

CSC 280
Data Structures

AVL Trees

Instructor
Sameh Elsharkawy, Ph.D.

E-mail: Elsharkawy@cua.edu

Office: Pangborn 311

Tel: (202) 319-4620

AVL Trees

- **AVL Tree Definition**
- AVL Tree Operations
 - Find
 - Insertion
 - Deletion
- AVL Tree Depth
- AVL Tree Performance

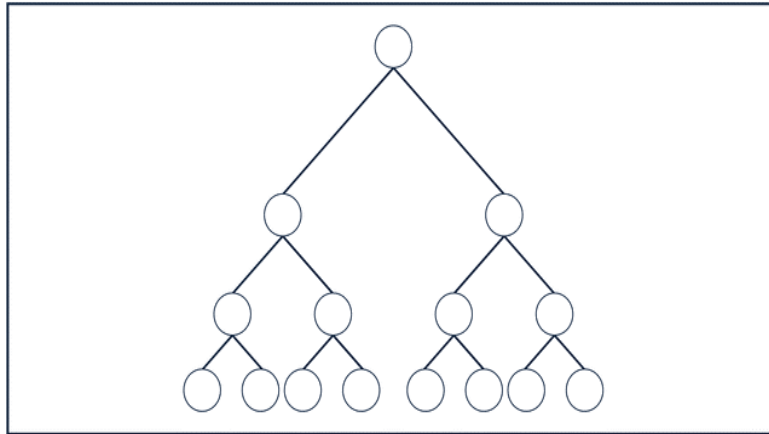
AVL Tree Definition

- **Motivation:** we want to **guarantee** $O(\log n)$ running time on the find/insert/remove operations.
- **Idea:** keep the tree balanced after each operation.
- **Solution:** AVL (Adelson-Velskii and Landis) trees.
- **AVL tree property:** for every node in the tree, the height of the left and right subtrees differs by at most 1.

AVL Tree Definition

- A **perfectly balanced** binary tree is a binary tree such that:
 - The height of the left and right subtrees of the root are equal
 - The left and right subtrees of the root are perfectly balanced binary trees

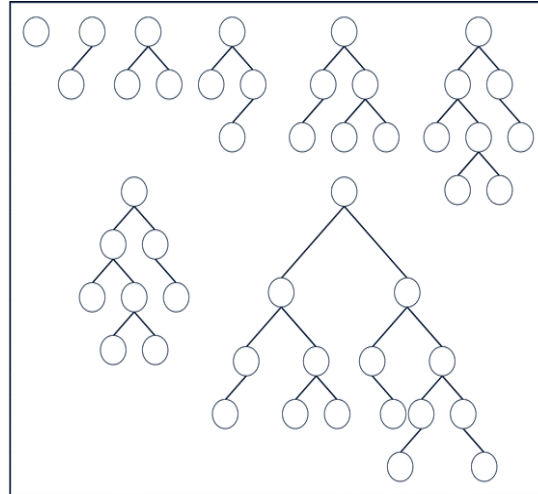
Perfectly Balanced Binary Tree



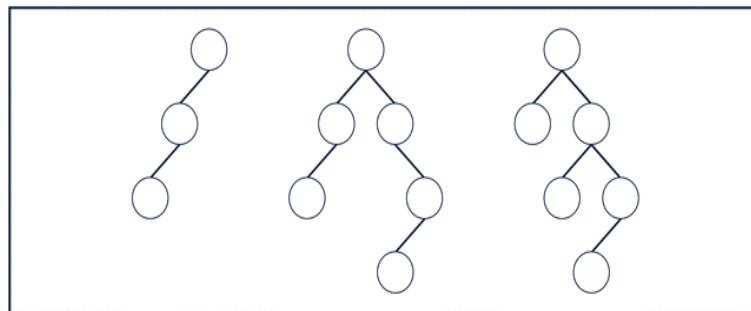
AVL Tree Definition

- An **AVL tree** (or **height-balanced tree**) is a binary search tree such that:
 - The height of the left and right subtrees of the root differ by at most 1
 - The left and right subtrees of the root are AVL trees

AVL Trees



Non-AVL Trees



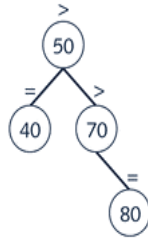
AVL Trees

- AVL Tree Definition
- **AVL Tree Operations**
 - Find
 - Insertion
 - Deletion
- AVL Tree Depth
- AVL Tree Performance

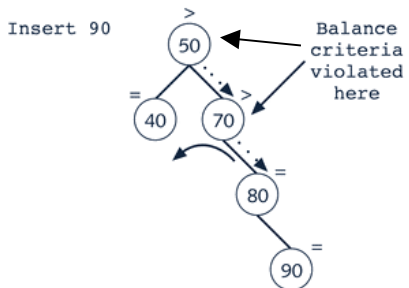
AVL Tree Operations

- AVL tree **Find** is the same as BST find.
- AVL tree **Insert/Delete**: same as BST insert/delete, except that we might have to “fix” the AVL tree after an insert.
- These operations will take time $O(d)$, where d is the depth of the node being found/inserted.
- What is the maximum height of an n -node AVL tree?

