AVL Trees

Balanced binary search trees

Idea: avoid long, skinny branches to get \( O(\log n) \) worst case add, remove, find

Method: keep track of measure of balance of each node's subtree

\[
(\text{height left subtree} - \text{height of right subtree})
\]

recognize when \(|\text{balance}| > 1 \), rebalance if so

an AVL tree with \( n \) nodes has height \( \leq \log n \)

14 29 40 20 50 15 99 45 18
Rotations

- Bottom-most unbalanced node

Inorder: \( T_1 \cdot y \cdot T_2 \cdot x \cdot T_3 \)

Single right rotation:

\[ 0(1) \] to fix

\[ O(1) \] to fix

\[ O(\log n) \] to recompute

Double right rotation:

\[ 0(1) \] to fix

\[ O\left(\frac{1}{2} \log n\right) \] to recompute

Double left rotation:

\[ 0(1) \] to fix

\[ O\left(\frac{1}{2} \log n\right) \] to recompute