Program Name: Comparison of Efficiency of Fibonacci Algorithms

Program Description

This program measures and compares the difference of running time between recursive and iterative implementations of Fibonacci Numbers calculation. The Fibonacci functions were obtained from the link: https://www.codeproject.com/tips/109443/fibonacci-recursive-and-non-recursive-c (Links to an external site.)

Originally I tried to use clock_t to calculate the time consumed by the program, however, I found the results for iterative Fibonacci function in Linux were very insensitive in showing the increase from running bigger Fibonacci numbers, therefore I had to use timespec approach to get higher resolution for Linux environment (and thanks to classmate Kelley Ronald's for the advice).

From my testing runs, it appears that the Fibonacci function would need a really large N to produce a noticeable result. However, due to the limitation for the size of numbers an int variable can store, I chose to cap N to 30 for the Fibonacci numbers, but running or 10000 times instead, I have also converted the results to milliseconds instead of seconds to get a better presentation for the results.

Results and Analysis

The results from the program is presented in Table 1 and Figure 1 below.

The iterative function for Fibonacci shows a steady, linear increase of running time from 10th Fibonacci to 30th Fibonacci. While the recursive function increases exponentially. The equation derived from the results are $y = 0.0261x + 0.0186$ and $y = 0.0665e^{0.4822x}$ respectively.

I was surprised to see how the recursive function slows down the CPU efficiency. From the textbook I understood that the recursive algorithms require multiple functions calls and more memory allocation, and I expected that the recursive version of Fibonacci will be less efficient than the non-recursive version, however not this much. I always thought writing the algorithms recursively is a superior, smart move, now I do realize there is also disadvantage for it, so the efficiency should be taken into account when design large programs. However, I also understand some functions are best to be implemented recursively, such as The Towers of Hanoi.

Table 1 – Time required for two algorithms to run Fibonacci for 10000 times

N (run for 10000 times)	Time using Iterative (ms)	Time using Recursive (ms)
10	0.272015	8.20612
12	0.333168	21.5543
14	0.387628	56.6947
16	0.428624	149.677
18	0.510432	393.695
20	0.536002	1030.42
22	0.59616	2705.78
24	0.634098	7082.9
26	0.69908	18355.1
28	0.752731	48212
30	0.798905	127131

Program Name: Comparison of Efficiency of Fibonacci Algorithms

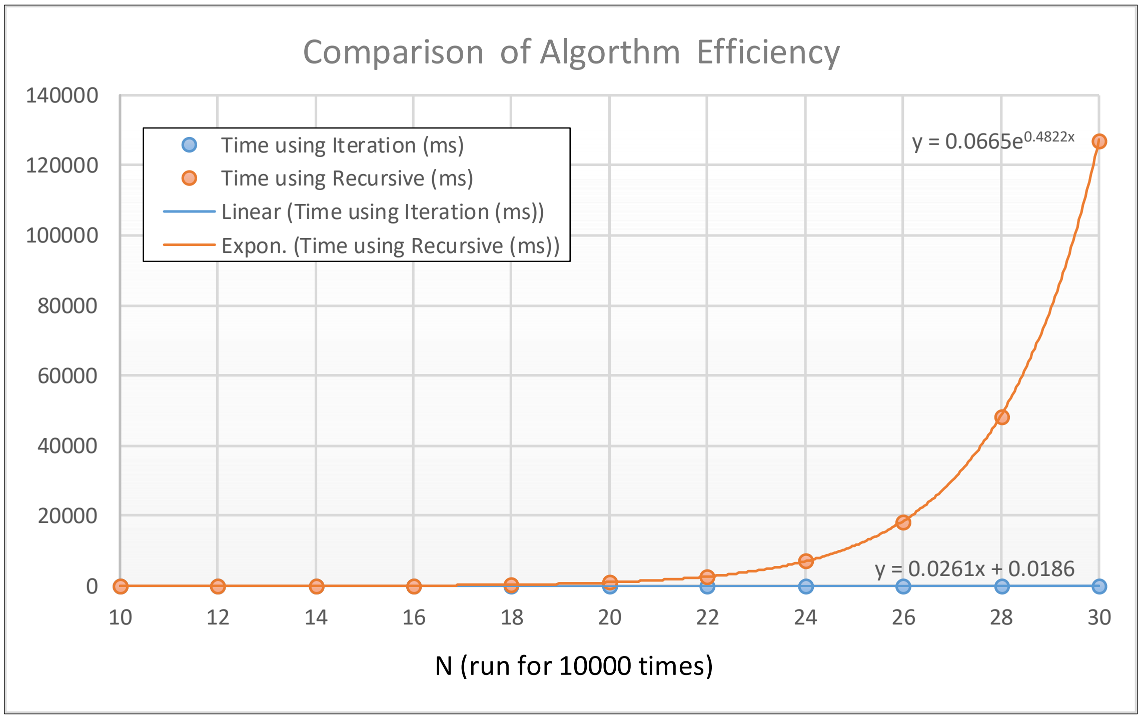

Figure 1 – Comparison between two Fibonacci algorithms