Jordan Hamilton

Professor Luyang Zhao

CS 162

29 November 2018

Lab 10 Analysis

Before I began testing the timing for recursive and non-recursive versions of the Fibonacci number calculator, my expectation was that the time required for the recursive version of the calculator would be exponential. For very small values of n, the number of function calls grew quickly from 3 for N = 2, to 9 for N = 4, so our original function call when n was 4 took 8 function calls before a value was returned. A for loop seemed more intuitive, and was more efficient by limiting the number of function calls and calculations that had to be performed with integers that fall out of scope more quickly at the end of the single function call.

Results

N	Recursive time (seconds)	Non-recursive time (seconds)
20	0.000158	0.000023
25	0.001173	0.000020
30	0.011667	0.000024
35	0.109590	0.000023
40	1.208	0.000027
45	12.374	0.000029
50	146.239	0.000035

As the table illustrates, the non-recursive time for this calculation is consistently quick, to the point that most values of N under 50 have roughly the same processing time. The recursive processing time is over 7 times longer for N = 25 than it is for N = 20, and over 12 times longer for N = 50 compared to N = 45. This illustrates how all the added function calls used in this algorithm rapidly deteriorate performance.