Homework 2

Foundations of Algorithms

Divesh Badod

1. Time complexity of $fibItHelper$ is $O(n) = n$.
   The recurrence relation in terms of addition is given by:
   $T_f(0) = 0$
   $T_f(1) = 1$
   $T_f(n) = T_f(n-1) + c$ c = Constant
   Solving by iteration
   $T_f(n) = T_f(n-1) + c$
   $= (T_f(n-2) + c) + c$
   $= (T_f(n-3) + c) + c + c$
   $= T_f(n-k) + c \ldots + c$ ←Identifying the pattern
   $= T_f(n-k) + kc$
   Let $k = n$,
   $T_f(n) = T_f(n-n) + nc$
   $= T_f(0) + nc$
   $= 0 + nc$
   $= nc$

2. Proof: - By Mathematical Induction
   - Observer that when $n = 1$ we have
     $L(a,b) = (b, a+b)$.
     $L(a,b) = \big(f(1;a,b), f(0;a,b) + f(1;a,b)\big)$.
     $L(a,b) = \big(f(1;a,b), f(2;a,b)\big)$.
   - Assume for $n = k$
     $L^k(a,b) = \big(f(k;a,b), f(k+1;a,b)\big)$.

     $L^{k+1}(a,b) = L\left(L^k(a,b)\right)$.
     $L^{k+1}(a,b) = L\big(f(k;a,b), f(k+1;a,b)\big)$.
     $L^{k+1}(a,b) = \big(f(k+1;a,b), f(k;a,b) + f(k+1;a,b)\big)$.
     $L^{k+1}(a,b) = \big(f(k+1;a,b), f(k+2;a,b)\big)$.

3. a b c Electronic Submission
   d. Time complexity of $fibPow$ is $O(\log n)$

4. a. An algorithm runs in pseudo-polynomial time if its running time is a polynomial in the numeric value of the input but not necessarily in the length of the input. In general, the numeric value of the input is exponential in the input length, which is why a pseudo-polynomial time algorithm does not necessarily run in polynomial time with respect to the input length.

   b. Time complexity of fib is $O(2^n)$ – not polynomial in time. Larger inputs take a large amount of time. Hence it is not a pseudo-polynomial time algorithm.

   c. Time complexity of $fibIt$ is $O(n)$ – not polynomial in time. Larger inputs require the same amount of bits to represent the input. Hence it is not a pseudo-polynomial time algorithm.

   d. Time complexity of $fibPow$ is $O(\log n)$ – logarithmic in time. For larger inputs it is sub-linear in time but it would be impractical for the algorithm to represent the inputs in terms of length because it is exponential in terms of length. Hence it is a pseudo-polynomial time algorithm.

5. a.
$$T(0) = 0$$
$$T(x) = T(x-1) + 5$$
$$\begin{aligned} T(n) &= T(n-1) + 5 \\ &= (T(n-2) + 5) + 5 \\ &= \big((T(n-3) + 5) + 5\big) + 5 \\ &= T(n-k) + 5 \ldots + 5 \leftarrow \text{Identify the pattern} \\ &= T(n-k) + 5k \end{aligned}$$
Let $k = n$,
$$\begin{aligned} T(n) &= T(n-n) + 5n \\ &= T(0) + 5n \\ &= 0 + 5n \\ &= 5n \end{aligned}$$

   b.
$$T(0) = 0$$
$$T(x) = x - 1 + T(x-1)$$
$$\begin{aligned} T(n) &= n - 1 + T(n-1) \\ &= n - 1 + \big(n - 1 + T(n-2)\big) \\ &= 2n - 2 + T(n-2) \\ &= 2n - 2 + \big(n - 1 + T(n-3)\big) \\ &= 3n - 3 + T(n-3) \leftarrow \text{Identify the pattern} \\ &= kn - k + T(n-k) \end{aligned}$$
Let $k = n$,
$$\begin{aligned} T(n) &= n^2 - n + T(n-n) \\ &= n(n+1) + T(0) \\ &= n(n+1) + 0 \\ &= n(n+1) \end{aligned}$$

