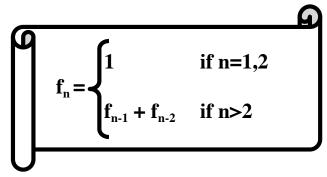
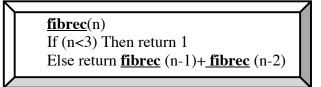
## Computer Science 308-250B Homework #2 Due Monday February 9, 2004, 13:30

In all the problems, <u>all calculations are done mod M</u> where M=3333373 in order to limit the size of the integers involved. This way we avoid problems due to overflow.

As we have seen in class, the Fibonacci numbers are defined as follows:

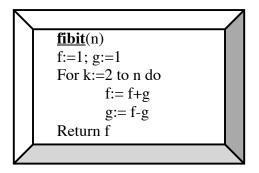


Thus the most natural way to write a program to compute one uses the next algorithm:



[15%] •1) Write a Java program that implements this algorithm. Find the time needed to compute  $f_1, f_2, f_3, f_4, f_5, ...$  and plot a graph of your results. What is the largest n for which you can compute  $f_n$  within 1 second ?

No **Java** details provided. The running time should be exponential. Indeed, the running time is  $\Omega(f_n)$  as explained in class before. An iterative way of computing the same values only keeps track of the last two values and use them to compute the new ones :



[15%] •2 Write a Java program that implements this algorithm. Find the time needed to compute  $f_1, f_2, f_3, f_4, f_5, ...$  and plot a graph of your results. What is the largest n for which you can compute  $f_n$  within 1 second ?

No **Java** details provided. The running time should be linear, i.e. O(n).

An alternate definition we have **NOT** seen in class for the Fibonacci numbers is:

$$f_{n} = \begin{cases} 1 & \text{if } n=1,2 \\ f_{n}^{2} + f_{n}^{2} & \text{if } n>2, n \text{ odd} \\ f_{\nu_{2}+1}^{2} - f_{\nu_{2}-1}^{2} & \text{if } n>2, n \text{ even} \end{cases}$$

[15%] •3a) Prove by mathematical induction that this new definition of  $f_n$  is correct.

**basis case:** n=1,2:  $f_1 = f_2 = 1$ induction step: Let n>2 and assume the formulas are correct for all k, 0<k<n. Then notation: F=f<sup>2</sup> **n odd**, (n-1 even, n-2 odd) **n even**, (n-1 odd, n-2 even)  $= f_{n-1} + f_{n-2}$ f<sub>n</sub>  $= f_{n-1} + f_{n-2}$ f<sub>n</sub> 
$$\begin{split} & =_*[F_{(n-1)/2+1} - F_{(n-1)/2-1}] + [F_{(n-1)/2} + F_{(n-3)/2}] \\ & = F_{(n+1)/2} - F_{(n-3)/2} + F_{(n-1)/2} + F_{(n-3)/2} \\ & = F_{(n+1)/2} + F_{(n-1)/2} \end{split}$$
 $=_{*}[F_{n/2}+F_{(n-2)/2}]+[F_{(n-2)/2+1}-F_{(n-2)/2-1}]$  $= F_{n/2} + F_{n/2-1} + F_{n/2} - F_{n/2-2}$  $= {}_{*} 2F_{n/2} + [f_{n/2} - f_{n/2-2}]^2 - F_{n/2-2}$  $= 2F_{n/2} + [F_{n/2} - 2f_{n/2}f_{n/2-2} + F_{n/2-2}] - F_{n/2-2}$  $= F_{n/2} + 2F_{n/2} - 2f_{n/2}f_{n/2-2}$  $=_{*} f_{n/2}[f_{n/2}+2f_{n/2}-2f_{n/2-2}]$ NOTE : "=\*" indicates use of the  $= f_{n/2}[f_{n/2}+2f_{n/2-1}]$  $= [f_{n/2+1} - f_{n/2-1}] [f_{n/2+1} + f_{n/2-1}]$ induction hypothesis. All other steps are  $=_{*} F_{n/2+1} - F_{n/2-1}$ simple algebraic manipulations.

[15%] •3b) Write a Java program that implements this algorithm. Find the time needed to compute  $f_1, f_2, f_3, f_4, f_5, \ldots$  and plot a graph of your results. What is the largest n for which you can compute  $f_n$  within 1 second ?

No **Java** details provided. The running time should be linear, i.e. O(n).

