Balanced Binary Search Trees - AVL Trees

"Symmetry is reallly important to preserve laziness."

- Danny Heap

Balance Factor

$height(v) =$ the length of the longest path from v to a leaf ($height(empty\,tree) = -1$)

balance factor $BF(v) = height(v.right) - height(v.left)$

AVL Trees

(Adelson-Velski-Landis Trees)

An AVL tree T is a BST where for every node $v \in T$, $-1 \leq BF(v) \leq 1$

Properties

- for n nodes, height is $\Theta(\log(n))$
  - $height(T) \leq 1.44 \log_2(n+2)$
- can do inserts and maintain balance property in $\Theta(\log(n))$ time

Operations

- `search(T, x)`
- `insert(T, x)`
  - insert `x` as a leaf where it should be ($\log(n)$ time)
  - go back up the tree and rebalance using rotations wherever necessary
    - if $BF(v) > 1$ and $BF(v.right) \in \{0, 1\}$ then

    `left_rotate(v)`
  - if $BF(v) > 1$ and $BF(v.right) = -1$ then `right_rotate(v.right)` (so that $BF(v.right) \in \{0, 1\}$) then `left_rotate(v)`
  - (mirror cases above if $BF(v) < -1$)

- `delete(T, x)`