|  |  |
| --- | --- |
| Computer Science 112 Computer Science with Java IISpring, 2016 |  |

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**Lab Report – Week *[1]* - *[Recursion Programming Assignment]***

*[Leo Vergnetti]*

*[CSCI 112-online] [Spring]*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**Assignment Analysis and Design**

*For this project, I chose the Fibonacci number program. This program requires two methods, one to calculate the program recursively, and the other to calculate the program iteratively. The program iterates through an array of test cases, and calculates the nth fibonacci number, it does so 5 times per method per test case. It uses System.nanoTime() to calculate runTimes in nanoseconds. Finally, the program outputs the average run times per method per test case to the console.*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**Assignment Code**

*Source code included in attached file.*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**Assignment Testing**

*To test this program i created an array of 10 values called testCase = {1,2,5,10,15,20,25,30,35,45}. In order to test these values, I used System.nanoTime() to calculate the runTime of each method. I did so 5 times / test case/ method to get a better average. By the third iteration, the recursive method was already taking significantly longer than the iterative method. By the 10th test case (calculate the 45th fibonacci number) the iterative method, running at an average of 2.2 microseconds, drastically outperformed the recursive method which clocked in at a whopping (wait for it) 9.3 seconds!*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**Assignment Evaluation**

*I think my knowledge of worst case runtimes, algorithmic complexity, and big O measurements is pretty subpar, but in this case at least, the iterative method completely outperforms the recursive method. The high overhead leads to such high runtimes as to almost make the 45th fibonacci number inconvenient to calculate. It ran at a little over 9 seconds so in 5 runs, it takes over 45 seconds to complete. That being said, I liked this assignment. It ties in perfectly with the chapter, and really helps to understand how the run times work. I think if I did it again, I would build a separate method for calculating averages, and let the user input dictate the values of both the nth fibonacci number, and the number of times it is calculated for the average runTime.*