Programming Question Answers

a)

Binary Fibonacci : Based on result printed out from execution from java file, whenever the required Fibonacci numbers become higher, it takes drastically longer time. I can not even observe the value more than 50 Fibonacci number because it takes too long time to measure. Therefore, it is exponential algorithm.

Linear Fibonacci : On the other hand, linear Fibonacci is worked as linearly. The run time takes all in a similar time. Generally speaking, it is gradually increasing; however, sometimes, it takes less time than previous one because the run time is not only depending on the algorithm complexity, but also it is affected by other software running in background. Nonetheless, we can conclude that it is linearly working.

b) Briefly explain why the first algorithm is of exponential complexity and the second one is linear (more specifically, how the second algorithm resolves some specific bottleneck(s) of the first algorithm). You can write your answer in a separate file and submit it together with the other submissions.

Assume that binary Fibonacci time complexity is $T(n) = C + T(n-1) + T(n-2) \Rightarrow T(n) = C + (2C) + T(n-2) + T(n-3) + T(n-3) + T(n-4) \Rightarrow T(n) = C + 2C + 4C + 8(T(n-a)) \Rightarrow \ldots \Rightarrow T(n) = C + 2C + 4C + \ldots + 2^n C \Rightarrow T(n) = 2^{n+1}-1$. Therefore, its big Oh is $O(2^n)$ and we can conclude that it is exponential complexity.

On the other hand, linear Fibonacci is $T(n) = C + T(n-1) \Rightarrow T(n) = C + C + T(n-2) \Rightarrow \ldots \Rightarrow T(n) = Cn$. Its big oh is $O(n)$ and we know that it is linear.

Instead of return only 1 number, it returns pair of Fibonacci number (k and k-1) and if next Fibonacci is needed, simply become (k +1 and k). Since we do not call 2 times per each Fibonacci number, and we just call 1 time per each Fibonacci number, the complexity becomes very simple.

c) Do any of the previous two algorithms use tail recursion? Why or why not? Explain your answer. If your answer is ``No'' then

i. design the pseudo code for a tail recursive version of Fibonacci calculator;

ii. implement the corresponding Java program and repeat the same experiments as in part (a) above. You will need to submit both the pseudo code and the Java program, together with your experimental results.

Because tail recursion is defined when we return only the calling method itself (May have exception for base case) and only calls one time in one method. Therefore, Binary Fibonacci is definitely not a tail recursion since it is not a linear algorithm and linear Fibonacci is not a tail recursion as well because it is not calling at the last (return step). So, the answer is No.

i) Pseudo code

Algorithm TailRecFib:

Input: A nonnegative integer k, previous Fibonacci number, previous previous Fibonacci number

Output: Fibonacci number (in a reverse way)

If k = 1 then

    return 1

else

    return k-1, previous Fibonacci + previous previous Fibnacci , previous Fibonacci

ii) Based on the output from the java execution (Check out.txt in TailRecursion folder), it is shorter time than linear algorithm because tail recursion is tail recursion does not require stacks. Therefore, it is faster than linear algorithm. For Java file, please check another folder.