Binary Search Trees : Building and Searching
Lab Assignment #7

by
Justin Anthony Timberlake

CS 303 Algorithms and Data Structures
October 11, 2017

1. Problem Specification

The purpose of this lab was to first implement a binarySearchTree class with the methods for inserting new nodes into the tree, printing out the values in an in-order traversal walk of the tree, and also a search, which takes a starting Node and a key value (in this case, a long) to locate specific values in the BST. The next goal was to read in the “UPC.csv” file which contained all the records, in which to search for the given search keys and records from the “input.dat” file.

2. Program Design

The steps taken to develop the program were:

a) Develop the Node class, on which the BST class is based.

b) Develop the BST class constructor, treeInsert and inOrderTreeWalk methods.

c) Generate some simple integer inputs to build the BST upon and demonstrate the inOrderTreeWalk for the in-class portion of the lab.

d) Develop the treeSearch method and alter the contents of the Node class to account for the String “data” instead of just integers.

e) Replace the display of the inOrderTreeWalk with file reading statements in the main method for reading in each line at a time from the “UPC.csv” file as well as the “input.dat” file, searching for each Node as it is generated and recording the total time it takes for all the searches, one after another.

f) Add print statements for the total time it took to build the list as well as to conduct all the searches done, cumulatively in nanoseconds. Store the values that have been found in the BST with an ArrrayList, which is then used to print out the given descriptions of said values after the benchmarks.

No additional methods were used in the primary class.

The following methods were used in secondary class, Node.

a) Node(long thing, String words) - constructor for objects of type Node, setting the data equal to the given String and the key value equal to the given long.

The following methods were used in secondary class, BST.

a) BST() - base constructor for objects of type BST.

b) treeInsert(Node z) – Puts the given node z in its proper place within an existing tree based on comparison values of the keys, which are long values.

c) inOrderTreeWalk(Node x) – starting at the x node, recursively prints out values of the keys of the left child, the current node, and then the right child, recursively. This generates sorted order.

d) treeSearch(Node x, long k) – starting at node x, this method searches recursively through the BST for the given long value, k. For searching through large trees recursively, stack overflows can occur with the use of recursion (using the stack). However, when testing my program, all such issues were not present in the final product of the code.

3. Testing Plan

Once the BST and Node class were built and tested in-class and only a few modifications were made for the homework portion, the only thing left to test was the actual time elapsed performing the operations mentioned in this assignment. This was calculated by getting the nanoTime() directly before and after each insert or search operation only and then adding their respective totals as the loop continues to parse and generate results to be searched, as seen in the code. These two values are then printed out for benchmark testing purposes. The specific descriptions of the given search values are printed after the bench mark of each of the operations to signify that all the elements have been found.

4. Test Cases

For building the given 177, 650 records in the “UPC.csv” file into a BST, the total time elapsed was 76,461,176,294 nanoseconds in a given test.

For searching the given 17 values in the “input.dat” file out of the aforementioned 177,650 records, the total time elapsed was 1,986,371 nanoseconds.

5. Analysis and Conclusions

Clearly, while the BST searching method is very fast for searching a given number of inputs, it takes a significant time to actually build and implement a proper BST in the first place, so when attempting to search for specific values, the very long building time needs to be considered in addition to the search time as opposed to something such as linear or binary searches.

6. References

All code in this assignment was either presented in prior labs and assignments or provided through means of pseudocode within the in-class portion of the lab.