Kyle Karthauser
Lab 10
6/8/18

EXPECTATIONS

My initial thoughts are that the recursive function will be more efficient than the “normal” or iterative one. The recursive function simply adds two integers and moves on; the iterative function adds and reassigns a value to temp inside of a while loop and increments a counter variable.

The recursive function is “shorter” in terms of lines of code: the part that does the “work” is only one line! It calls f(n-1) + f(n-2).

The portion of the iterative function that “does the work” is 4 lines.

RESULTS

At first, I think I’m right (with n = 1).
My guess above (based on the number of “instructions” in the code) seems to be right. But in hindsight, it only applies in this one very limited case when there’s so little memory/processing overhead involved.

But then… even with an increase of one order of magnitude, suddenly we see that the iterative function is 2x as fast as the recursive.

It gets ugly from there.

The final delta, at n = 30, is 576,900%.

n	Recursive (nanoseconds)	Iterative (nanoseconds)	Delta (%)
1	1957	2180	Negligible
10	3623	1330	100
20	83,753	1452	5600
30	8,347,597	1446	576,900

ANALYSIS

In retrospect, and with a little bit of reading, this makes sense. The recursive function sucks up resources as it goes by calling increasingly large blocks of memory to be frozen. And once all the “dominoes” are in place, so to speak, and the function is going to get around to adding up all

these stored values, the values that are being added together are large. And the memory which those values occupied is freed up.

The iterative function, on the other hand, is doing one thing consistently and incrementing and storing as it goes. It requires less “moving parts” or less memory management and overhead, as each step doesn’t have to be preserved somewhere for later use. The data is fed in, processed, and discarded before moving on.