Asif Bux
ENSF 593 Lab 11

Complexity Analysis

Let *n* be the total number of records stored in the data structure.

1. Assuming that the records are inserted into the tree in random order, what is the height of your tree expressed using big-O notation?

If the nodes are inserted in randomized order in the BST, then the Big O notation will be O (log n), as shown in the figure below for each traversal there is significant less ways to get to the node such for height 2, N/4. Also, by solving and adding (N + N/2 + N/4 + N/6 + N/9… 1) the time cost of this iteration would result in Big O(logn).

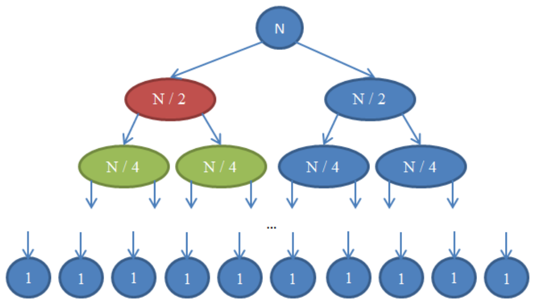


Figure 1 Big O of BST
Picture taken from https://es.wikipedia.org/wiki/Inducci%C3%B3n_estructural_y_definiciones_recursivas

2. What is the worst-case height of the tree? What input gives the worst case?

The worst-case height of the tree would be when, the tree is created in way comparable to a LinkedList, where one node is significantly long, either left of right child that extends to create a LinkedList like structure. In the this, the tree's height will be n-1. Each following input has to be either all less or greater than its parent node and this will degenerate into LinkedList as shown in the diagram.

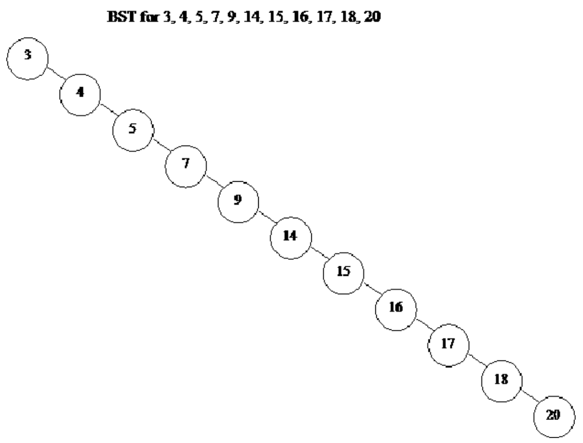


Figure 2: BST Picture taken from: http://cs-study.blogspot.com/2012/11/avl-tree.html

