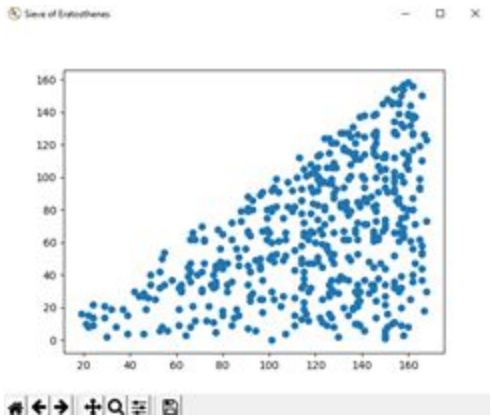
Graph of Average Modulus Divisions
Number Range(**n**): 1-500 incremented by 1.
Algorithms average-case efficiency class using Euclid's algorithm: Θ(logn)
The number of average modulus divisions slows down as the numbers used increase. When we compare these algorithms side by side, we can see that using modulus divisions is more efficient. The reason for this is because the number of basic operations used is much lower as the input grows larger.

Task2:



Number of Modulus Division taken to compute the GCD in the Fibonacci sequence(m)
Worst case efficiency class: n
How does this compare to the average case efficiency of Euclid's algorithm from task 1:
The amount of divisions in the Fibonacci sequence is roughly twice as many when compared to the amount of divisions used in Euclid's algorithm from task 1
How does the analysis change when measuring the actual time taken instead: The actual time taken

Task3:



Basic operations compared to max list length
Number range(**m**): Random input
Number range(**n**): Random input
The bigger the input the more basic operations required: Θ(g) time, where the x-axis is the maximum size of the two lists of common prime factors and y is the number of comparisons between two lists using sieves of eratosthenes to generate the prime factors from 1-k