Christopher Merrill

CS162

11/28/2016

**Lab 10: Results, Comparison and Analysis**

|  |  |  |
| --- | --- | --- |
| **n** | **Iterative Time (sec)** | **Recursive Time (sec)** |
| 0-28 | 0 | 0 |
| 29 | 0 | 0.01 |
| 30 | 0 | 0.02 |
| 31 | 0 | 0.03 |
| 32 | 0 | 0.04 |
| 33 | 0 | 0.06 |
| 34 | 0 | 0.09 |
| 35 | 0 | 0.17 |
| 36 | 0 | 0.28 |
| 37 | 0 | 0.41 |
| 38 | 0 | 0.68 |
| 39 | 0 | 1.08 |
| 40 | 0 | 1.75 |
| 41 | 0 | 2.81 |
| 42 | 0 | 4.56 |
| 43 | 0 | 7.39 |
| 44 | 0 | 12.04 |
| 45 | 0 | 19.30 |
| 46 | 0 | 31.15 |
| 47 | 0 | 50.43 |
| 48 | 0 | 81.5 |
| 49 | 0 | 132.8 |
| 50 | 0 | 212.87 |

At this point it was taking too long to calculate the Fibonacci Number using the recursive method, so I continued on only with the iterative method to see which values of *n* would be comparable to the recursive time. From the table above, it is clear to see that the recursive method begins to slow down at *n*=29 and then continues to slow at an exponential rate as *n* increases. By n=50 it is already taking 3.5 minutes to compute.

|  |  |
| --- | --- |
| n | **Iterative Time (sec)** |
| 1000 | 0 |
| 10000 | 0 |
| 100000 | 0 |
| 1000000 | 0 |
| 10000000 | 0.03 |
| 100000000 | 0.31 |
| 1000000000 | 3.09 |

It is clear after a quick analysis that the iterative function performs much better for large n. Both functions perform the same for n < 29, but the recursive function slows down significantly after that point. The iterative function, on the other hand, doesn’t begin to slow down until n = 10000000.