Graded Exercise 2

Duong Le

Problem 1

a.

Consider the following matrix multiplication:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} a+b & a \\ c+d & c \end{pmatrix}$$

Now, if we plug the identity $F(i)$, $F(i-1)$, $F(i-1)$, and $F(i-2)$ to the above matrix multiplications, we have:

$$\begin{pmatrix} F(i) & F(i-1) \\ F(i-1) & F(i-2) \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} F(i)+F(i-1) & F(i) \\ F(i-1)+F(i-2) & F(i-1) \end{pmatrix} = \begin{pmatrix} F(i+1) & F(i) \\ F(i) & F(i-1) \end{pmatrix}$$

So, by putting the Fibonacci number in to a matrix and multiply it with $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$, we can receive a matrix contains the next Fibonacci numbers. Notice that the first Fibonacci matrix is $\begin{pmatrix} \text{F}(2) & \text{F}(1) \\ \text{F}(1) & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$, and the matrix can only start with $i > 2$, we have the following:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} F(n+1) & F(n) \\ F(n) & F(n-1) \end{pmatrix}$$

To receive the identity required by the questions, we can multiply the resulting matrix with the vector $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$:

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

b.

Base on the previous part, we can make an algorithm to calculate the nth Fibonacci number. The function *product* is the function to calculate matrix multiplication using Strassen's algorithm:

Algorithm 1 $power(A, n)$

```
if (n = 0) then
    return1
else if (n = 1) then
    returnA
else
    if n%2 = 0 then
        k ← n/2
        A^k ← power(A, k)
        product(A^k, A^k, 2)
    else
        k ← (n−1)/2
        A^k ← power(A, k)
        intermediate ← product(A^k, A^k, 2)
        product(intermediate, A, 2)
    end if
end if
```

Algorithm 2 $fib(n)$

```
if (n = 0) then
    fib(n) ← 0
else if (n ≤ 2) then
    fib(n) ← 1
else
    A ← (1 1; 1 0)
    B ← power(A, n)
    return B(0, 0)
end if
```

c.

Consider, incase n is odd, we need 14 arithmetic operations to combine the subproblems (There are two calls to the *product* method). So, we have:

$$T(n) = T(\frac{n}{2}) + \Theta(14)$$

Applying Master Theorem, we have:

$$\begin{align} c_{crit} &= log_2 1 = 0 \tag{1} \\ \Rightarrow T(n) &= \Theta(n^0 \log n) = \Theta(\log n) \tag{2} \end{align}$$

Now, if n is even, we need 7 arithmetic operations to combine the subproblems. Again, we have:

$$T(n) = T(\frac{n}{2}) + \Theta(7)$$

Applying the Master Theorem, we also have $T(n) = \Theta(\log n)$
$\Rightarrow$ The time complexity of the algorithm is $\Theta(\log n)$

Problem 2

a.

We have: Consider the following identities, with $n \in \mathbb{N}$:

$$\binom{n}{0} = \binom{n}{n} = 1$$

Consider next, for $k \in \mathbb{N}$, and $k < n$:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

We can define a function $binomial(n, k)$:

Algorithm 3 $binomial(n, k)$

if $(n = k) \mid\mid (k = 0)$ **then**
  $binomial(n, k) \leftarrow 1$
else
  $binomial(n, k) \leftarrow (binomial(n-1, k-1) + binomial(n-1, k))$
end if

b.

We have, the recurrance relation of the algorithm is:

$$T(n, k) = T(n-1, k-1) + T(n-1, k) + O(1)$$

c.

We have, from the previous part, where O(1) is the time to sum up the numbers:

$$\begin{aligned}
T(n,k) &= T(n-1,k-1) + T(n-1,k) + O(1) && (3)\\
&= T(n-2,k-2) + 2T(n-2,k-1) + T(n-2,k) + 3O(1) && (4)\\
&= T(n-3,k-3) + 3T(n-3,k-2) + 3T(n-3,k-1) + T(n-3,k) + 5O(1) && (5)\\
&= ... && (6)
\end{aligned}$$

Consider that for each steps, the number of summands is doubled, and we need 2^n arithmetic operators to sum up the sub problem. So we get the following:

$$\begin{aligned}
T(n,k) &\leq O(2^n) + O(2^n) && (7)\\
\Leftrightarrow T(n,k) &= O(2^n) && (8)
\end{aligned}$$

Consider $n \geq 4$:

$$2^n < 1 * n!$$

$\Rightarrow$ By definition, the function has a time complexity of $o(n!)$