Announcements
AVL Trees:

Height-Balance Property:
for every node of the tree, the heights of the children differ by at most 1.

\[ \Rightarrow \text{max height} \leq 2 \log_2 n \]

(How do we calculate height again?)

Recursive: look at 2 children, take max, +1
Insert: Do BST insert.

Then find lowest unbalanced node $z$: 
Key operation:

- pivot (x)

\[ \begin{array}{c}
T_1 \quad T_2 \\
\quad x \quad y \\
T_3 \\
\end{array} \begin{array}{c}
T_1 \quad T_2 \quad T_3 \\
\end{array} \]
Removing in AVL trees

Step 1: Remove - just like in BST

Step 2: Re-balance (if removal violated H-B property.)

Note: Unlike insert, remove could actually un-balance all the way to the root.
Example:
remove (44)

Remove
Algorithm to remove

- Remove as in BST

- Track lower node that was removed.

- Travel up tree, searching for unbalanced nodes (it fixing) until you reach the root.
Performance

For insert & delete, follow root to leaf path at most 3 times:
- find
- next in inorder (for remove)
- travel back up tree balancing

At each node:
- reset height
- \( \leq 2 \) pivots
- reset heights again (if pivot)

\( \leq \) at most 60 operations: \( O(1) \)

Total time: \( O(\log n) \)
Next HW:

Remove in an AVL tree.

Caution: Testing will be a significant portion of your grade!

Lots of cases, since imbalances could propagate all the way up the tree.

I'll post this, but it won't be due until a week after the exam.