[15%] •4a) Prove the following by mathematical induction for all n>0.

$$\begin{pmatrix}
0 & 1 \\
1 & 1
\end{pmatrix}^{n} = \begin{pmatrix}
f_{n-1} & f_{n} \\
f_{n} & f_{n+1}
\end{pmatrix}$$

**basis case:** n=1,2 :

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{1} = \begin{pmatrix} f_{0} & f_{1} \\ f_{1} & f_{2} \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{2} = \begin{pmatrix} f_{1} & f_{2} \\ f_{2} & f_{3} \end{pmatrix}$$

induction step: Let n>2 and assume the formula is correct for n-1. Then

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^{n} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} n-1 \\ 1 & 1 \end{pmatrix}^{n-1}$$

$$= * \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} f_{n-2} & f_{n-1} \\ f_{n-1} & f_{n} \end{pmatrix}$$

$$= \begin{pmatrix} f_{n-1} & f_{n} \\ f_{n-2} + f_{n-1} & f_{n-1} + f_{n} \end{pmatrix}$$

$$= \begin{pmatrix} f_{n-1} & f_{n} \\ f_{n} & f_{n+1} \end{pmatrix}$$

A natural way of computing this exponentiation is to use a method similar to the one we used in Homework #1 for numbers (below I stands for the 2 by 2 identity matrix):

ExpMOD(A,b) If (b=0) Then return I Else If (b mod 2)=0 Then return ExpMOD(A<sup>2</sup>, b/2) Else return A(ExpMOD(A<sup>2</sup>, (b-1)/2)) [15%] •4b) Write a Java program that implements this algorithm. Find the time needed to compute  $f_1, f_2, f_3, f_4, f_5, ...$  and plot a graph of your results. What is the largest n for which you can compute  $f_n$  within 1 second (give an estimate if n is too big...)?

No **Java** details provided. The running time should be logarithmic, i.e. O(log n).



[10%] •5) If M is rather small (say M<1000) and you wish to compute the value  $f_n \mod M$ ; find a <u>very</u> fast algorithm to compute this value, even for very large values of n. hint: Show that there exists an integer R, 0<R<M<sup>2</sup> such that  $f_n \mod M = f_{n \mod R} \mod M$ .

Since the value of  $\mathbf{f}_n \mod \mathbf{M}$  is completely defined by  $\mathbf{f}_{n-1} \mod \mathbf{M}$  and  $\mathbf{f}_{n-2} \mod \mathbf{M}$  it is clear that whenever  $\mathbf{f}_{n-1} = \mathbf{f}_{n+R-1} \mod \mathbf{M}$  and  $\mathbf{f}_{n-2} = \mathbf{f}_{n+R-2} \mod \mathbf{M}$  then  $\mathbf{f}_n = \mathbf{f}_{n+R} \mod \mathbf{M}$ . Therefore if we find a value  $\mathbf{R}$  such that  $\mathbf{f}_{R+1} = \mathbf{1}$  (= $\mathbf{f}_1$ ) mod  $\mathbf{M}$  and  $\mathbf{f}_{R+2} = \mathbf{1}$  (= $\mathbf{f}_2$ ) mod  $\mathbf{M}$  then for any positive  $\mathbf{n}$  we have  $\mathbf{f}_n = \mathbf{f}_{n+R} \mod \mathbf{M}$  and thus  $\mathbf{f}_n \mod \mathbf{M} = \mathbf{f}_{n \mod R} \mod \mathbf{M}$ .

All is left to prove is that there exists an integer **R**,  $0 < \mathbf{R} < \mathbf{M}^2$  such that  $\mathbf{f}_{\mathbf{R}+1} = 1 \mod \mathbf{M}$ and  $\mathbf{f}_{\mathbf{R}+2} = 1 \mod \mathbf{M}$ . If we consider the sequence

 $f_1 \bmod M, f_2 \bmod M, f_3 \bmod M, \dots, f_{M^2} \bmod M, f_{M^{2}+1} \bmod M, f_{M^{2}+2} \bmod M$ 

we enumerate  $M^2+1$  pairs ( $f_i \mod M$ ,  $f_{i+1} \mod M$ ) of consecutive values of the Fibonacci sequence. Now since only  $M^2$  such pairs (x,y) exist, there must exist an i and a j such that ( $f_i \mod M$ ,  $f_{i+1} \mod M$ ) = ( $f_j \mod M$ ,  $f_{j+1} \mod M$ ). Let  $R=i-j<M^2$ . Notice that if we start with ( $f_i \mod M$ ,  $f_{i+1} \mod M$ ) = ( $f_j \mod M$ ,  $f_{j+1} \mod M$ ) then  $f_{i-1} \mod M = f_{j-1} \mod M$  as well because  $f_{n-1} = f_{n+1} - f_n$ . We obtain ( $f_i \mod M$ ,  $f_{i+1} \mod M$ ) = ( $f_j \mod M$ ,  $f_{j+1} \mod M$ ) which means that ( $f_i \mod M$ ,  $f_{i+1} \mod M$ ) = ( $f_{i+R-1} \mod M$ ) = ( $f_{i+R-1} \mod M$ ,  $f_{i+R-1} \mod M$ ) which implies that ( $f_{i-1} \mod M$ ,  $f_i \mod M$ ) = ( $f_{i+R-1} \mod M$ ,  $f_{i+R-1} \mod M$ ) which implies that ( $f_{i-2} \mod M$ ,  $f_{i+R-1} \mod M$ ) which implies that ( $f_{i-1} \mod M$ ) = ( $f_{i+R-2} \mod M$ ,  $f_{i+R-1} \mod M$ ) which implies by induction that

 $(\mathbf{f}_1 \mod \mathbf{M}, \mathbf{f}_2 \mod \mathbf{M}) = (\mathbf{f}_{R+1} \mod \mathbf{M}, \mathbf{f}_{R+2} \mod \mathbf{M}).$ 

