Project 1

Binary Search Tree

CS241 - Data Structures & Algorithms II
MWF 11:45 - 12:50pm
Cal Poly Pomona
Dr. Hao Ji

James McCarthy
jdmccarthy@cpp.edu

January 30 th, 2017

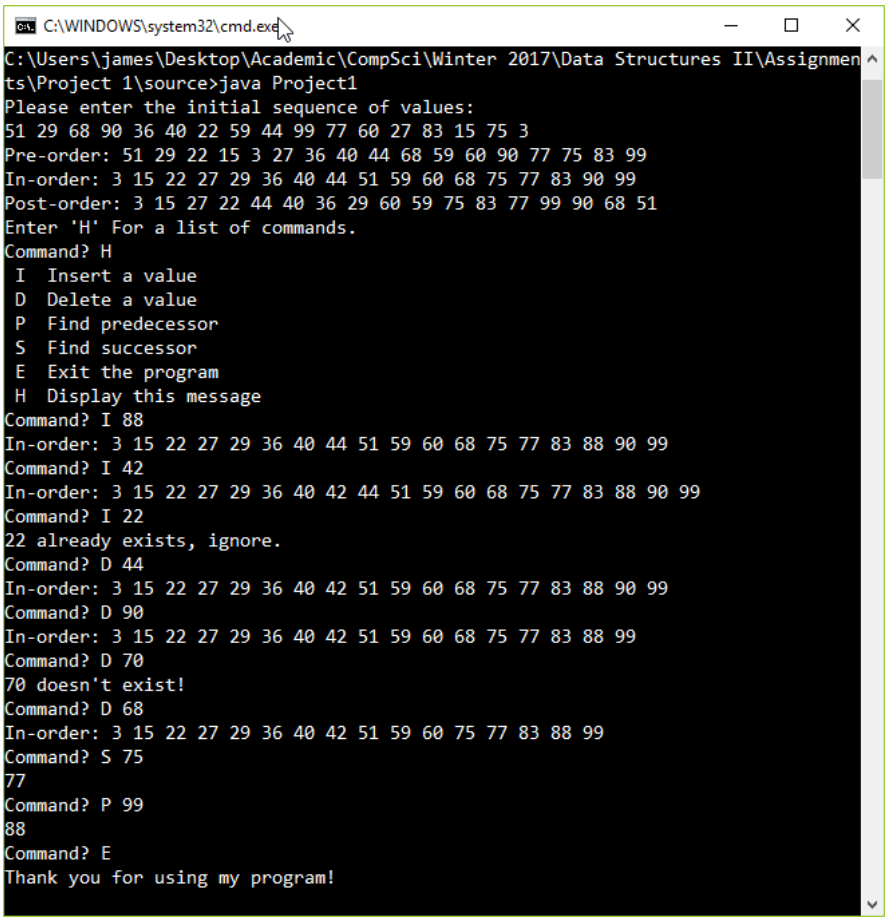

Figure 1: Program output.

1 Project Description

A binary search tree is an abstract data type that combines the hierarchical organization of a tree with sorted organization that reduces the time complexity involved in locating data stored in the tree. In order to better understand the structure and application of a binary search tree, this project utilizes a basic BST implementation written. Written in the Java programming language, this project additional serves the purpose of facilitating familiarization with the language itself.

2 Specification

The program reads from the standard input, via terminal, a sequence of integer values, each seperated by a single space. The program then builds a binary search tree from the input sequence in the order in which they were entered and performs the following tasks:

- Prints to the terminal the tree's values using preorder, inorder, and postorder traversal methods.
- Allows the user to optionally add any non-duplicate value to the tree or delete any value present in the tree.
- Allows the user to enter a value, that is present in the tree, to produce a printed output of the given value's inorder predecessor.
- Allows the user to enter a value, that is present in the tree, to produce a printed output of the given value's inorder successor.

The aforementioned functionality, including adding and deleting, is implemented recursively with special regard to efficiency, ensuring that the average time complexity does not exceed $\log_2(n)$. The program does not support duplicate value entries and handles any attempt to enter duplicate values accordingly.

3 Testing Methodology

To ensure the program met the outlined specification without error, rigorous testing was applied incrementally over the course of development. Once the basic framework of a tree data structure was created, tests were applied to subsequent tree method implementations to ensure rigorous adherence to the order and structure that makes a tree a binary search tree.

First, a simple inorder traversal method was created to print the values in the tree's nodes recursively. The inorder traversal was chosen over other traversal methods because in a correctly formatted binary search tree, the values will print in ascending order, making any error in the tree's structure easily recognizable given any set of data values. Thereafter, the add and delete methods were tested thoroughly using inorder printing for flaws and

were tweaked until no such flaws remained. Finally, the predecessor and successor methods were tested similarly and the program was run continually in its entirety and tweaked until it matched exactly the specification given.

Outside of the core project specification, additional testing was also done to ensure that the program did not crash on invalid inputs. Regular expressions and string matching were used to ensure that only valid inputs could be entered and the program would not crash. Also, Java 8's array stream feature was used to ensure that no overflow error could occur when passing values from the input stream into the sequence array. If there is any complication regarding the use of Java 8+ features, please respond via email and an alternative can be provided.

4 Comments

Since the beginning of this course I have tried my best to obtain a thorough understanding of the material, rather than just the level of understanding required to receive a decent grade. Among the many topics I am learning this quarter, data structures and algorithms is the one in which I am most interested in personally. I have taken it upon myself to devote more of my available study time to reading and understand the book. Additionally, I have endeavored to complete many of the optional programming exercises given in the book. In doing so, I have grown accustomed with being provided beforehand with most of the code required for a project. However, on this project I wanted to test how well I truly understood the implementation of a binary search tree and so I closed my book and began working completely from memory.

For the most part, isolating myself from the temptation to copy code went smoothly, but that isn't to say there weren't times where I stumbled and had to seek some external reference. This happened occasionally when I became stuck and could not figure out how to fix a bug or when I encountered topics that were not explicitly covered in the book, specifically the inorder successor and predecessor methods. Being a programmer that typically relies heavily on either internet or textbook references, I am suprised that I was able to accomplish as much as I did using my own personal knowledge of this particular subject. Despite having to occasionally seek outside help, I feel confident that I am capable of implementing a binary search tree on paper, either for an exam or a potential job interview.