AVL Height Balance
- for each node $v$, the heights of its subtrees differ by $\leq 1$.

AVL tree: A binary search tree that satisfies this condition.

AVL Trees I
- Basic defs
- Height props
- Rotations

Theorem: An AVL tree of height $h$ has at least $F_{h+3} - 1$ nodes.

Proof: (Induct. on $h$)
- $h = 0$: $n(h) = 1 = F_3 - 1$
- $h = 1$: $n(h) = 2 = F_4 - 1$

For $h \geq 2$:

$$n(h) = 1 + n(h-1) + n(h-2)$$

$$n(h) = 1 + (F_{h+2} - 1) + (F_{h+1} - 1)$$

$$n(h) = F_{h+3} - 1$$

for any $k' < h$

Corollary: An AVL tree with $n$ nodes has height $O(\log n)$.

Proof: Fact: $F_n \approx \varphi^n / \sqrt{5}$ where

$$\varphi = (1 + \sqrt{5}) / 2 \text{ “Golden ratio”}$$

$$n \approx \varphi^h$$

$$h \leq \log_{\varphi^2} n + c$$

$$h \leq \log_{1.44} n$$

$\Rightarrow n \approx \varphi^h$$

$\Rightarrow h \leq \log_{1.44} n / \log_{1.44} 10$

$O(\log n)$

Conjecture: The minimum number of nodes in an AVL tree of height $h$ is $F_{h+3} - 1$.
Double rotations:

AVLNode rebalance (AVLNode p)
if (p == null) return p
if (balanceFactor(p) < -1)
    if (height(p.left.left) >= height(p.left.right))
        p = rotateRight(p)
    else p = rotateLeftRight(p)
else if (balanceFact(p) > +1)
    [...] (symmetrical)
updateHeight(p); return p

AVLNode insert (Key x, Value v, AVLNode p)
if (p == null) p = new AVLNode(x, v)
else if (x < p.key)
    p.left = insert(x, v, p.left)
else if (x > p.key)
    p.right = insert(x, v, p.right)
else throw Error - Duplicate!
return rebalance(p)

AVL Trees II
- double rotations
- insertion

Find: Same as BST. ~O(log n)
Insert: Same as BST but as we "back out" rebalance

How to rebalance? Bal = -2
Left-left heavy
Left-right heavy
Right-right heavy
Right-left heavy

AVL Tree:

AVLNode: Same as BSTNode (from Lect 4) but add:
        int height

Utilities:

BSNNode rotate LeftRight (BSNNode p)
p.left = rotateLeft (p.left)
return p

int height (AVLNode p)
return \( \begin{cases} 
  0 & \text{if } p \text{ is null} \\
  p.\text{height} & \text{otherwise}
\end{cases} \)

void updateHeight (AVLNode p)
p.height = 1 + max (height (p.left),
height (p.right))

int balanceFactor (AVLNode p)
return height (p.right) - height (p.left)
Balance factor -2
Deletion: Basic plan
- Apply standard BST deletion
- find key to delete
- find replacement node
- copy contents
- delete replacement
- rebalance

AVL Trees III
- Deletion
- Examples

AVL Node delete (Key x, AVL Node p)
: same as BST delete
return rebalance(p)

Examples:
Note: Tree stores heights
Bal. factors computed as needed
**Node types:**
- **2-Node**
  - 1 key
  - 2 children

- **3-Node**
  - 2 keys
  - 3 children

**Recap:**
- **AVL:** Height balanced
- Binary tree

- **2-3 tree:** Height exact
- Variable width

**2-3 Trees**

**Def:** A 2-3 tree of height $h$ is either:
- Empty ($h = -1$)
- A 2-Node root and two subtrees, each 2-3 tree of height $h-1$
- A 3-Node root and three subtrees, ... height $h-1$.

**Thm:** A 2-3 tree of $n$ nodes has height $O(\log n)$

**Roughly:** $\log_3 n \leq h \leq \log_2 n$

**Example:**
2-3 tree of height 2

**Conceptual tool:**
We’ll allow 1-nodes
- 4-nodes temporary
- 1-node
  - 4-node
Announcements - 02/23

- Programming Assignment 1
  → Due Wed of next week 11:59pm (3/1)
  → Handout, skeleton, + test data available
  → Autograder is up

- HW2 - Coming soon
  → Due Tue, Mar 7

- Midterm 1 - Thu, Mar 9, in class