6. a. $T(1) = 1$

$T(x) = 2T(x-1)$

$T(n) = 2T(n-1)$

$= 2\big(2T(n-2)\big)$

$= 2\Big(2\big(2T(n-3)\big)\Big)$ ←Identify the pattern

$= 2^k T(n-k)$

Let $k = n-1$,

$T(n) = 2^{n-1}T\big(n-(n-1)\big)$

$= 2^{n-1}T(1)$

$= 2^{n-1}1$

$= 2^{n-1}$

b. $T(1) = 1$

$T(n) = n + T\left(\frac{n}{2}\right)$

$T(x) = x + T\left(\frac{x}{2}\right)$

Assuming $n = 2^m$

$T(2^m) = 2^m + T\left(\frac{2^m}{2}\right)$

$= 2^m + T(2^{m-1})$

$= 2^m + (2^{m-1} + T(2^{m-2}))$

$= 2^m + (2^{m-1} + (2^{m-2} + T(2^{m-3})))$ ←Identify the pattern

$= 2^m + \cdots 2^{m-k+1} + T(2^{m-k})$

Let $k = m$,

$T(2^m) = 2^m + \cdots 2^{m-m+1} + T(2^0)$

$= 2^m + \cdots 2^1 + T(1)$

$= 2^m + \cdots 2^1 + 1$

$= 2^m + \cdots 2^1 + 2^0$

$= \sum_{i=0}^{m} 2^i$

$= 2^{m+1} - 1$

$= 2^m . 2 - 1$

$T(n) = 2n - 1$

c. $T(1) = 1$

$T(n) = 1 + T\left(\frac{n}{3}\right)$

$T(x) = 1 + T\left(\frac{x}{3}\right)$

Assuming $n = 3^m$

$$T(3^m) = 1 + T\left(\frac{3^m}{3}\right)$$
$$= 1 + T(3^{m-1})$$
$$= 1 + (1 + T(3^{m-2}))$$
$$= 1 + (1 + (1 + T(3^{m-3}))) \quad \leftarrow \text{Identify the pattern}$$
$$= 1 + \cdots 1 + T(3^{m-k})$$
$$= k + T(3^{m-k})$$

Let $k = m$,

$T(3^m) = m + T(3^0)$
$= m + T(1)$
$= m + 1$
$T(n) = \log_3 n + 1$

7. a. $T(1) = 1$

$T(n) = aT(n-1) + bn$
$= a\big(aT(n-2) + b(n-1)\big) + bn$
$= a\left(a\big(aT(n-3) + b(n-2)\big) + b(n-1)\right) + bn$
$= a^3T(n-3) + a^2b(n-2) + ab(n-1) + bn \leftarrow$Identify the pattern
$= a^kT(n-k) + \sum_{i=0}^{k-1} a^i b(n-i)$

Let $k = n - 1$

$T(n) = a^{n-1}T(1) + \sum_{i=0}^{n-2} a^i b(n-i)$
$= a^{n-1} + bn + \cdots + a^{n-2}b\big(n-(n-2)\big)$
$= a^{n-1} + bn + \cdots + a^{n-2}b2$

So, $O(a^n)$ if $a > 1$ and $O(n)$ if $a < 1$

b. $T(1) = 1$

$T(n) = aT(n-1) + bn\log n$
$= a(aT(n-2) + (b(n-1)\log(n-1))) + bn\log n$
$= a(a(aT(n-3) + b(n-2)\log(n-2)) + b(n-1)\log(n-1)) + bn$
$= a^3T(n-3) + a^2b(n-2)\log(n-2) + ab(n-1)\log(n-1) + bn\log n \leftarrow$Identify the pattern
$= a^kT(n-k) + \sum_{i=0}^{k-1} a^i b(n-i)\log(n-i)$

Let $k = n - 1$

$T(n) = a^{n-1}T(1) + \sum_{i=0}^{n-2} a^i b(n-i)\log(n-i)$
$= a^{n-1} + bn + \cdots + a^{n-2}b2\log n$

$O(a^n)$ if $a > 1$ and $O(n\log n)$ if $a < 1$

c. $T(1) = 1$

$$T(n) = aT(n-1) + bn^c$$
$$= a(aT(n-2) + b(n-1)^c) + bn^c$$
$$= a(a(aT(n-3) + b(n-2)^c) + b(n-1)^c) + bn^c$$
$$= a^3T(n-3) + a^2b(n-2)^c + ab(n-1)^c + bn^c \leftarrow \text{Identify the pattern}$$
$$= a^kT(n-k) + \sum_{i=0}^{k-1} a^i b(n-i)^c$$

Let $k = n - 1$

$$T(n) = a^{n-1}T(1) + \sum_{i=0}^{n-2} a^i b(n-i)$$
$$= a^{n-1} + bn + \cdots + a^{n-2}b\big(n-(n-2)\big)^c$$
$$= a^{n-1} + bn + \cdots + a^{n-2}b2^c$$

$O(a^n)$ if $a > 1$ and $O(n^c)$ if $a < 1$

d. $T(1) = 1$

$$T(n) = aT\left(\frac{n}{2}\right) + bn^c$$
$$= a\left(aT\left(\frac{n}{4}\right) + b\left(\frac{n}{2}\right)^c\right) + bn^c$$
$$= a\left(a\left(aT\left(\frac{n}{8}\right) + b\left(\frac{n}{4}\right)^c\right) + b\left(\frac{n}{2}\right)^c\right) + bn^c$$
$$= a^3T\left(\frac{n}{8}\right) + a^2b\left(\frac{n}{4}\right)^c + ab\left(\frac{n}{2}\right)^c + bn^c \leftarrow \text{Identify the pattern}$$
$$= a^kT\left(\frac{n}{2^k}\right) + \sum_{k=1}^{n} a^k\left(\frac{n}{2^k}\right)b(n-(k-1))^c$$

Let $k = \log n$

$$T(n) = a^{\log n}T(1) + \sum_{k=1}^{\log n} a^{\log n}\left(\frac{n}{2^{\log n}}\right) + b(n-(\log n - 1))^c$$

$O(n^c)$