More on BSTs

As we have described them, BSTs can become unbalanced. Note that in general BSTs cannot be perfectly balanced.

An AVL tree is a balanced BST. In an AVL tree, for each node in the tree, the heights of its left and right subtree can differ by no more than one.

Note that balancing refers to height not the number of nodes in a subtree.

Because it is balanced, the height of an AVL tree is O(logn)

AVL tree notation

Each node in an AVL tree is assigned a label

LH = left height

EH = even height

RH = right height

AVL operations

Like most balanced search trees, AVL tree balance the tree after insertion and deletion. Search, min, and max are unmodified (stay the same)

All tree insertion is a 2-step algorithm:

1. Insert the new value as a leaf by searching for it (will not break search property, but may break balance)
2. Rebalance the tree if necessary by examining nodes along the search path (only look at the half you go down)

AVL trees are rebalance via rotations

4 cases for insertion

1. Left-of-left: left high subtree of a node that was left high already and now is unbalanced

Step required:

* Rotate the unbalanced node to the right
* The left child of the unbalanced node becomes the new root of the subtree
* Any right child of the new root becomes the left child off the old root

1. Right-of-right: symmetric to case 1
2. Right-of-left:

Step required:

* Rotate the left child of the unbalanced node to the left
* This all result in case 1
* Rotate the unbalanced node to the right as before now

1. Left-of-right: symmetric to case 3