Joseph Petitti Project 1 Report

To test the time efficiency of each algorithm, I timed the algorithms running the same *m* and *n* over ten trials. Recording the amount of time each algorithm took to calculate the GCD of *m* and *n* reveals which algorithm is most efficient for sufficiently large/small numbers. The big-O notation for time efficiency and space efficiency were calculated by examining the code and using logic.

The most time-efficient algorithm was Euclid's algorithm for sufficiently large input numbers. Because the Euclid algorithm divides the input number on each iteration, it has a worst case runtime of $O(\lg(n))$. The integer checking method was least time-efficient except for very small inputs with an easy-to-compute GCD. Its worst case runtime grows as $O(n)$. The middle school method was usually in between the other two algorithms, because its worst case runtime grows as $O(\sqrt{n})$, because the lists of prime factors check each number up to the square root of the input.

Euclid's algorithm and the integer checking method were most space-efficient, because they each only create one variable regardless of the size of the inputs. Therefore the space-efficiency of these algorithms is constant. The middle school method was the least space-efficient because it creates two arrays of prime factors, and another array of common prime factors, the size of which grows as $O(\lg(n \cdot m))$ with the input size.

		Runtimes (seconds)			
m	**n**	**Euclid**	**Integer Checking**	**Middle School**	**GCD**
31415	14142	5.9730e-6	1.6320e-3	5.3760e-5	1
1	2	7.2530e-6	2.1340e-6	9.3870e-6	1
1814274	259896	5.9740e-6	3.0140e-3	2.0053e-5	4998
181427400	25989600	5.5460e-6	2.8481	2.6027e-5	499800
24	60	6.4000e-6	5.5470e-6	2.0480e-5	12
202030202	20202020202	2.1334e-5	2.2448e+1	9.7280e-5	2
360	18	1.7920e-5	2.1330e-6	1.3653e-5	18
42	36	3.6940e-6	5.5460e-6	1.3653e-5	6
3	9	4.6930e-6	2.1340e-6	1.1947e-5	3
1234567890	987654321	2.5600e-5	1.7092e+2	6.9760e-4	9
	Time Efficiency:	$O(\lg(n))$	$O(n)$	$O(\sqrt{n})$	
	Space Efficiency:	$O(1)$	$O(1)$	$O(\lg(n \cdot m))$	