Foundations of Algorithms
Homework 2

Arthur Nunes-Harwitt

$$\begin{array}{lcl} F_0 & = & 0 \\ F_1 & = & 1 \\ F_n & = & F_{n-1} + F_{n-2} \end{array}$$

$$\begin{array}{rcl} f(0; a, b) & = & a \\ f(1; a, b) & = & b \\ f(n; a, b) & = & f(n-1; b, a+b) \end{array}$$

Theorem 1 *For any $n \in \mathbb{N}$ if $n > 1$ then $f(n; a, b) = f(n-1; a, b) + f(n-2; a, b)$.*

Theorem 2 *For any $n \in \mathbb{N}$, $F_n = f(n; 0, 1)$.*

Theorem 3 *For any $n \in \mathbb{N}$, $F_n = \frac{1}{\sqrt{5}}(\varphi^n - \hat{\varphi}^n)$, where $\varphi = \frac{1+\sqrt{5}}{2}$ and $\hat{\varphi} = \frac{1-\sqrt{5}}{2}$.*

1. The function `fib` implemented the recurrence F_n. We can characterize the time taken by `fib` in terms of the number of additions (plusses) performed. We can write a recurrence $T_F(n)$ that computes this number as follows. Observe that there are no additions performed for the base cases. Hence $T_F(0) = 0$ and $T_F(1) = 0$. Also observe that the number of additions to compute F_n is one more than the number of additions to compute F_{n-1} together with the number of additions to compute F_{n-2}. Hence $T_F(n) = 1 + T_F(n-1) + T_F(n-2)$. Putting this together, we have the following recurrence.

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

   (a) Prove using the strong form of induction that for any $n \in \mathbb{N}$, $T_F(n) = F_{n+1} - 1$.

   (b) Use limits together with theorem 3 to show that $T_F(n) \in \Theta(\varphi^n)$.

2. The function `fibItHelper` implemented the recurrence $f(n; a, b)$. What is the time complexity of `fibItHelper`? Write down a recurrence relation $T_f(n)$ that characterizes the time complexity in terms of the number of additions (plusses) performed; then solve the recurrence exactly using iteration.

3. Notice that f is repeatedly operating on the numbers a and b.
   Let $L : \mathbb{N}^2 \to \mathbb{N}^2$ be defined by $L(a, b) = (b, a+b)$. Then $f(n; a, b)$ can be understood as $(L^n(a, b))_1$. Prove this assertion by using mathematical induction to prove that for any $n \in \mathbb{N}$, $L^n(a, b) = (f(n; a, b), f(n+1; a, b))$.

4. (**project**) Write a function `fibPow` that takes a natural number n, and returns $(L^n(0,1))_1$.

   (a) First choose a representation for L. (HINT: The variable L is used because the function is a linear operator. Functional programmers beware!)

   (b) Then implement an algorithm to raise objects of that type to the nth power that requires only $\mathcal{O}(\log(n))$ "iterations."

   (c) Finally, implement `fibPow` using the representation of L and the power algorithm.

   (d) What is the asymptotic time complexity of `fibPow`?

5. Look up the definition of *pseudo-polynomial time*.

   (a) Write down the definition.

   (b) Is `fib` a pseudo-polynomial time algorithm? Explain.

   (c) Is `fibIt` a pseudo-polynomial time algorithm? Explain.

   (d) Is `fibPow` a pseudo-polynomial time algorithm? Explain.

6. Solve the following recurrences exactly using the iteration method. In all cases, $T(0) = 0$. Answers should be expressed in terms of $T(n)$.

   (a) $T(n+1) = T(n) + 5$

   (b) $T(n+1) = n + T(n)$

7. Solve the following recurrences exactly using the iteration method. In all cases, $T(0) = 1$. Answers should be expressed in terms of $T(n)$.

   (a) $T(n+1) = 2T(n)$

   (b) $T(n+1) = 2^{n+1} + T(n)$

8. Solve the following recurrences exactly using the iteration method. In all cases, $T(1) = 1$. Answers should be expressed in terms of $T(n)$.

   (a) $T(n) = n + T(n/2)$ (Assume n has the form $n = 2^m$.)

   (b) $T(n) = 1 + T(n/3)$ (Assume n has the form $n = 3^m$.)

9. Solve the following recurrences using the iteration method and express the answer using $\mathcal{O}$-notation. In all cases, $T(1) = 1$, and a, b, and c are constants greater than or equal to one.

   (a) $T(n) = aT(n-1) + bn$ make a distinction between the cases $a = 1$ and $a > 1$.

   (b) $T(n) = aT(n-1) + bn\log(n)$ make a distinction between the cases $a = 1$ and $a > 1$.

   (c) $T(n) = aT(n-1) + bn^c$ make a distinction between the cases $a = 1$ and $a > 1$.

   (d) $T(n) = aT(n/2) + bn^c$ make a distinction between the cases $\frac{2^c}{a} = 1$ and $\frac{2^c}{a} \neq 1$.