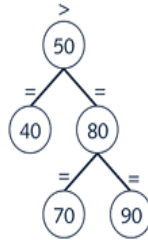
AVL Tree Insertion



AVL Tree Insertion

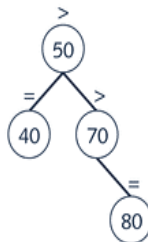


AVL Tree Insertion

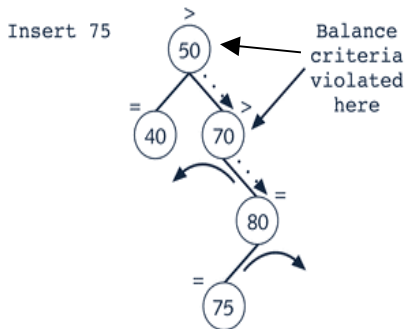


AVL Tree Balance Property Restored

AVL Tree Insertion



AVL Tree Insertion



AVL Tree Rotations

- Reconstruction procedure: **rotating** tree
- **left rotation** and **right rotation**
- Suppose that the rotation occurs at node x
- **Left rotation:** certain nodes from the right subtree of x move to its left subtree; the root of the right subtree of x becomes the new root of the reconstructed subtree
- **Right rotation:** certain nodes from the left subtree of x move to its right subtree; the root of the left subtree of x becomes the new root of the reconstructed subtree

