AVL Trees
AVL Tree Definition (§ 9.2)

AVL trees are balanced.

An AVL Tree is a binary search tree such that for every internal node $v$ of $T$, the heights of the children of $v$ can differ by at most 1.
Insertion in an AVL Tree

- Insertion is as in a binary search tree.
- Always done by expanding an external node.
- Example:

Before insertion:

- Binary tree with nodes 17, 32, 48, 62, 44, 78, 50, 88.

After insertion:

- Insertion of 54 into the tree.
- The tree remains balanced.

© 2004 Goodrich, Tamassia
AVL Trees
Insertion Example, continued

unbalanced...

...balanced
Removal in an AVL Tree

Removal begins as in a binary search tree, which means the node removed will become an empty external node. Its parent, \( w \), may cause an imbalance.

Example:

Before deletion of 32

After deletion
Rebalancing after a Removal

Let $z$ be the first unbalanced node encountered while travelling up the tree from $w$. Also, let $y$ be the child of $z$ with the larger height, and let $x$ be the child of $y$ with the larger height.

We perform a rotation to restore balance at $z$.

As this restructuring may upset the balance of another node higher in the tree, we must continue checking for balance until the root of $T$ is reached.
Running Times for AVL Trees

- a single rotation is $O(1)$
  - using a linked-structure binary tree
- find is $O(\log n)$
  - height of tree is $O(\log n)$, no restructures needed
- insert is $O(\log n)$
  - initial find is $O(\log n)$
  - Restructuring up the tree, maintaining heights is $O(\log n)$
- remove is $O(\log n)$
  - initial find is $O(\log n)$
  - Restructuring up the tree, maintaining heights is $O(\log n)$