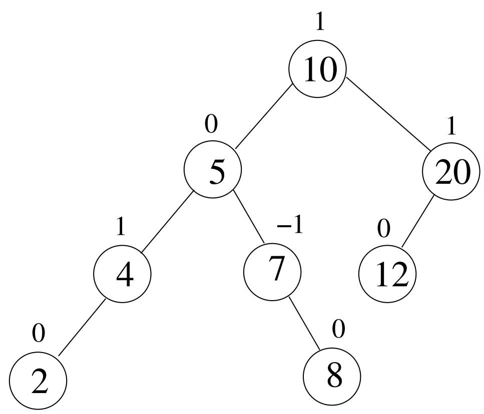
AVL Trees - a Balanced Binary Search Tree

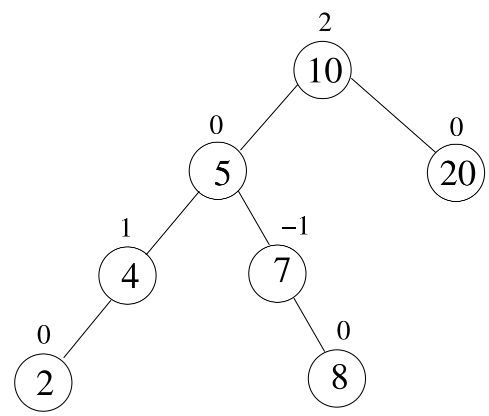
Balanced Binary Search Trees

- **What is a Balanced Binary Search Trees:**
  It is a binary search tree with height $\Theta(\log n)$, where n is the # of nodes in the tree.

- **AVL Trees:**
  - An AVL tree is a BST.
  - For each node, the difference between the height of its left and right subtrees is either $+1, 0$, or -1.
    Ex:



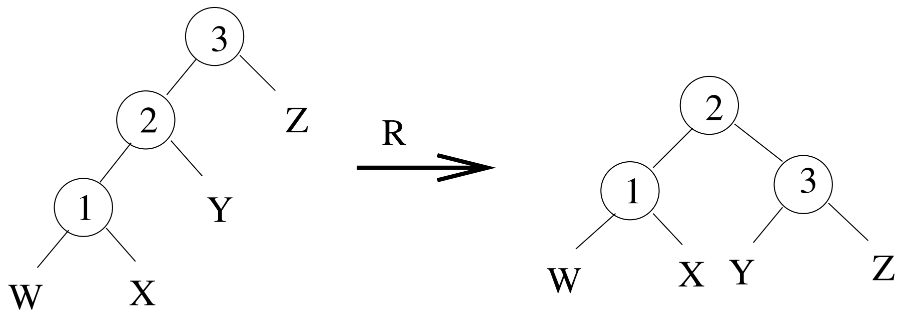
An AVL tree



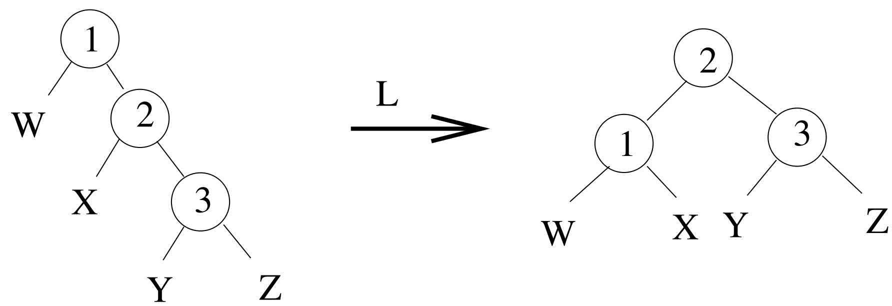
not an AVL tree

  - Technique to maintain AVL tree: 4 different rotations.
    See next page.