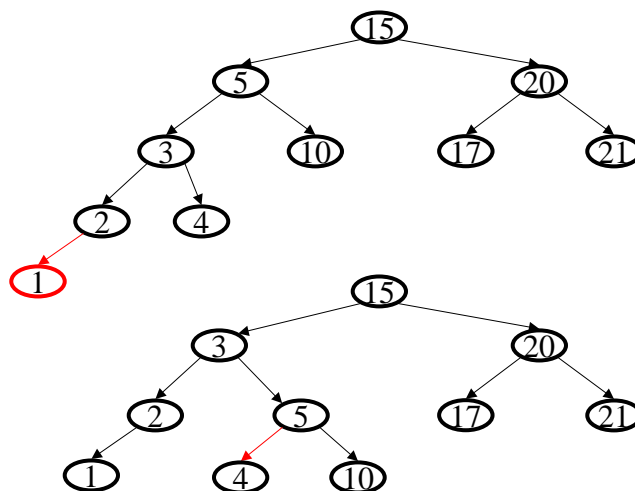
AVL Tree Insertion

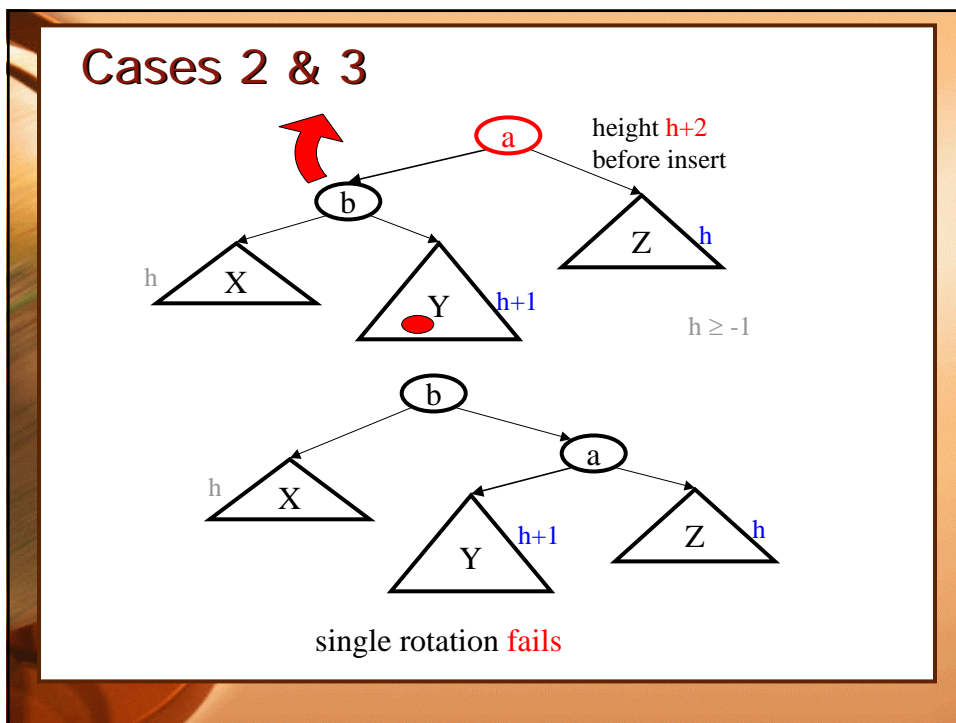
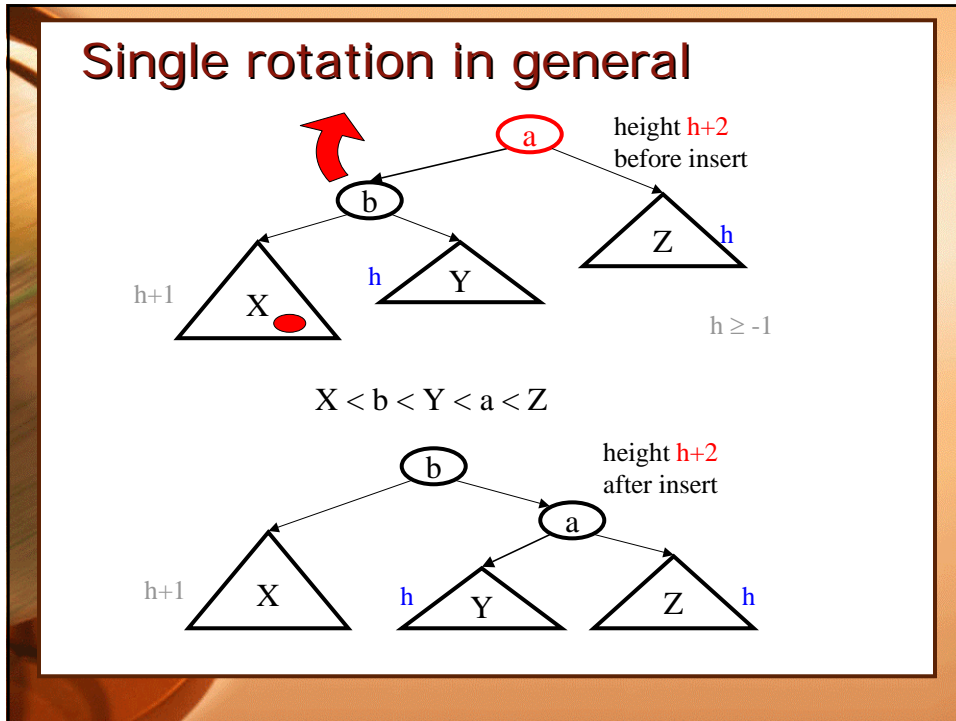
- Let x be the **deepest** node where an imbalance occurs.
- Four cases to consider. The insertion is in the
 - left sub-tree of the left child of x .
 - right sub-tree of the left child of x .
 - left sub-tree of the right child of x .
 - right sub-tree of the right child of x .

