CS4115 Week09 Lab Exercise

Lab Objective: We will use this second lab series to explore a numerical solution to the Fibonacci series. The Fibonacci series crops up in a number of different guises (worst-case AVL trees, analysing the running time a function that *computes* Fibonacci numbers, and other situations, too). Here's a quick summary of the tasks:

❶ Review the upper and lower bounds discussed in class and in homework

❷ Consider how we could derive a series of better and better bounds

In Detail

❶ We considered the Fibonacci sequence

$$F_n = F_{n-1} + F_{n-2}$$

and what its solution might be. Shown below is the argument that the n^{th} Fibonacci, F_n, is less than $\left(\frac{5}{3}\right)^n$.

Lemma 1. *If $F_n = F_{n-1} + F_{n-2}$ then $F_n < \left(\frac{5}{3}\right)^n$.*

Proof. We prove it by induction.
Base case: choose $n = 2$.

$$\begin{aligned}
F_2 &= F_1 + F_0 \\
&= \left(\frac{5}{3}\right)^1 + \left(\frac{5}{3}\right)^0 \\
&= \frac{5}{3} + 1 \\
&= \frac{8}{3} \\
&= \frac{24}{9} \\
&< \frac{25}{9} \\
&= \left(\frac{5}{3}\right)^2
\end{aligned}$$

Inductive case: assume that $F_n < \left(\frac{5}{3}\right)^n$ holds for any n that is less than some number K. We now want to use this assumption to climb to the next rung of the ladder: to establish that $F_n < \left(\frac{5}{3}\right)^n$ for any n less than $K+1$.

$$
\begin{aligned}
F_n &= F_{n-1} + F_{n-2} \\
&< \left(\frac{5}{3}\right)^{n-1} + \left(\frac{5}{3}\right)^{n-2} \\
&= \left(\frac{3}{5}\right)\left(\frac{5}{3}\right)^{n} + \left(\frac{3}{5}\right)^{2}\left(\frac{5}{3}\right)^{n} \\
&= \left(\left(\frac{3}{5}\right) + \left(\frac{3}{5}\right)^{2}\right)\left(\frac{5}{3}\right)^{n} \\
&= \left(\frac{15+9}{25}\right)\left(\frac{5}{3}\right)^{n} \\
&= \frac{24}{25}\left(\frac{5}{3}\right)^{n} \\
&< \left(\frac{5}{3}\right)^{n}
\end{aligned}
$$

□

The key to the inductive step argument is the fraction in blue above: find a fraction that is as close as possible to, but just less than, 1. With this in mind we can possibly do better than $F_n < (\frac{5}{3})$. If we generalise our solution and assume that it is going to be of the form $F_n = (\frac{a}{b})^n$ then we want to find the pair of integers a, b so that

$$\left(\left(\frac{b}{a}\right) + \left(\frac{b}{a}\right)^2\right)\left(\frac{a}{b}\right)^n = \frac{ab+b^2}{a^2}\left(\frac{a}{b}\right)^n < \left(\frac{a}{b}\right)^n$$

So we need $\frac{ab+b^2}{a^2} < 1$, but we also want to find better and better pairs (a_i, b_i) so that each successive pair is closer to 1.

$$\frac{a_{i-1}b_{i-1} + b_{i-1}^2}{a_{i-1}^2} < \frac{b_i a_i + b_i^2}{a_i^2} < \frac{b_{i+1}a_{i+1} + b_{i+1}^2}{a_{i+1}^2} < \cdots < 1$$

We also want $a > b$ for otherwise $a/b < 1$ would mean that $(a/b)^n \to 0$.

For the impatient amongst you the final ratio, $\frac{a}{b}$ turns out to be approximately 1.6180339887, the famous *Golden Ratio*, but this still doesn't give you the values of a and b. As always, Wikipedia has the full scoop on the significance of this number.

❷ We can think of the search for better and better (a, b) pairings as examining entries of a grid where the rows are increasing values of a and the columns are increasing values of b.

When $a > b$ we only need to look at entries below the diagonal. As we explore the grid we compute the formula $\frac{ab+b^2}{a^2}$ and save it if it closer to 1 than the previous best. Your task is to find as good an approximation to F_n as you can, when expressed as

$$F_n = \left(\frac{a}{b}\right)^n$$

with a, b integers.

One way you could begin to get a handle on pairs of a and b would be to create a small spreadsheet with the numbers $1, 2, \ldots$ in both the first row and column and then putting the "just less than 1" formula in each cell. By eyeballing the cells you could pick out better and better pairings.

But this method soon breaks down: you might spot the pairing $a = 5, b = 3$ and even $a = 13, b = 8$ this way but what chance do you have of finding the next best pairing of $a = 34, b = 21$? Clearly a program is needed.

Conceptually the procedure is straightforward: two `for` loops, one that iterates over possible integer values for a and the other for b. You might start off with some upper limits on a and b (to terminate the loops) but once the upper limits are reached the program terminates and no better values can be found. Sure, you could go back and change the upper limits to larger values and rerun but 1) this just finds the same values as in the previous run before searching through the "new" part of the search space, and 2) it's a pain to repeatedly edit-compile-run the program.

If you like you can stop here and you will get partial marks depending on how good a pair you can come up with in a fixed amount of time. The ambitious amongst you may want to read on, however. We have not thought of the exact fraction of the marks that will go for a solution along these lines but it will not be more than 5% out of the possible 8%.

You should soon realise that a better way is to let the program run *indefinitely*, like an infinite loop. But how do we do this? We need to search indefinitely in two directions, seemingly simultaneously, for if we allow our search to go towards infinity along one direction (say the a direction) then we may never find any improvement. Can you think of a way where the *two* directions are searched towards infinity simultaneously?

The remainder of the marks for this assignment will be given for writing the search program so that it can run indefinitely finding better and better bounds.