Executive Summary

I first conducted my experiment by making sure my three algorithms, the Euclidean, Consecutive Integer Checking (CIC), and Middle School Procedure (MSP), were running. Then I made a user interface that inputted the user's m and n inputs into the three functions. Then I had the user function return the answer to make sure the algorithm is correct and the time to compare time efficiencies. I filled those values in my table (see below). I learned many lessons such as analyzing the functions I wrote using Big O notation, and realizing that some functions are slower/faster than I actually thought they were. The most efficient algorithm with respect to time was the Euclidean algorithm. This is because there are no loops or if statements. There are just variables that assign each other values and returns the GCD. The CIC took the longest for big inputs and the MSP took the longest for smaller inputs. However, CIC and Euclid were the most space efficient while the MSP is the least space efficient. This is because for Euclidean and CIC, there aren't a lot of variables used and variables assigned to values. In MSP, there are three lists that are grow as prime factors are added. The MSP code does each major part step-by-step, which takes longer. It also uses a combination of if statements, for loops, and while loops which make the growth higher. Overall, Euclid was the most time and space efficient, CIC was the least time efficient for bigger inputs and the next most space efficient, and MSP was the least time efficient for smaller inputs and the least space efficient.

Trials	m	n	Euclid Runtime	Integer Checking Runtime	Middle School Procedure Runtime	GCD	Comments
1	31415	14142	0.000007153	0.004212141	0.001003981	1	
2	100	14230	0.000003815	0.000030756	0.000609875	10	
3	43	7	0.000003099	0.000010014	0.000024080	1	
4	900	1800	0.000002861	0.000007868	0.000025272	900	
5	678915	22320	0.000002623	0.007019043	0.000073910	45	
6	60	24	0.000003099	0.000011921	0.000024080	12	
7	588	988	0.000003099	0.000163078	0.000030041	4	
8	993	677	0.000001907	0.000098944	0.000314713	1	
9	9	789	0.000004292	0.000005722	0.000056744	3	
10	11	1353	0.000000954	0.000007153	0.000016689	11	

Time Efficiency:

To elaborate more on the time efficiency for the three functions, there are best cases and worst cases for each one. In general, the Euclid function runs faster than the other two functions. The best case for the Euclid function is when m and n divide evenly. This will always be a constant so O(1). The worst case for the Euclid function is O(log(n)) because it will always be smaller than O(n). The best case for the consecutive integer checking method (CIC) is also when m and n divide evenly which is a constant so again, O(1). When you have two prime numbers as inputs for CIC, it is the worst case so O(n) where n is the minimum of the inputs n and m. For the middle school procedure (MSP), the best case is O(1). This is because for first and second while loop, the best case is 2 (since it is checking to see how many 2's can divide evenly in the number given) and for the for loops, it is 1 (when you only have 1 element in the list). These will always be a constant so O(1). The worst case is O(n) because the first and second while loops go through O(n) times since there will never be anything bigger than the inputs m and n (when you do m/d or n/d), the first for loop goes through O(n) times depending on the inputs and its prime factorization, and you don't need to go through last for loop because the list of common prime factors will always be smaller than the list of prime factors which is also O(n). In other words, when you have the same numbers inputted for m and n, it would have to go through the while and for loops again and again. Altogether, it is O(n) where n is the number of prime factors for the first input.

Time Efficiency	Euclid	Consecutive Integer	Middle School
Best Case	O(1)	O(1)	O(1)
Worst Case	O(log(n))	O(n) Where n is the minimum of inputs n and m	O(n) Where n is number of prime factors for the first input

Space Efficiency:

There are also best and worst cases to elaborate more on the space efficiency for each of the three functions. The best case for all of the three functions is when it is constant. For Euclid, the best case is constant or O(1) because there are a fixed number of variables whose size is not changing and are just assigned to each other. For CIC, the best case is O(1) because even though the value of t is changing, the size in memory is not changing and therefore the space is fixed as there are no other variables added in. For the MSP, the best case is O(1) because the size and space would only change by a small amount caused by a constant. The worst case for Euclid and CIC is also when it is constant, but it varies for the Middle School procedure because it deals with adding different amounts to the list. The worst case for Euclid is O(1) because having two consecutive numbers in the Fibonacci sequence, which are constants. For CIC, the worst case is O(1). For both Euclid and CIC, since the size will not change, there is no growth progression so it is also constant. The best case for the MSP is O(1) since it means having one element/constant in the lists to work with. The worst case for MSP is O(n) because the size and space is changing/growing regularly as prime factors are being added to the lists.

Space Efficiency	Euclid	Consecutive Integer	Middle School
Best Case	O(1)	O(1)	O(1)
Worst Case	O(1)	O(1)	O(n)