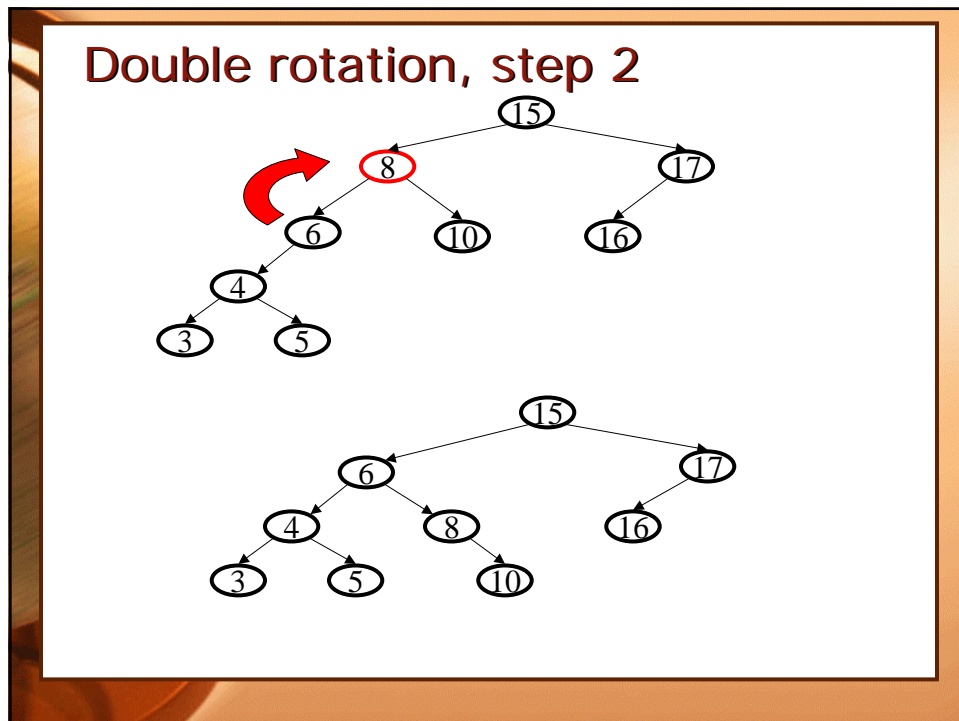
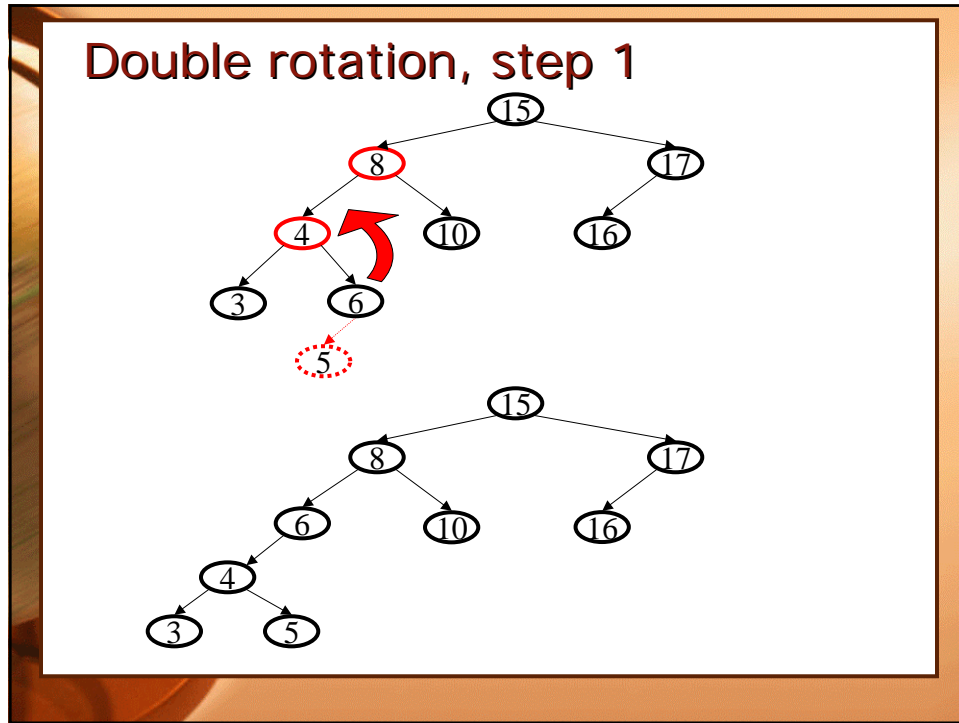
Idea: Cases 1 & 4 are solved by a **single rotation**.

