Binary Search Tree

Navid Hariri

CS 182

December 8, 2018

# Report

A theoretical full or balanced tree with 2^n – 1, with n being the number of levels on the desired tree, and the whole expression would be equivalent to the number of nodes in the tree. If n were to be set equal to10, it would result in 1023 nodes in the binary search tree. Theoretically, or the best case scenario, the tree would have 10 levels of nodes with root being equal to the first level. The root would then be the median value and it would have the same amount of nodes on the right side and on the left side, thus making the tree fully balanced. With a fully balanced tree the time complexity of the algorithm would be equivalent to Log(n). Theoretically, half of the nodes should be at the leaves of the fully balanced tree. If the tree had more nodes on the left side than the right side, it would be said to be heavier on the left side than the right. The same would be said for having more nodes on the right side than the left side. The worst case scenario for the binary search tree would be if every single number value inserted into the tree was from ascending or deciding number which would turn the tree into a singly linked list with a time linear complexity or big O(n).

The simulation asked to insert 2^10 – 1 or 1023 nodes into the tree and find what level each node is on. The first trial required that the values from 0 to 1023 be shuffled and then inserted into the binary search tree. This should then be done 1,000 times and the average recorded in sample and the average of the averages is shows in the table below for Create Average. This tree would represent the average case for the binary search tree. In a theoretical case this tree should have 10 levels and each node either has two children or is a leaf. In a theoretical tree, most nodes should be on level 10, however, the simulation shows that the average case most nodes are on level 12, which is close to 10 but the tree is a bit unbalanced. Resulting most nodes to not be on level 10 but to be on level 12. It also shows that on average the maximum node level of the tree was on level 22, making the tree very much unbalanced.

The simulation also shows that when deleting half of the nodes the majority of the nodes were on level 11 and the height of the tree is 20. To compare this with a theoretical tree that would have 2^9 -1 should have 9 levels with every node having two children and the last level being all leaves. There should be half of the tree or 256 on the leaves on level 9. However, the simulation shows that for an average most nodes are on level 11 with the height of the majority of the tree being 20. This again is close to the theoretical tree because most nodes are on level 11 which is close to 9, however, the length of the tree is a lot longer making the tree again lopsided.

The last simulation shows more of the same, and that when the nodes are reinserted the level of tree becomes similar to the beginning when it was first created with most nodes being on level 11 and the maximum height being 12. Showing that for all simulation the tree is close to being theoretical, however, the height is very different, and upon creation, deletion, and reinsertion. The tree was mostly similar, and did not change much.

The binary search tree currently is somewhat generic and with more time I could have implemented CompareTo in the delete method as well making it more generic.

# UML Diagram

# 

# Results

|  |  |  |
| --- | --- | --- |
| Name | Mean | Max |
| Create Max | 22.16 | 30.00 |
| Create Average | 12.04 | 15.08 |
| Delete Max | 19.21 | 28.00 |
| Delete Average | 10.58 | 13.63 |
| Reinsert Max | 20.50 | 28.00 |
| Reinsert Average | 11.51 | 14.30 |