Homework 4
M1522.000900 Data Structure (2019 Fall)

2013-12815 Dongjoo Lee

1 Q1

For a binary tree that has n leaves, the number of 2-degree nodes be $n-1$.
In the tree that has k nodes, let the number of leaves be $L(k)$ and the number of 2-degree nodes be $T(k)$ Let the proposition $P(k)$: the binary tree that has k nodes, $L(k) - T(k) = 1$.
To proove this by induction,

Proof. **Base case: At** $k = 1$, $L(k) = 1$ **and** $T(k) = 0$.
$L(k) - T(k) = 1$**, so,** $P(k)$ **holds for** $k = 1$.

Inductive Step: Assume that $P(k)$ **holds for** $k = n$.
When tree grows by 1 node, in terms of be-attatched node there are 3 cases by degree.
Case 1: The new node cannot be attatched to existing node if it is 2-degree node.
Case 2: The new node can be attatched to existing node if it is 1-degree node. And this node becomes 2-degree node. In this case, $= T(n+1) = T(n)+1$ **and** $L(n+1) = L(n) + 1$. **So,** $L(n+1) - T(n+1) = L(n) - T(n) = 1$.
Case 3: The new node can be attatched to existing node if it is leaf. In this case, there is no change in $T(n+1)$ **and** $L(n+1)$. **So,** $L(n+1) - T(n+1) = L(n) - T(n) = 1$.

In all cases, $L(n+1) - T(n+1) = 1$. $\therefore P(n+1)$ **holds for** $k = n+1$.

QED □

2 Q2

(1) $f(n) = logn^2 = 2logn$
$g(n) = logn + 5$
For $n_0 = 10^6$ and $c = 1$, $f(n) \geq c \cdot g(n)$, $\forall n > n_0$
$\therefore f(n) = \Omega(g(n))$
And for $n_0 = 0$ and $c = 2$, $f(n) \leq c \cdot g(n)$, $\forall n > n_0$
$\therefore f(n) = O(g(n))$
$\therefore \mathbf{f(n)} = \theta(\mathbf{g(n)})$

(2) $f(n) = \sqrt{n} = n^{\frac{1}{2}}$
$g(n) = logn^2$
For $n_0 = 10^2$ and $c = 1$, $f(n) \geq c \cdot g(n)$, $\forall n > n_0$
$\therefore \mathbf{f(n)} = \mathbf{\Omega}(\mathbf{g(n)})$

(3) $f(n) = n$
$g(n) = log^2 n = (logn)^2$
At $n > 10$, $logn < n^{\frac{1}{2}}$, thus, $(logn)^2 < (n^{\frac{1}{2}})^2 = n$
For $n_0 = 10$, and $c = 1$, $f(n) \geq c \cdot g(n)$, $\forall n > n_0$
$\therefore \mathbf{f(n) = \Omega(g(n))}$

(4) $f(n) = logn^2 = 2logn$
$g(n) = log^2 n = (logn)^2$
At $n > 10$, $logn > 1$ and $(logn)^2 > logn$
For $n_0 = 10$ and $c = 2$, $f(n) \leq c \cdot g(n)$, $\forall n > n_0$
$\therefore \mathbf{f(n) = O(g(n))}$

3 Q3

(1) (a) Before the 1st for loop, time complexity is $\theta(1)$. For the 1st for loop, all the comparison and calculation and assignment occur constant times at each repetitions. So, 1st for loop's time complexity is $\theta(N)$. Similarly, 2nd for loop's time complexity is $\theta(M)$.
$\therefore$ **Total time complexity is $\theta(\mathbf{N + M})$.**

(b) Before the 1st for loop, memory allocating occurs for a and b. So, its space complexity is $\theta(1)$. For the 1st for loop, memory allocating occurs only for i and a at each repetitions. So, its space complexity is $\theta(1)$. Similarly, 2nd for loop's space complexity is also $\theta(1)$.
$\therefore$ **Total space complexity is $\theta(\mathbf{1})$.**

(2) Before the 1st for loop, time complexity is $\theta(1)$. For the outer for loop, i increases by 1, so it is repeated $c_1 \cdot n$ times. For the inner for loop, j increases in $2^1, 2^2, \cdots, 2^(lgn)$, so it is repeated $c_2 \cdot lgn$ times. Therefore, the whole repetition occurs $c_1 \cdot c_2 \cdot n \cdot lgn$ times.
$\therefore$ **Total time complexity is $\theta(\mathbf{n \cdot lgn})$.**

(3) An algorithm X is asymptotically more efficient than Y means X will always be a better choice for large inputs.
$\therefore$ **(b)**

4 Q4

(1) $T(n) + 1 = 3 \cdot (T(n-1) + 1)$
$T(n) + 1 = 3^{n-1} \cdot (T(0) + 1)$
$\therefore T(n) = 3^{n-1} \cdot (T(0) + 1) - 1 = 3^{n-1} \cdot T(0) + 3^{n-1} - 1$
Thus, for $c = T(0) + 1$ and $n_0 = 0$, $T(n) \leq c \cdot 3^{n-1}$, $\forall n > n_0$
$\therefore \mathbf{T(n) = O(3^n)}$

(2) Let's assume that the new machine Y has similar algorithm with X. X takes t seconds for n inputs. As Y is 27 times faster than X, Y takes $\frac{1}{27}t$ for the same n inputs.

$c \cdot 3^{n-1} = 27 \cdot c' \cdot 3^{n-1} = t$
$c' = \frac{1}{27}c$
$\therefore$ Time complexity of Y, $Y(n) = c \cdot 3^{n-4}$
By the result, **Y can handle $\mathbf{n+3}$ inputs** while X handles n inputs in the same t.

5 Q5

(1) In the function powerN, comparison occurs 1 time in the 1st line. At the 2nd line, function call occurs. If time complexity of function in Figure 1. is $T(n)$, then, $T(n) = 1 + T(n-1)$

$\therefore$ If n is large enough, $\mathbf{T(n)} = \theta(\mathbf{n})$.

(2) If time complexity of function in Figure 2. is $S(n)$, then,

$$S(n) = \left\{\begin{matrix} 1 & \text{for } n = 0 \\ 1 + S(\frac{n}{2}) & \text{for } n = 2^k \\ 1 + S(\frac{n-1}{2}) & \text{for } n = 2^k - 1 \end{matrix}\right\}$$

$\therefore$ If n is large enough, $\mathbf{S(n)} = \theta(\mathbf{lgn})$.