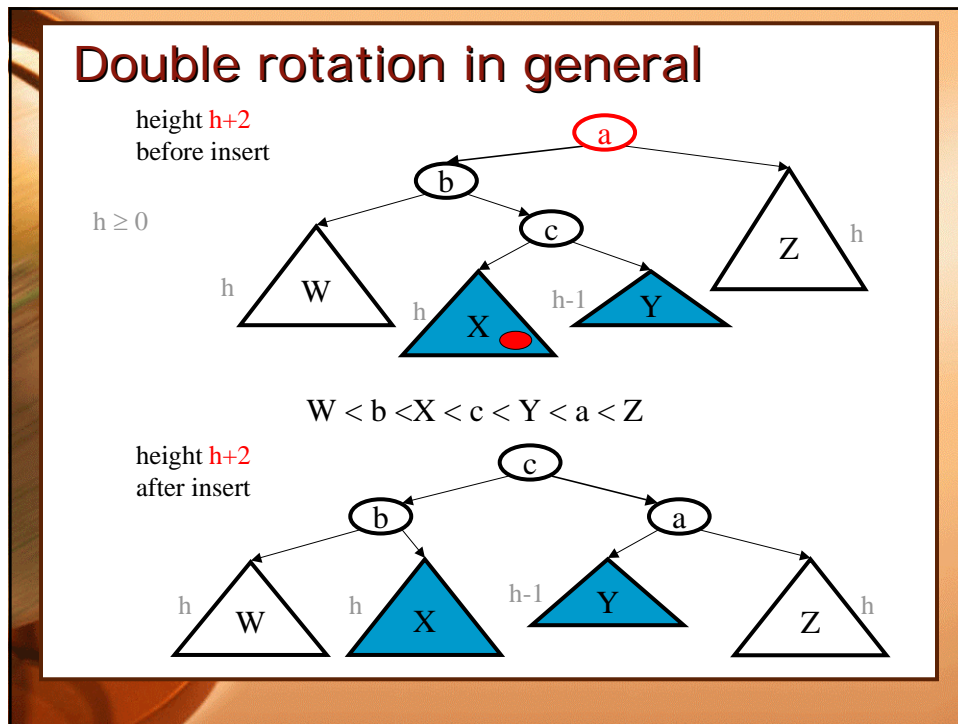
Cases 2 & 3 are solved by a **double rotation**.

Single rotation example









AVL Tree Deletion

- Similar to BST deletion
- After deletion, AVL property is restored using rotation
- Imbalance cases are similar to the insertion operation

AVL Trees

- AVL Tree Definition
- AVL Tree Operations
 - Find
 - Insertion
 - Deletion
- **AVL Tree Depth**
- AVL Tree Performance

Depth of an AVL Tree

Theorem: Any AVL tree with n nodes has height less than

$$1.441 \log n$$

Proof: Given an n -node AVL tree, we want to find an upper bound on the height of the tree.

Fix h . What is the smallest n such that there is an AVL tree of height h with n nodes?

Let S_h be the set of all AVL trees of height h that have as few nodes as possible.

Depth of an AVL Tree (Cont)

Let w_h be the number of nodes in any one of these trees.

$$w_0 = 1, w_1 = 2$$



Suppose $T \in S_h$, where $h \geq 2$. Let T_L and T_R be T 's left and right subtrees. Since T has height h , either T_L or T_R has height $h-1$. Suppose it's T_R .

By definition, both T_L and T_R are AVL trees. In fact, $T_R \in S_{h-1}$ or else it could be replaced by a smaller AVL tree of height $h-1$ to give an AVL tree of height h that is smaller than T .

Depth of an AVL Tree (Cont)

Similarly, $T_L \in S_{h-2}$.

Therefore, $w_h = 1 + w_{h-2} + w_{h-1}$.

Claim: For $h \geq 0$, $w_h \geq \phi^h$, where $\phi = (1 + \sqrt{5}) / 2 \approx 1.6$.

Proof: The proof is by induction on h .

Basis step: $h=0$. $w_0 = 1 = \phi^0$.

$h=1$. $w_1 = 2 > \phi^1$.

Induction step: Suppose the claim is true for $0 \leq m \leq h$, where $h \geq 1$.

Depth of an AVL Tree (Cont)

Then

$$\begin{aligned}
 w_{h+1} &= 1 + w_{h-1} + w_h \\
 &\geq 1 + \varphi^{h-1} + \varphi^h \quad (\text{by the i.h.}) \\
 &= 1 + \varphi^{h-1} (1 + \varphi) \\
 &= 1 + \varphi^{h+1} \quad (1 + \varphi = \varphi^2) \\
 &> \varphi^{h+1}
 \end{aligned}$$

Thus, the claim is true.

From the claim, in an n -node AVL tree of height h ,

$$n \geq w_h \geq \varphi^h \quad (\text{from the Claim})$$

$$\begin{aligned}
 h &\leq \log_{\varphi} n \\
 &= (\log n) / (\log \varphi) \\
 &< 1.441 \log n
 \end{aligned}$$

AVL Trees

- AVL Tree Definition
- AVL Tree Operations
 - Find
 - Insertion
 - Deletion
- AVL Tree Depth
- **AVL Tree Performance**

AVL Tree: Running times

- **Find** takes $O(\log n)$ time, because height of the tree is always $O(\log n)$.
- **Insert**: $O(\log n)$ time because we do a find ($O(\log n)$ time), and then we may have to visit every node on the path back to the root, performing up to 2 single rotations ($O(1)$ time each) to fix the tree.
- **Delete**: $O(\log n)$ time.