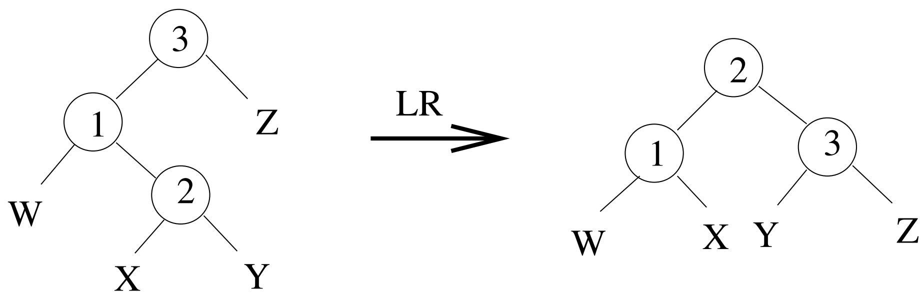
R Rotation:



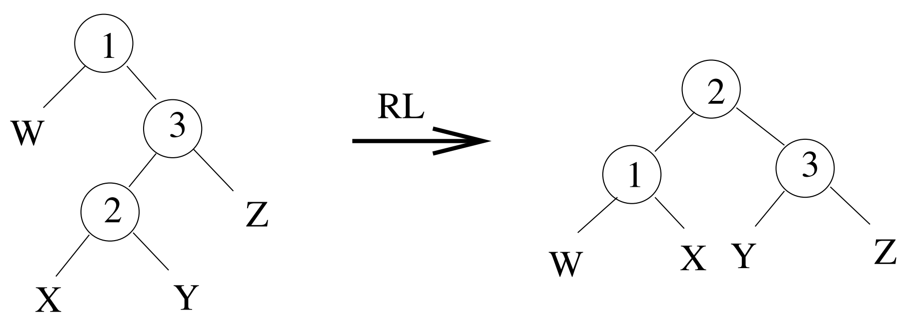
L Rotations:



LR Rotation:



RL Rotation:



– **AVL Insertion:** Two steps.

1. Perform the BST-Insert(T, z).
2. Perform re-structuring through rotations if necessary.
   - ∗ A rotation is performed when a subtree rooted at a node whose **balanced factor** becomes $+2$ or -2 after BST-Insert. If there are several such unbalanced nodes, we rotate at a node A that is closest to the newly inserted leaf.
   - ∗ To decide which rotation to use, we start from the node A and test the balanced factor in the following way.

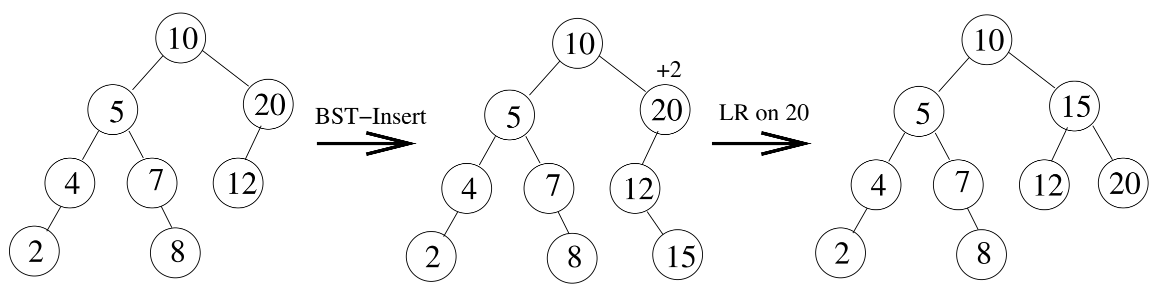
     $+2 \xrightarrow{L} +1/0 \xrightarrow{L}$: R Rotation.

     $+2 \xrightarrow{L} -1 \quad \xrightarrow{R}$: LR Rotation.

