COMP114 - Assessment 3
Semester 2 – 2011
Binary Tree Properties

Assessment Information

Assessment Number	3
Contribution to Overall Mark	30%
Submission deadline	*Friday 6th May 2011, 16.00*

Relevant Learning Outcomes for this Assessment

1	Awareness of different areas of CS for which experimental methods are relevant.
4	Ability to select and apply appropriate experimental methods.
5	Ability objectively to assess, analyse, and present experimental results.

Assessment Description – Overview

The aim of this assessment is to consider how properties of random binary trees vary depending on the method used to generate these. While there is a minimum group of properties that ought to be examined, the manner in which these are reviewed and the nature of additional properties is deliberately left open-ended. Some suggestions are offered at the end of the detailed description.

Assessment Description – Detail

The lectures introduced a class of stuctures called *Binary Trees* and discussed two approaches that could be used to generate random binary trees.

The module web pages provide *two* Java classes which should be copied to your local filestore.

BinaryTree.class
RandomTrees.class

The first of these provides a realisation of the BinaryTree class exactly as described in the module notes, including the seven instance methods – IsLeafNode() and the pairs of methods to Set and Get each of the three fields.

The second class provides implementations of *three* random generation methods – two of which were outlined in the lectures –

Method Name	Description
public static BinaryTree RootDown(**int** n)	Returns an n-leaf random binary tree using the recursive RootDown approach.
public static BinaryTree LeafUp(**int** n)	Returns an n-leaf random binary tree using the iterative LeftUp approach.
public static BinaryTree UniformTree(**int** n)	Returns an n-leaf random binary tree with each possible n-leaf binary tree being *equally likely*.

For those of you who would like to see how all of these methods are implemented in Java, the source code is also available in RandomTrees.java (also on the module web pages). As mentioned in the lectures the Uniform generation method is *non-trivial*, and you should not feel concerned if you don't follow how it works.

In this assessment you are asked to compare the properties of random binary trees produced by these methods. In particular, is to examine the following:

a. How does the average *depth* of the trees generated by each method compare to the number of leaf nodes?

b. What are the *maximum* and *minimum* depths seen with each size and method?

c. Are you able to form any estimates of *how quickly* any of these averages increases in terms of the *number* of leaf nodes n? If, as was claimed, in the lectures, the methods RootDown and LeafUp are biased to "shallow" trees, then the average depth (as n increases) should be "roughly" log n. How could this be tested using the results found? (**Hint:** How does the value $AverageDepth(n)$/Math.log((double)n) change?)

(Note: recall that the package BasicStats described in Assessment 2, provides a method which computes the average value of an array of m integer values).

In order to answer (a) and (b), you will need to implement (in Java) a method for computing the depth of a given n-leaf binary tree, T, such as the one outlined in the lecture notes, i.e. the approach in Algorithm 1 below.

In addition, for (a) and (b) you will have to decide the following:

1. What range of n (the number of leaf nodes) to use (smallest value?, highest value?, gap between successive values?)

2. How many trees should be generated for each n and with each method. (10? 20?, 50?, some function of n itself?)

Algorithm 1 Algorithm for finding depth of binary tree

```
Depth(BinaryTree T)
if T is a single leaf node then
  return 1
else
  return 1 + Maximum of {Depth(T.Left), Depth(T.Right)}
end if
```

What should be Submitted

a. The *source* code of the Java program through which your experiments are carried out.

b. A summary of the binary tree properties examined in your experiments.

c. A report (in the form of a table) presenting your findings for each of the tree measures considered. These tables should be organised in the form,

Table 1: Binary Tree property, e.g. Average Depth

n (Number of leaves)	RootDown	LeafUp	Uniform

d. A summary of your conclusions from the experiment.

e. A completed and signed *Declaration On Plagiarism and Collusion Form.*

How the work should be submitted

Items (a–e) should handed in to the *Student Office.* A cover sheet should indicate all of the following information:

a. The Assessment *number.*

b. Your **name** and University **e-mail address**

c. Your lab group.

d. The name of the *demonstrator* responsible for this group.

e. Your degree programme, e.g. G400 Computer Science, G500 Computer INformation Systems.