Jeffrey Ngo

CS 162

Lab 10

12/8/19

Lab 10 Reflection

After looking at the two implementations, the recursion is less line of code compared to the iterative. Even though it is less, it seems more complex, where as the iterative implementation is easier to read. A difference between the two is that recursion calls itself and the iteration uses a loop to be repeatedly executed. Since recursion uses more memory, the recursive Fibonacci implementation should be slower than the iterative implementation.

Test Results on PC

N	Iterative Time	Recursive Time
5	0	0
10	0	0
20	0.002	0.001
30	0.001	0.06
35	0.001	0.65
40	0	7.209
45	0.002	80.19

Test results on Flip

N	Iterative Time	Recursive Time
5	0	0
10	0	0
20	0	0
30	0	0.03
35	0	0.4
40	0	3.03
45	0	35.7

Test results on flip server is much faster than when I compiled on my personal computer.

The recursive implementation was slower majority of the time. The only time it was quicker than iterative implementation was when N = 20. For the first few numbers of N the times were relatively close between two until N = 35 or greater. That is when the recursive implementation started to take much longer. For the iterative implementation all the times were close to each other. Not sure why it took longer the find the 20^{th} Fibonacci number than the 30^{th}, 35^{th}, and 40^{th} number. I only able to find the 45^{th} Fibonacci number, if I enter a larger number it would return a negative number. I tried searching for a solution for this and came across using unsigned long long int but it did not make a difference. The only time I was able to enter larger number for the iterative implementation was when I commented out or remove the recursive function. I was able to only find the 75^{th} Fibonacci number. I wish I was able to enter a larger number to see a more significant change in the iterative time.