Insertion Order matters

- Difference between these binary search trees?
  - Insert: 4, 2, 1, 3, 6, 5, 7
  - Insert: 1, 2, 3, 4, 5, 6, 7

Self-balancing Trees

- Variants of binary search tree
- Self-balancing
  - Additional rules for how to structure tree
  - Needs operations to move nodes around
- Constraints on height differences of leaves
- Ensures that tree does not degenerate to list

Complexity of Operations

- Lookup Time
  - Balanced tree: $O(\log n)$
  - Unbalanced tree: $O(n)$
- Insertion time
  - Balanced tree: $O(\log n)$
  - Unbalanced tree: $O(n)$
- Balance does matter for efficient trees.
AVL Tree

- What is AVL tree?
  - A self-balancing binary search tree
  - Adelson-Velskii and Landis' tree, named after the inventors
  - For every node \( v \) of tree \( T \), the heights of the children of \( v \) differ by at most 1.
  - Height: \( O(\log n) \)

Insertion

- Insertion is as in a binary search tree.

Restructuring

- Single Rotation:
  - \( T_0 \)
  - \( T_1 \)
  - \( T_2 \)
  - \( T_3 \)
  - single rotation
  - \( a = x \)
  - \( b = y \)
  - \( c = z \)
Restructuring

- Double rotations:

  \[
  \begin{align*}
  &a = z \\
  &b = x \\
  &c = y \\
  &\text{double rotation}
  \\
  &a = y \\
  &b = z \\
  &c = x \\
  &\text{double rotation}
  \\
  &a = z \\
  &b = x \\
  &c = y
  \end{align*}
  \]

Insertion Example

Example 10

After insertion unbalanced

Before deletion of 32

After deletion: unbalanced

Removal

Reconstructing after a Removal

- Node definition

  - \( z \): the first unbalanced node encountered while travelling up the tree from the violating node \( w \)
  - \( y \): the child of \( z \) with the larger height
  - \( x \): the child of \( y \) with the larger height
Reconstructing after a Removal

- Recursive reconstructing
  - Restructuring may upset the balance of another node higher in the tree.
  - We must continue checking for balance until the root is reached.

AVL Tree Performance

- Restructuring
  - A single restructure takes $O(1)$ time.
  - Using a linked-structure binary tree

- Search
  - A search takes $O(\log n)$ time.
  - The height of tree is $O(\log n)$, no restructuring is needed.

- Insertion
  - An insertion takes $O(\log n)$ time.
  - Restructuring up the tree, maintaining heights is $O(\log n)$.

- Deletion
  - A deletion takes $O(\log n)$ time.
  - Restructuring up the tree, maintaining heights is $O(\log n)$.