CPSC 319

<u>Assignment #3: Binary Search Trees</u>

Winter 2022

Lab Report Due Date: March 18th, 2022

By: Nasih Nazeem

Lab Summary

The goal of this lab is to create a binary search tree that can read an input file holding data that is used to insert into the tree. We would create traversal methods for depth-first and breadth-first traversal. Lastly, they should visit each node in those traversal methods and write to two separate output files.

Binary Search Trees

- a3input1.txt

Please view the .png screenshot that was submitted.

- a3input2.txt

Please view the .png screenshot that was submitted.

Complexity Analysis

1. If tree is well-balanced we can read the tree to have a Big-O of O(log n)
2. Worst-case would be when the tree starts to look more like a Linked List. If the tree was inserted is an alphabetical order, we would only see one subtree extend from the root node.
3. Answers:
    a. The worst case scenario for a depth-first in-order traversal would look like a Linked List, meaning the Big-O Notation would be O(V).
    b. For breadth-first, the worst case scenario would have all nodes on the same level and the Big-O Notation for it would be O(n) where n is the maximum number of nodes in a single level.
    c. This depends on the way the nodes are visited and the order in which the nodes are being inserted into the tree. For example, if we were to insert nodes in a worst-case scenario, visiting the nodes from a top-down method (BFS) where you visit each level first will use less memory than attempting to visit each node in a DFS traversal strategy. If the input nodes were to be inserted into the tree at random, this would mean that DFS would use less memory as there is only a need to store a maximum of 2 values, where if we use BFS traversal, we would be storing more than 2 values at a time. DFS makes it easier to reach each individual node whilst maintaining the least amount of memory, in a scenario where the inserted nodes and input at random.

References:

1. *Binary Search Tree | Set 2 (Delete)*. (2014, January 30). GeeksforGeeks.

   https://www.geeksforgeeks.org/binary-search-tree-set-2-delete/

2. *Binary Search Tree Visualization*. (n.d.). Www.cs.usfca.edu. https://www.cs.usfca.edu/~galles/visualization/BST.html
3. *DFS vs BFS (in detail)*. (2020, May 6). OpenGenus IQ: Computing Expertise & Legacy. https://iq.opengenus.org/dfs-vs-bfs/#:~:text=The%20time%20complexity%20of%20DFS%20is%20O%20%28V
4. *How to check if file exists in Java*. (2014, November 5). HowToDoInJava. https://howtodoinjava.com/java/io/how-to-check-if-file-exists-in-java/#:~:text=Check%20if%20file%20exists%20with%20File.exists%20%28%29%20method
5. *How to get the total number of lines of a file in Java – Mkyong.com*. (n.d.). Mkyong.com. https://mkyong.com/java/how-to-get-the-total-number-of-lines-of-a-file-in-java/
6. *Java Exceptions (Try...Catch)*. (n.d.). Www.w3schools.com. https://www.w3schools.com/java/java_try_catch.asp#:~:text=The%20technical%20term%20for%20this%20is%3A%20Java%20will
7. *Java output formatting for Strings*. (n.d.). Stack Overflow. Retrieved March 18, 2022, from https://stackoverflow.com/questions/4418308/java-output-formatting-for-strings
8. *Level Order Traversal of a Binary Tree: Breadth First Search*. (n.d.). Www.enjoyalgorithms.com. Retrieved March 18, 2022, from https://www.enjoyalgorithms.com/blog/level-order-traversal-of-binary-tree