

CAS CS 112 – Spring 2008, Assignment 4

Due at 10:00 pm on Thursday, March 27

Visualizing Binary Trees

In this assignment, we will augment the binary tree classes presented in the textbook with routines to create large binary trees by reading input from the user, and routines to print out binary trees in human-readable format. We will write routines to print the tree using two methods, one a text-based method described below; and the second, a graphical method that draws the trees using lines and circles on a canvas (similar in spirit to questions 4.38 - 4.40 on p. 163 in our textbook).

The assignment consists of the following components:

1. Start from the Binary Search Tree and AVL Tree implementation of Chapter 4.3. Recall that source code in the textbook is available at:

<http://www.cs.fiu.edu/~weiss/dsaajava2/code/>.

2. Augment these classes to support bulk insertion of integer values. Insertions will be specified in the form of a text string in JSON (JavaScript Object Notation). The JSON string “[12, 15, 76, 23, 99]” should result in an insertion of 12, followed by an insertion of 15, etc. Implement this as a member function `void BulkInsert(String s)`.
3. Next, augment these classes to support textual printing of the values in the trees. In order to accomplish this, we will logically assign an (x, y) coordinate to each tree node. Following question 4.38 in our text, it is appropriate for the x-coordinate of the node to be proportional to the inorder traversal number of the node in the tree, and for the y-coordinate of the node to be proportional to the (negative of) the depth of the node in the tree. Therefore, we will need member variables to store, and routines to compute the inorder traversal number and depth of each node in the tree (you may choose what to name these). Then write a member routine `void PrintAsText()` that prints a BST or AVL tree to screen. For example, printing the BST resulting from the bulk insertion above should look something like:

```
12
 15
   76
  23 99
```

4. Now, write a member routine `void PrintOnCanvas()` that prints a tree on a graphical canvas by making calls to graph-assembling functions `DrawCircle(x, y, value)` and `DrawLine(i, j)` as described in Question 4.39 in our text. We will provide a package which implements these functions and demonstrate how your program should integrate with this package in lab during the week after Spring Break.

5. Finally, write a main function that tests the routines you have written. Specifically, your code should repeatedly query the user for a JSON string corresponding to a sequence of insertions, create both a BST and an AVL tree resulting from that sequence of insertions, and print these trees both as text and on canvas. Prompt the user to clear the screen and the canvasses before processing the next string.

Submit your code in three files called `BST.java`, `AVL.java` and `driver.java`. Note that the routines you are asked to furnish are identical for both binary search trees and AVL trees; extra credit will be provided if you are able to restructure the provided implementations so that binary search trees are a base class that implement the new routines, and AVL trees are a derived class of BSTs that inherit the new routines.