Csci 335 Assignment 2

Due Wednesday, October 9th

Using and Comparing Tree Implementations

The goal of this assignment is to become familiar with trees and compare the performance of the basic binary search tree with the self-balancing AVL tree. You may start with the book's implementation but you will have to modify it to complete the assignment.

Create a data structure that has as data members a string and a set of strings. This will be the data that we store in our trees and passed in as a template. This class should be comparable using the string member as the comparison key.

Create a test routine that does the following:

1. You will receive as input two text files, `rebase210.txt` and sequences`.txt`. After the header, each line of `rebase210.txt` contains the name of a restriction enzyme and possible DNA sites the enzyme may cut (with a ' indicating the cut location) in the following format:

   `enzyme_acronym/recognition_sequence/…/recognition_sequence//`

   For each of these lines you will insert a node into a search tree that contains as its search key a recognition sequence and as additional data the corresponding restriction enzyme. It is possible for a recognition sequence to be the same for several different enzymes, in which case you will add the new enzyme to the set of enzymes that match a given DNA sequence.

   Count and print the total number of recursive calls to insert after processing the entire file.
2. Allow the user to input a recognition sequence through standard input. Search for it in the tree and if it exists print the corresponding enzymes that would match it.
3. Print the number of nodes in your tree n and $\log_2 n$. Compute and print the average depth of your search tree, i.e. the internal path length divided by n. Also print the ratio of the average depth to $\log_2 n$. E.g., if average depth is 6.9 and $\log_2 n = 5.0$, then you should print $\frac{6.9}{5.0} = 1.38$.
4. Search the tree for each sequence in the `sequences.txt` file. Print the total number of recursive calls to the `contains` method.
5. Remove every other sequence in `sequences.txt` from the tree. Print the total number of recursive calls to the `remove` method.

6. Compute the statistics in step 3 and the search in step 4 again on the tree after the removals done in step 5.

You will test the above routine on both the binary search tree with regular deletion and the AVL tree with lazy deletion. You should write the test routine using templates so each tree can be used interchangeably. The trees should have identical interfaces. Name your program **`testTrees`**. It should run from the terminal as follows:

`testTrees <database file name> <queries file name> <flag>`

`<flag>` should be 0 for binary search tree and 1 for AVL tree.

Include a brief comparison of the statistics you observed in your README. Are the results what you would expect given the analysis of these data structures?