Lab 7 Recursion - II

Goal

In this lab you will design and implement improved recursive algorithms for computing Fibonacci numbers. Tail recursion will be explored as well.

Resources

- Chapter 7: Recursion
- `Lab7Graphs.pdf` – Full size versions of the blank graphs for this lab

Java Files

- *`RecursiveFibonacci.java`*
- *`TestFibonacci.java`*
- *`TimeFibonacci.java`*

A Better Version of Fibonacci

The first goal is to implement the better versions of fibonacci that were discovered in the pre-lab exercises. They will be timed.

Refer to the pre-lab exercises and complete the implementation of the method `better()` in `RecursiveFibonacci.java` (Use the height limited doubly recursive formula on slide 10.)

Compile `TestFibonacci.java`.

Checkpoint: Run TestFibonacci. All tests for the better Fibonacci formulation should pass.

A Tail Recursive Version of Fibonacci

Complete the implementation of the method `secondMSB()` in `RecursiveFibonacci.java`. Don't forget to change the return statement.

Complete the implementation of the method `reduceBy2ndMSB()` in `RecursiveFibonacci.java`.

Test the two methods you just created.

Refer to the recursive formulation from the tail recursion section on Fibonacci in the pre-lab exercises and create a recursive helper method in `RecursiveFibonacci.java` that uses `secondMSB()` and `reduceBy2ndMSB()`.

Complete the method `tailRecursive()` in `RecursiveFibonacci` that calls the tail recursive helper method you created.

Compile `TestFibonacci.java`

Checkpoint: Run TestFibonacci. All tests for the both Fibonacci formulations should pass.

Timing the implementations

Compile `TimeFibonacci.java`.

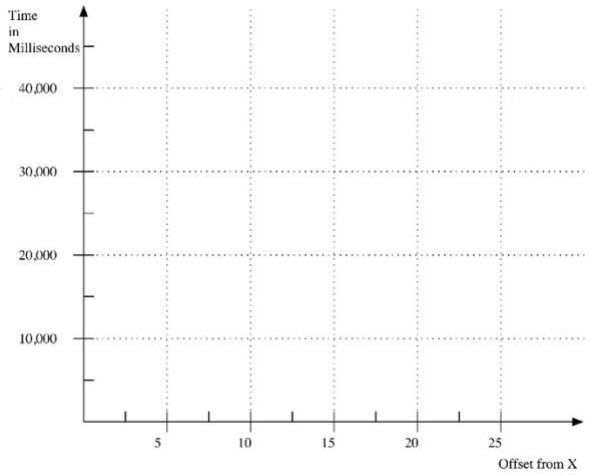
Run the code for succesively larger even values of n and record the first value of n for which the time is greater than 100 milliseconds. (24 is a good place to start your search.)

FIRST EVEN VALUE OF N FOR WHICH THE TIME OF BASIC FIBONACCI IS GREATER THAN 100 MILLISECONDS	X =

Fill in the values for *n* in the following table. Run the program and fill in the times. Stop timing when the time is longer than 100,000 milliseconds (about 2 minutes).

N	TIME IN MILLISECONDS TO COMPUTE F(N) USING THE BASIC FIBONACCI RECURSION
X =	
X+ 2 =	
X+ 4 =	
X+ 6 =	
X+ 8 =	
X+ 10 =	
X+ 12 =	
X+ 14 =	
X+ 16 =	
X+ 18 =	
X+ 20 =	
X+ 22 =	
X+ 24 =	
X+ 26 =	
X+ 28 =	
X+ 30 =	

Plot points from the preceding table on the following graph. Don't worry about plotting the points that are off the graph.



Draw a smooth curve that approximates the points.

Comment out the call to `timeBasic()` in `TimeFibonacci`.

Uncomment the code to time the better and tail recursive versions of Fibonacci in `TimeFibonacci`.

Run `TimeFibonacci`. Enter 100 for n and 100000 for the number of trials. Fill in the value in the table.

TIME IN MILLISECONDS TO COMPUTE F(100) USING THE BETTER RECURSIVE FORMULA	T =

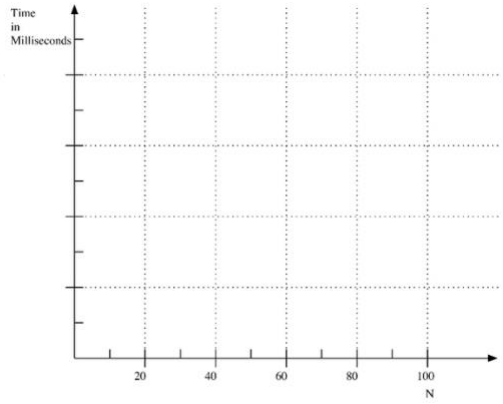
Complete the following computation.

TRIALS = 10000 / T =

Fill in the following table. Use the value of TRIALS from the previous step for the number of trials.

	TIME IN MILLISECONDS FOR BETTER FIBONACCI	TIME IN MILLISECONDS FOR TAIL RECURSIVE FIBONACCI
n=10		
n=20		
n=30		
n=40		
n=50		
n=60		
n=70		
n=80		
n=90		
n=100		

Plot points (in different colors) for the times for two different versions of Fibonacci in the table. Put appropriate value labels on the *y*-axis of the graph.



Note that even though the better formulation allows computations for larger values of N in terms of time, the computed value also grows exponentially and is still problematic. All the methods use the long data type and will quickly overflow.

Post-Lab Follow-Ups

1. Develop a recursive algorithm for computing the second most significant bit of a number n.

2. Develop a recursive algorithm for computing the result of removing the second most significant bit from a number n.

3. Look at the ratio of the times for computing Fibonacci numbers F(n) and F(n+2) using the basic recursive formula. Given that you know F(X), predict the amount of time it would take to compute F(X+50). Predict the amount of time it would take to compute F(X+100).

4. Write a loop to compute successive values of F(n) using the tail recursive version. For what value of n does the computation overflow?