Jay Offerdahl

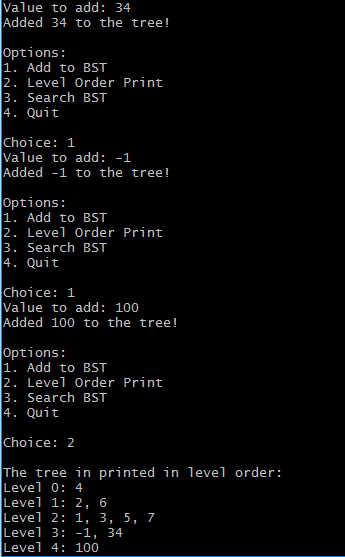
EECS 560

Lab 2 Tests

**Testing the BinarySearchTree class:**

1. Adding values

To test the push(int) method, I essentially copied “tree->push(x)” with several different values for x, and printed out the tree. An example of input would be the text file that contained: “1 2 3 4 5 6 7”, the output using the level order traversal & print was as expected. Next, I added several values to this tree and printed out the tree in level order style again, which gave proper output.

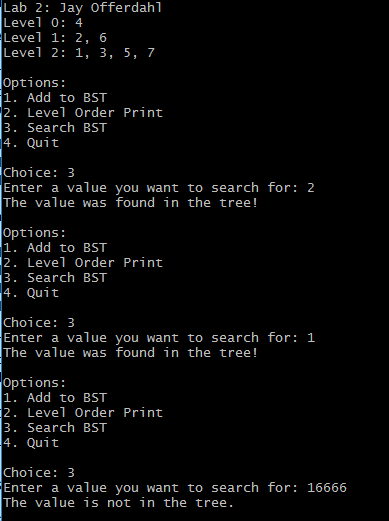


1. Constructors/getters/setters

These methods were tested throughout testing the rest of the program and weren’t complex.

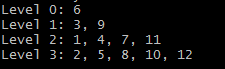
1. Search

Searching the BinarySearchTree involved running many commands with differing inputs to check functionality. For example, if I added values to the tree so the following tree was created:



1. Level order traversal & printing

Testing the level order traversal and printing took some time as I had to resolve an error when checking the children of the current node. My testing included the previous tests when adding values, as well as changing the input. It’s worth noting that the tree isn’t full unless there are 2^k – 1 nodes, so output looks strange when this is the case. For example, the following output shows the tree with input list 1, 2, … 12:

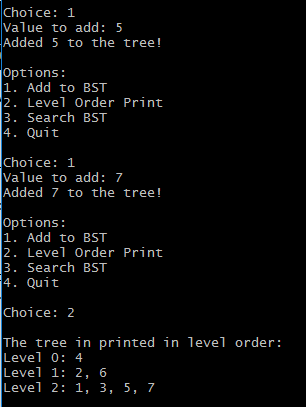


After drawing out the tree, the input is in accordance with what is expected.

1. Loading a pre-sorted array

When loading the pre-sorted data file, I spent most of my time thinking about the algorithm before actually writing it, so minimal testing was needed. The testing for this function consisted of changing the input file data2.txt with different values, as in the previous section in order to ensure correct output. However, to ensure the function is correctly working, I also inputted all the numbers (in level order) from 1 – 7 manually, and compared that to the correct output.

Here’s the output after manually entering values:



And here’s the output using the method I created to load the tree in O(n) time:

