# Foundations of Algorithms
# Homework 2

Arthur Nunes-Harwitt

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

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

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

1. 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 performed; then solve the recurrence.
2. 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 $n \in \mathbb{N}$, $L^n(a, b) = (f(n; a, b), f(n+1; a, b))$.
3. (**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 $n$th 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 time complexity of `fibPow`?
4. 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.

5. 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) $T(n) = aT(n-1) + bn$

   (b) $T(n) = aT(n-1) + bn\log(n)$

   (c) $T(n) = aT(n-1) + bn^c$

   (d) $T(n) = aT(n/2) + bn^c$