6. AVL Trees, AVL Sort

Link: https://www.youtube.com/watch?v=FNeL18KsWPc

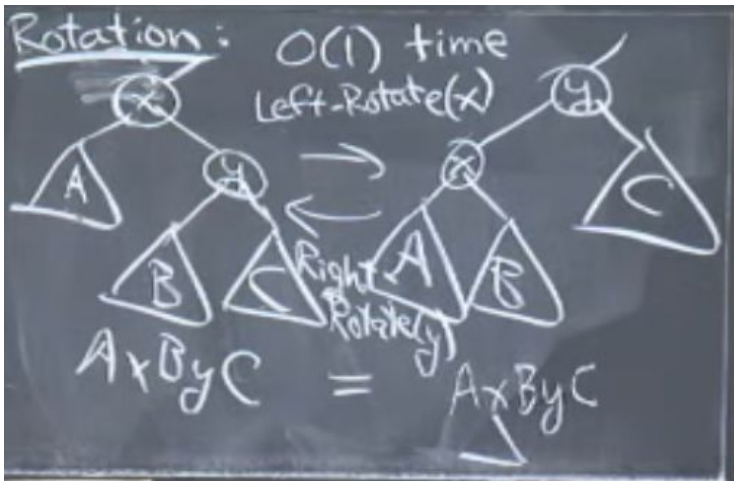
Balanced Search Tree

- Height - length of the longest path going from the root down to the leaf
- A tree is balanced if h = O(lgn)
- Height of a node
    - Length of longest path from it down to a leaf
    - max(height(left child),height(right child)) +1
- Rebalancing should occur when the height gets too large

AVL Trees

- Require heights of left and right children of every node to differ by at most +-1
- The reason why its not 0 is because it's difficult to accomplish a perfectly balanced BST
    - It's impossible in trees with even amount of nodes
- AVL trees are balanced:
    - Worst case is when right subtree has height 1 more than the left for every node
    - N;h = min # nodes in an AVL tree of height h

AVL Insert

1) Simple BST insert
2) Fix AVL property
    a) From changed node up, suppose x is lowest node violates AVL
    b) Assume right(x) higher aka x.right
3) 

4) If x's right child is right-heavy or balanced

AVL Sort

- Insert n items O(nlgn)
- In-order transversal - O(n)
- AVL allows you to find successor and predecesor

    - Heap allows you to find min and max

Abstract Data Type

- Insert and delete
- Min
- successor/pred