COMP4500/7500 Advanced Algorithms & Data Structures
Sample Solution to Tutorial Exercise 1 (2014/2)*

School of Information Technology and Electrical Engineering, University of Queensland

August 5, 2014

1. Consider a binary tree structure defined using a Java class definition like:

```
class TreeNode {
  String    info;
  TreeNode  left;
  TreeNode  right;
}
```

Here `info` contains the data item in the node; `left` and `right` are "pointers" to the left and right subtrees respectively. If a subtree is empty then its corresponding pointer is `null`.

Give **recursive** procedures for the following problems in pseudocode or Java-like pseudocode. Carefully explain any assumptions you need to make.

The Java environment is not well-defined; to answer this question in Java requires knowledge of the available methods and class variables. For example, with a traversal method we could piggy-back on it easily to count the number of nodes. So the solutions here use pseudocode like that used in the text.

(a) Determine the size of (number of nodes in) a tree. (Remember to deal correctly with the case of an empty tree, that has no nodes.)

Sample solution.

```
SIZE(T)
if T == NULL
    return 0
else
    return SIZE(T.left) + SIZE(T.right) + 1
```

In Java, if the methods are written within `TreeNode` that will handle the case of a non-empty tree. A clean way to extend the methods to handle empty trees is to have a top-level class `Tree` and have `TreeNode` as a sub-class that represents non-empty trees and a second sub-class `EmptyTree` that represents an empty tree. The SIZE method of `EmptyTree` would return 0.

(b) Determine the height of a tree.

Sample solution.

```
HEIGHT(T)
if T == NULL
    return 0
else
    return MAX(HEIGHT(T.left), HEIGHT(T.right)) + 1
```

This assumes that the HEIGHT of an empty tree is 0 and that a function max is suitably defined.

(c) Determine whether a value given as a parameter is a member of a tree, assuming the tree is ordered as a binary search tree.

Sample solution.

MEMBERBST(T, E)
if T == NULL
  return FALSE
elseif E == $T.\mathit{info}$
  return TRUE
elseif $E > T.\mathit{info}$
  return MEMBERBST$(T.\mathit{right}, E)$
else
  return MEMBERBST$(T.\mathit{left}, E)$

This assumes that == and $>$ are suitably defined on the type of E.

(d) Determine whether a value is a member of an *unordered* tree.

Sample solution.

MEMBER(T, E)
return $T \neq$ NULL and (E == $T.\mathit{info}$ or MEMBER$(T.\mathit{right}, E)$ or MEMBER$(T.\mathit{left}, E)$)

An alternative approach is to use **if** statements.

2. (CLRS Exercise 3.2-2, p60 (3rd), p57 (2nd), CLR Exercise 2.2-3, p37 (1st))
Prove $a^{\log_b n} = n^{\log_b a}$.

Remember that the logarithm (base b) for any positive y, i.e., $\log_b y$, is defined to be the power to which b is raised to get y, so by definition $b^{\log_b y} = y$. You may also use the fact that $(x^a)^b = x^{(a \cdot b)}$.

Sample solution.

$$\begin{aligned}
a^{\log_b n} &= (b^{\log_b a})^{\log_b n} && \text{— } b^{\log_b a} = a \\
&= b^{(\log_b a)\cdot(\log_b n)} && \text{— prop. powers} \\
&= b^{(\log_b n)\cdot(\log_b a)} && \text{— mult. commutative} \\
&= (b^{\log_b n})^{\log_b a} && \text{— prop. powers} \\
&= n^{\log_b a} && \text{— } b^{\log_b n} = n
\end{aligned}$$

3. Use mathematical induction on n to prove the following linearity property of summations for all $n \geq 0$.

$$\sum_{k=1}^{n} (ca_k + b_k) = c \sum_{k=1}^{n} a_k + \sum_{k=1}^{n} b_k$$

Note that, by definition, for any term t_k: $\sum_{k=1}^{0} t_k = 0$.

