Phillip Janowski

CS3130 – Project 1

Part E

The iterative formula has a theoretical growth of O(n) because for finding the Fibonacci series nth term, the program will go through the while loop n time.

|  |  |
| --- | --- |
| while(counter != nth) { | N + 1 |
| temp = current; | N |
| current = current + last; | N |
| last = temp; | N |
| counter++; | N |
| } |  |
| Return current | 1 |

The recursive formula has a theoretical growth of O(1.618^(n+1), an exponential growth due to the difference between n and two having to call the function twice, once for the Fib(n-1) term and once for the Fib(n-2) term. This recursion tree grows exponentially from the root, n, by 1.618, being the golden ratio, raised to the n+1 power.

|  |  |
| --- | --- |
| if(n <= 2) { | ((1+√5)/2)^(n+1) + 1 |
| return 1; | 1 |
| } else { | ((1+√5)/2)^(n+1) |
| return fib(n-1) + fib(n-2); | ((1+√5)/2)^(n+1) |
| } |  |

|  |  |  |
| --- | --- | --- |
| Nth Term | Iterative Time (nano seconds) | Recursive |
| 5 | 125 | 493 |
| 10 | 141 | 1525 |
| 20 | 220 | 112534 |
| 40 | 374 | 1.66526\*10^9 |
| 45 | 854 | 1.9176\*10^10 |

Part C Data

For the iterative function’s time, we do not see a linear time growth progression as we theoretically should have, however, this can be explained by the fact that was the nth term gets higher, the larger the numbers that that be added together, making later iterations take longer during the addition step of the function.

For the recursive function’s time, we do see an actual exponential time growth but not exactly in line with the golden ratio raised to n+1, due to the extra time it takes to add larger numbers, but the time grows exponentially none the less.