CS355 Assignment 1: Analysis

Victor Rogers, Martin Tuck, Tyler Yasaka

Analysis Question 1:

The runtime for GoToPrev() is O(h) where h is the height of the tree. In the worst case scenario, the function has to traverse to the leaf furthest away from the root to find the cursor position. Then, if the leaf we found was the smallest we would have to traverse to the largest value in the right sub tree. Therefore, in the worst case we traverse the height of the tree twice (2h). Therefore, GoToPrev() is O(h).

The runtime of GoToNext() is also O(h) where h is the height of the tree. Unlike GoToPrev() and many other methods in the BST class, GoToNext() uses iteration instead of recursion. In GoToNext() the worst case is the cursor being at the largest node. Although they have the same big O value, it can be expected that GoToNext() will run more efficiently by using iteration. There is a lot of overhead associated with making function calls and recursion will use up a lot of memory on the stack.

Analysis Question 2:

GoToBeginning and GoToEnd are “mirror functions”, so to speak. That is, they do exactly the same thing except one little detail is opposite (like in this case, go left instead of right or vice versa). Consequently, their runtimes are identical.

The runtimes of GoToBeginning and GoToEnd are linear. That is, they have a Big O of *n* (with n being the number of nodes in the tree) in the worst case. The worst case for GoToBeginning is if all of the nodes (other than the root) are left children, and the worst case for GoToEnd is if all of the nodes (other than the root) are right children. In the worst case, the function must traverse every node in the tree, and no more nodes, in order to reach the beginning or the end. Thus in the worst case, the number of iterations this function would have to make is equivalent to the the number of nodes in the tree.

The data value at the end of the GoToBeginning function is going to be the smallest value in the tree (if it is not empty). The data value at the end of the GoToEnd function is going to be the largest value in the tree (if it is not empty).

Analysis Question 3:

When using a temporary pointer to traverse a list, the temporary pointer must be assigned either to null, or to a node already in the list upon declaration. When the temporary pointer is declared, if node space is allocated, then every node that the temporary pointer traversed over would be overwritten by the empty node pointed to by the temporary pointer. The apparent danger is that all of the data traversed would be lost, and there could also be a potential for memory leaks.