Homework Assignment 5

The dictionary search done in Lab Assignment 9 using a TreeMap shows that a TreeMap is a very useful structure. In this assignment, you are to implement your own binary search tree for doing lookup as if done by a TreeMap.

The web page provides you with the same input data as for the lab assignment, as well as just the abstract of the paper as small input that you can use to test your code. In addition, the web page provides you with a sorted version of the words, with duplicates removed, of the abstract. The big difference between the abstract as written in the paper and the sorted version of the abstract is that if you implement a binary search tree with data that is already in sorted order, you will get a worst-case binary search tree that is just one long list extending down.

You are to use a Record class similar to the Record class used in so many other of the assignments in this class. In this case, your Record need have nothing as the data payload other than a String that is the data element. An Interface for the Record can be found on the web page.

You are to write a binary search tree class that implements the Interface named IBST.java that is linked off the web page.

Your tree should be a tree of instances of a BTNode class that implements the Interface named IBTNode.java that is linked off the web page.

Note the sample output. If we define the height of the root to be 0, and then the height of any node to be 1 larger than the height of its parent, then every node has a height value that can be assigned to it. In a perfectly balanced tree, the maximum height of a tree of \( N \) nodes should be \( \log N \) and there should be \( 2^h \) nodes of height \( h \). You are to write code that will dump the tree in inorder fashion, which for a binary search tree should be in perfectly sorted order. (It’s not sufficient for being correct that you get things in sorted order when you print in inorder fashion, but it is necessary that this be true.) As you traverse the BST in inorder fashion, you are to compute the height of every node and to print out any new maximum height that you encounter. You must also at the end of this traversal be able to print out the number of nodes and the average height as well as a frequency count of the number of nodes of any given height. This last can be done with an array, and you are free to dimension this array fixed at 100 elements.

You do not need to dump the tree in preorder or postorder fashion. I did this just to make sure my code was correct.