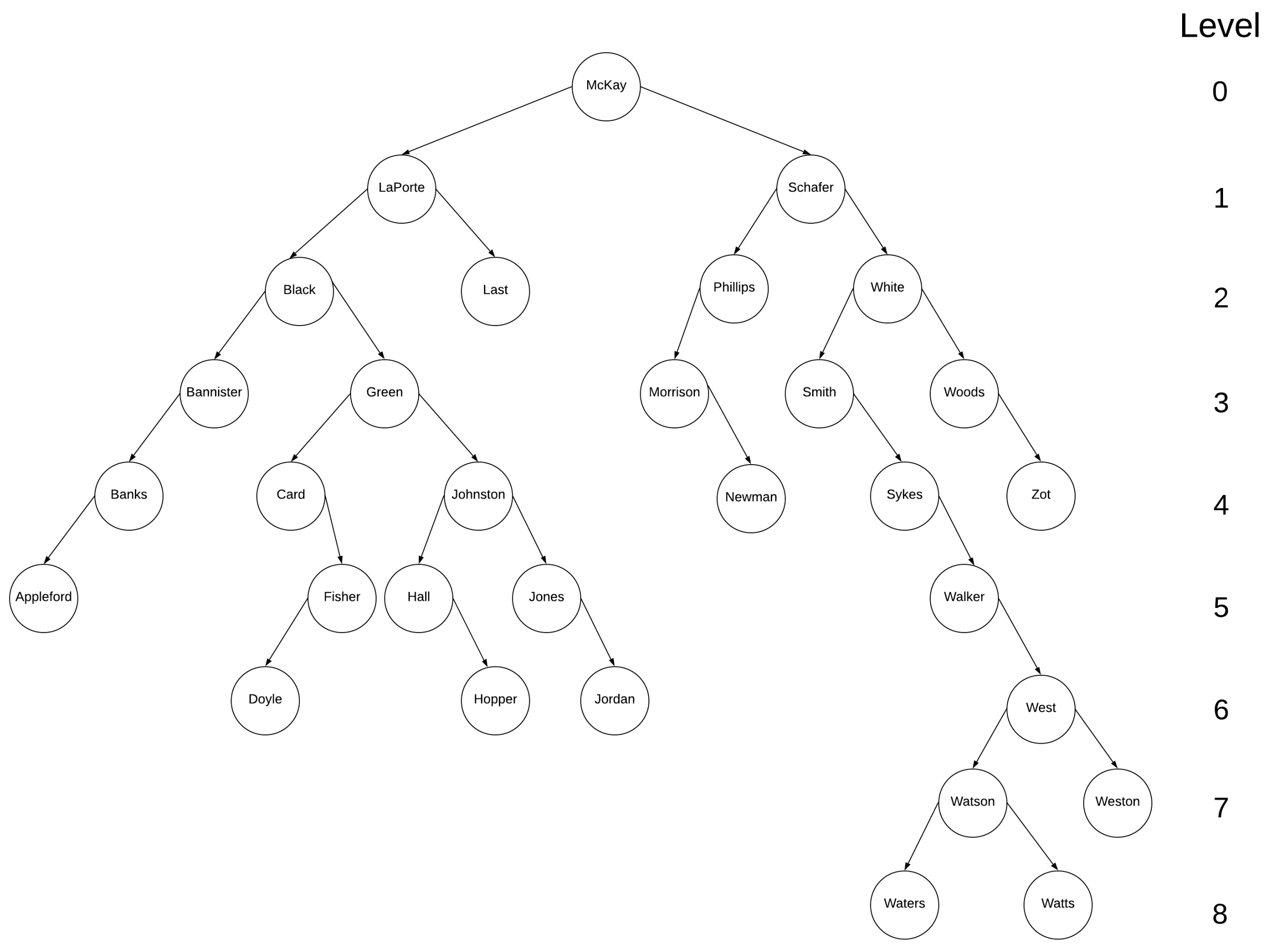
3. What is the worst-case space complexity of the depth-first, in-order traversal and breadth-first traversal? Compare your implementation of these two methods: is there one that will outperform another in terms of memory usage for a specific data set? Discuss.

The worst-case space complexity for a depth-first traversal would happen when the BST has degenerated into a LinkedList as shown in the example above. The traversal will run for n-1 for entire length of the tree before returning back to the root in the recursive code of my implementation. The Big O would be O(n-1) or O(n) for depth-first, in order traversal.

On the other hand, the worst-case space complexity for a breadth-first traversal would happen when BST is spread out in way where every level is filled n nodes or a perfect tree where the nodes at the last level will be (n/2) + 1 and since the worst-case space complexity for breadth-first traversal is also **O(n).** One point to note, however, if the tree was to degenerate in LinkedList for breadth traversal, then the complexity would be the best case as where Big O is O(1) due to queue implementation and have one element at very level.

Memory usage is directly linked with height of the BST, so in this case, the best case or average case space complexity for depth-first traversal would be O(logn). However, for long and unbalanced trees such as LinkedList alike, breadth-first traversal would perform better due to queue implementation and minimum memory usage with O(n) space complexity.

Picture of the BST


Level
0
1
2
3
4
5
6
7
8
McKay
LaPorte
Schafer
Black
Last
Phillips
White
Bannister
Green
Morrison
Smith
Woods
Banks
Card
Johnston
Newman
Sykes
Zot
Appleford
Fisher
Hall
Jones
Walker
Doyle
Hopper
Jordan
West
Watson
Weston
Waters
Watts