COMP4500/7500 Advanced Algorithms & Data Structures
Tutorial Exercises 1 (2014/2)*

School of Information Technology and Electrical Engineering, University of Queensland

August 5, 2014

Background information

As in most courses, the best way to learn new material is to work on examples that require you to exercise your new knowledge. To this end attempting the revision exercises is essential to making progress in the subject. It is also good preparation for the assignments and examinations.
This first set contains revision exercises on recursion and the mathematical notations that are used in the course. A treatment of some mathematical foundations may be found in Appendix A of the textbook. (Additional coverage is provided in CLRS, Chapter 3.) You may be a bit rusty on this material, but it should come back to you with a bit of practice.

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.

   (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.)

   (b) Determine the height of a tree.

   (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.

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

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 $(x^a)^b = x^{(a \cdot b)}$.

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: $\sum_{k=1}^{0} t_k = 0$.

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

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

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

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

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.

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

7. (Programming problem, advance notice)
   The following problem is included in the Revision Material 2. It may be a good idea to start work on it now. You might be interested in extending the practical analysis to other sorting algorithms.

   Algorithms for both insertion sort and merge sort have been studied. You may use any implementations that you wish.

   Perform an empirical analysis of these algorithms by running them in order to determine their execution time for different inputs (eg, the best-case and worst-case inputs for insertion sort) and different sizes of input (eg, 1000, 2000, 4000). Compare your results with the results of the theoretical analysis of these algorithms. (You may find it useful to graph your experimental results.)

   You may also like to experiment with profiling the execution of the algorithms.