Data Structures
Into to AVL Trees

Cain Susko

Queen's University
School of Computing

February 19, 2022

Balanced BST

A slight problem with Binary Search Trees is that if they are unbalanced (ie. The left subtree of the root-node is larger than the right) then the efficiency of a BST can be diminished. This is because if a tree is unbalanced there is a high lekelyhood that the data is organized in an ordered list rather than the intended tree structure. The solution to this problem is the AVL tree.

AVL Tree

An AVL tree is a Binary Search Tree that uses modified indert and delete operations to stay Balanced as its elements change. It does this by maintaining a Balance factor in each node of 0, 1, or -1.
An AVL Tree is more suitable to search through often (rather than the Red Black tree, which will be covered later, which is more suited for often inserts and deletes).

Balance Factor the balance factor is, for a tree node T, the height of the node T's right subtree minus the height of the left subtree.

$$BF(T) = H(T.R) - H(T.L).$$

This can be interpereted as we only allow at most 1 node difference between the two subtrees. Note, the height is the max number of *edges* between a given node and a leaf. Additionally, the height of an empty tree is -1.
We focus on height so much because the Big O, or complexity for a search, is increased by how high the tree is, as is cannot have more than 2 nodes off of one node and has a max width for each height,

Height The Height of an AVL tree with N nodes is $O(\log N)$. To Prove this, observe the following:
Let us bound $n(h)$ which is the minimum number of nodes an AVL tree of height h can have. The base cases are:

$$n(0) = 1 \;\; n(1) = 2 \;\; n(2) = 4.$$

From this basis we can say that for $n(h) > 2$, an AVL Tree of height h has, at minimum, the root node with one AVL subtree of height $h - 1$ and another

AVL subtree of height $h-2$. Thus $n(h)$ for AVL subtrees can be generalized as for AVL subtrees can be generalized as:

$$n(h) = 1 + n(h-1) + n(h-2) \qquad n(h) > 2^i * n(h-2i).$$

However, we can even further simplify the inequality on the right by removing the variable i by claiming that $i = \lceil \frac{h}{2} - 1 \rceil$. If h is even then, $h - 2i = h - (h - 2) = 2$. And if h is odd then $h - 2i = h - \big(2\lceil \frac{h}{2} \rceil - 2\big) \implies h - ((h+1) - 2 = 1$. Thus, we can generalize the inequality to the following:

$$n(h) \geq 2^{(\frac{h}{2})-1}.$$

Additionally, a tree of height h has a search complexity of $O(\log N)$ where N is the number of nodes in said tree.
The Height of any given node is the maximum between its left and right subtree plus 1. This means that there is a relationship bewteen different nodes' height and the height could be calculated recursively.
However, within an AVL tree it is common to maintain a height for each node as a field within the node which is altered whenever the nodes' height is altered as the height is needed often to measure and maintain the balance factor of the tree (and thus, each of its nodes).