Tristan Santiago
March 17, 2018
CS-162-400
Lab 10 Report

This experiment was relatively simple to conduct. Using cplusplus.com, I found two simple ways to calculate the time it takes to run iterative or recursive functions for Fibonacci numbers. When the program runs, the user first chooses whether they want to use the iterative or recursive function, then enters the number of terms to calculate. The program then calculates how long it takes the system to find the final number.

The first test I tried with recursive was n = 100. However, this test proved to be ill-conceived. The program never finished calculating the final number. Through systematic testing, I eventually worked my way down to n = 45, which still took quite a long time, all things considered.

This was where the real data was. Once I discovered I needed to be testing with 45 or less, I made sure to test the same number a few times, to verify the authenticity of the data. I was making sure the results were consistent. It is very clear who the 'winner' of the two is:

The Iterative function was much faster in all tests conducted.

The recursive function averaged about 25 second working through 45 numbers, while the iterative function was around .0004 seconds! These results were quite surprising to observe, since up until the n= 20 mark, both functions took almost precisely the same amount of time to calculate. It is worth nothing that I primarily conducted these tests in C9. I conducted a few tests in Flip as well, and noted similar results. Once I concluded my testing, I added input validation to only accept integers from 1 to 44, with the goal of improving the user's overall experience. In Flip, 44 terms with the recursive function takes approximately 28.63 seconds. 45 terms would take much longer. I felt the range of integers from 1 to 44 with both functions properly demonstrated the time disparity between the respective functions and as a result, decided to submit with those limits in place.

(Please note that results in different environments may produce slightly different results, though the relative outcome of the faster function should remain relatively the same.)