University of Illinois at Urbana-Champaign
Department of Computer Science

MP6: Trees – Word Counting, Part 2

CS 225 Data Structures
Spring Semester, 1999

Handed out: Friday, March 12, 1999
Due date: Saturday, March 27, 1999 at 5:00 PM

1 Introduction

In this MP, you will continue the work you started in MP5 by adding some additional features to the classes you were using. This MP, like MP5, will be reasonably short, and will give you additional practice with trees and streams, as well as providing you with some interesting statistical results to examine.

2 The programming assignment

To begin with, you will need to copy the given files into your own directory. The files are located in:

```
~cs225/src/mp6        (MP code)
~cs225/src/mp6/test   (test files)
```

3 Your assignment

Your task is to make the following changes to the given files:

1. Add a second data member to the `WordCounter` class, an AVL Tree object. Then, change the `InsertWord` function so that it inserts the words into both trees, rather than just one. (This change should not be much more complicated than a cut-n-paste; i.e. it really is as easy as you think it is.) The idea is that since an AVL tree has different performance than a BST (due to the balance of an AVL tree) using both types of trees will allow you to make comparisons between the behavior of the two trees. The existing `WriteToFile` and `ReadToFile` functions will no longer be correct, since you have added the AVL tree data member, but don't worry about that. In addition, for the `PrintWordInfo` function, change it so that a lookup is done in both member trees, and not just one (even though you only print the result once – the reason for this will be clear when you read the next item).

2. Add data members to the tree classes to store statistics. Specifically, you want a data member to store the number of node element comparisons needed for the last insert, remove, or find (any type of find) and also a data member to store the number of comparisons needed for all the inserts, finds, and removes so far (since the last clearing of the statistics). You also want three public functions in your tree classes: `ClearStats()`, which sets both statistic values to 0; `ReturnTotal()`, which returns the "total" statistics value, and `ReturnRecent()`,

which returns the "most recent operation" statistics value. Along with this, you want to add to the tree classes the ability to actually use these statistic values to count comparisons, meaning that when an insert, remove, or any type of find operation is invoked, the appropriate number of comparisons (most recent and total) will be stored in the statistic variables once the operation has completed.

3. Add input stream and output stream functions (`operator>>` and `operator<<`) to the BST and AVL tree classes. Since these are template friend functions you will likely need to handle the compilation in a manner similar to how `KeyPair` was handled. See the bottom of `keypair.h` for details. These stream operators should write the tree without destroying it, and should read in the tree information in manner that allows the tree to be reproduced exactly – including the original *structure* of the tree. In addition, add the same two operators to the `WordCounter` class – we will open files externally and then use these operators to write and read the WordCounters to and from the files.

Be sure to take advantage of the power of inheritance where you can. Also, don't forget your PSP work!

4 Handing in your code

To handin your MP6 code, use the command

```
handin cs225 mp6 wordcounter.h wordcounter.C bstree.h bstree.C avltree.h avltree.C
```