Sample solution. Base case: for $n = 0$ the summations are all 0. Hence

$$\begin{aligned}
\textstyle\sum_{k=1}^{0} (ca_k + b_k) &= 0 \\
&= c \cdot 0 + 0 \\
&= c \textstyle\sum_{k=1}^{0} a_k + \sum_{k=1}^{0} b_k
\end{aligned}$$

Inductive case: assume the property holds for $n = m$ and prove it holds for $n = m + 1$.

$$\begin{aligned}
\textstyle\sum_{k=1}^{m+1} (ca_k + b_k) &= (ca_{m+1} + b_{m+1}) + \textstyle\sum_{k=1}^{m} (ca_k + b_k) && \text{— prop. summation} \\
&= (ca_{m+1} + b_{m+1}) + c \textstyle\sum_{k=1}^{m} a_k + \sum_{k=1}^{m} b_k && \text{— ind. hypothesis} \\
&= c(a_{m+1} + \textstyle\sum_{k=1}^{m} a_k) + b_{m+1} + \sum_{k=1}^{m} b_k && \text{— regrouping} \\
&= c(\textstyle\sum_{k=1}^{m+1} a_k) + \sum_{k=1}^{m+1} b_k && \text{— prop. summation (twice)}
\end{aligned}$$

4. Use induction to prove the following property of arithmetic series for all $n \geq 0$.

$$\sum_{k=1}^{n} k = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

Sample solution. Base case: for $n = 0$ the summation is 0. Hence $\sum_{k=1}^{0} k = 0 = \frac{0(0+1)}{2}$.

Inductive case: assume the property holds for $n = m$ and prove it holds for $n = m + 1$.

$$\begin{array}{lcll} \sum_{k=1}^{m+1} k & = & (m+1) + \sum_{k=1}^{m} k & \text{— prop. summation} \\ & = & (m+1) + \frac{m(m+1)}{2} & \text{— ind. hypothesis} \\ & = & \frac{2(m+1)+m(m+1)}{2} & \text{— putting } m+1 \text{ over 2} \\ & = & \frac{(m+1)(m+2)}{2} & \text{— factoring } m+1 \end{array}$$

5. (CLRS Exercise A.1-1, p1149 (3rd), p1062 (2nd), CLR Exercise 3.1-1, p45 (1st))

   Find a simple formula for: $\displaystyle\sum_{k=1}^{n}(2k-1)$.

   Sample solution. Using the linearity property (see Question 3 above) we can rewrite the summation to use an arithmetic series (see Question 4).

$$\begin{array}{lcll} \sum_{k=1}^{n}(2k-1) & = & 2\sum_{k=1}^{n} k - \sum_{k=1}^{n} 1 & \text{— linearity} \\ & = & 2\frac{n(n+1)}{2} - \sum_{k=1}^{n} 1 & \text{— arithmetic series} \\ & = & n(n+1) - n \cdot 1 & \text{— constant summation} \\ & = & (n^2+n) - n & \\ & = & n^2 & \end{array}$$

6. Use induction to prove the following property of geometric (or exponential) series, for all $n \geq 0$ and real $x \neq 1$.

$$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n = \frac{x^{n+1}-1}{x-1}$$

   For $|x| < 1$, show the following holds when the summation is infinite.

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$$

   An infinite summation is defined to be the limit as n tends to infinity of the corresponding finite summation.

$$\sum_{k=0}^{\infty} t_k = \lim_{n\to\infty} \sum_{k=0}^{n} t_k$$

   Sample solution. Throughout the following we assume $x \neq 1$ (as given). We start with the finite sum.

   Base case: for $n = 0$, $\sum_{k=0}^{0} x^k = x^0 = 1 = \frac{x^{0+1}-1}{x-1}$, as $x \neq 1$.

   Inductive case: assume the property holds for $n = m$ and prove it holds for $n = m + 1$.

$$\begin{array}{lcll} \sum_{k=0}^{m+1} x^k & = & x^{m+1} + \sum_{k=0}^{m} x^k & \text{— prop. summation} \\ & = & x^{m+1} + \frac{x^{m+1}-1}{x-1} & \text{— ind. hypothesis} \\ & = & \frac{(x-1)x^{m+1}+x^{m+1}-1}{x-1} & \text{— putting } x^{m+1} \text{ over } x-1 \\ & = & \frac{x^{(m+1)+1}-x^{m+1}+x^{m+1}-1}{x-1} & \text{— expanding } (x-1)x^{m+1} \\ & = & \frac{x^{(m+1)+1}-1}{x-1} & \end{array}$$

   An infinite summation is defined to be the limit of the finite summations to n, as n tends to infinity.

$$\begin{array}{lcll} \sum_{k=0}^{\infty} x^k & = & \lim_{n\to\infty} \sum_{k=0}^{n} x^k & \text{— infinite sum} \\ & = & \lim_{n\to\infty} \frac{x^{n+1}-1}{x-1} & \text{— geometric sum} \\ & = & \frac{0-1}{x-1} & \text{— } \lim_{n\to\infty} x^{n+1} = 0 \text{ for } |x| < 1 \\ & = & \frac{1}{1-x} & \end{array}$$