4005-800 Algorithms

Homework 2

Christopher Wood

March 30, 2012

PROBLEM 1.

Solution.

The time complexity of the recurrence F_n can be defined as follows.

$$\begin{aligned} T_F(0) &= 1 \\ T_F(1) &= 1 \\ T_F(n) &= T_F(n-1) + T_F(n-2) \end{aligned}$$

The solution to $T_F(n)$ can be solved using the method of homogeneous equations, which yields the result that $T_F(n) = \Theta(\phi^n)$, where $\phi = \frac{1+\sqrt{5}}{2}$ (the golden ratio).

PROBLEM 2.

Solution.

The time complexity of the $fibIt$ routine can be found by solving the recurrence relation that defines $fibIt$. More specifically, we can such a recurrence relation for $fibIt$ by analyzing the number of additions performed, which corresponds to the following equation.

$$\begin{aligned} T_f(0) &= 0 \\ T_f(1) &= 0 \\ T_f(n) &= T_f(n-1) + 1 \end{aligned}$$

This is because there is only one addition made in each recursive call from $f(n; a, b)$ to $f(n-1; b, a+b)$, and there are no additions made in the two cases where $n = 0$ and $n = 1$.

In order to solve this recurrence relation we can expand out the expression and attempt to identify the pattern. This process is shown below.

$$
\begin{aligned}
T_f(n) &= T_f(n-1)+1 \\
&= (T_f(n-2)+1)+1 = T_f(n-2)+2 \\
&= (T_f(n-3)+1)+2 = T_f(n-3)+3 \\
&= \ldots \\
&= (T_f(n-k)+1)+k = T_f(n-k)+k
\end{aligned}
$$

Based on this pattern, we can reach the first base case of this recurrence relation ($T_f(1)$) when $(n-k)=1$, meaning that $k=(n-1)$. Thus, we have the following.

$$
\begin{aligned}
T_f(n) &= T_f(n-(n-1))+(n-1) \\
&= T_f(1)+(n-1) \\
&= 0+(n-1) \\
&= n-1
\end{aligned}
$$

Based on this observation we can see that $T_f(n) \in \Theta(n)$, or simply $T_f(n) = \Theta(n)$.

PROBLEM 3.

Solution.

Base ($n=0$)

When $n=0$, we have the following equality.

$$
\begin{aligned}
L_0(a,b) &= (f(0;a,b), f(1;a,b)) \\
&= (a,b)
\end{aligned}
$$

Induction ($n>0$)

First, we assume that $L^n(a,b) = (f(n;a,b), f(n+1;a,b))$. Now we show that $L^{n+1}(a,b) = (f(n+1;a,b), f(n+2;a,b))$.

$$
\begin{aligned}
L^{n+1}(a,b) &= L(L^n(a,b)) \text{ (by multiplication powers)} \\
&= L(f(n;a,b), f(n+1;a,b)) \text{ (by induction)} \\
&= (f(n+1;a,b), f(n;a,b)+f(n+1;a,b)) \text{ (by definition of } L) \\
&= (f(n+1;a,b), f(n+2;a,b)) \text{ (by Theorem 1)}
\end{aligned}
$$

Thus, $L^{n+1}(a,b) = (f(n+1;a,b), f(n+2;a,b))$, as desired. Therefore, we know that $f(n;a,b) = (L^n(a,b))_1$.

PROBLEM 4.

Solution. TODO: talk about representation, mention multiplication algorithm for logn time, give fibpow listing, list time complexity
TODO: matrix
TODO: repeated squaring
TODO: code here
TODO: time complexity

PROBLEM 5-a. *Write down the definition of pseudo-polynomial time.*

Solution.

Definition 1. Pseudo-polynomial time is the complexity class that encompasses all functions $f(n)$ that run in polynomial time in the numeric value of n, which is exponential in the length of n.

PROBLEM 5-b. *Is fib a pseudo-polynomial time algorithm? Explain.*

Solution.

Definition 2. TODO

PROBLEM 5-c. *Is $fibIt$ a pseudo-polynomial time algorithm? Explain.*

Solution.

Definition 3. TODO

PROBLEM 5-d. *Is $fibPow$ a pseudo-polynomial time algorithm? Explain.*

Solution.

Definition 4. TOOD