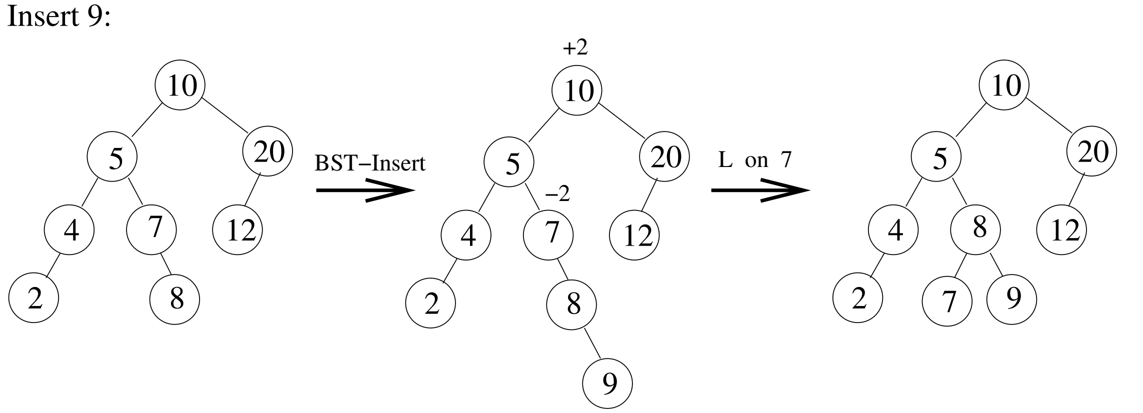
     $-2 \xrightarrow{R} +1 \quad \xrightarrow{L}$: RL Rotation.

     $-2 \xrightarrow{R} -1/0 \xrightarrow{R}$: L Rotation.

Ex:
Insert 15:



Insert 9:

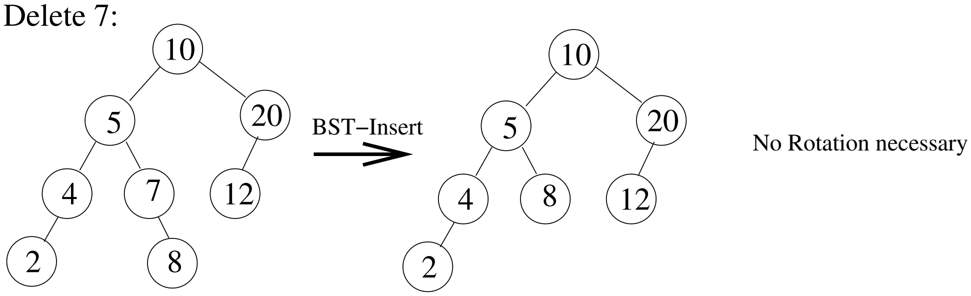


– **AVL Deletion:** Two steps.

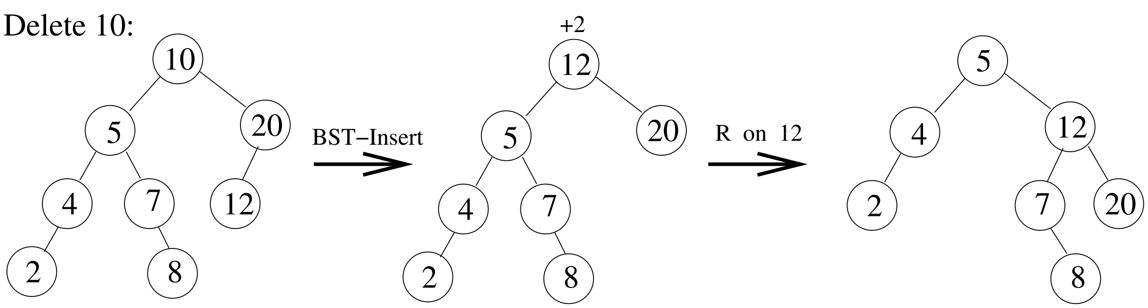
1. Perform the BST-Delete(T, z).
2. Perform re-structuring through rotations if necessary.
   - ∗ A rotation is performed when a subtree rooted at a node whose **balanced factor** becomes $+2$ or -2 after BST-Insert. If there are several such unbalanced nodes, we rotate at a node A that is closest to the newly deleted node (i.e., the node has been physically removed).
   - ∗ To decide which rotation to use, we start from the node A and test the balanced factor in the following way.

     $+2 \xrightarrow{L} +1/0 \xrightarrow{L}$: R Rotation.
     $+2 \xrightarrow{L} -1 \quad \xrightarrow{R}$: LR Rotation.
     $-2 \xrightarrow{R} +1 \quad \xrightarrow{L}$: RL Rotation.
     $-2 \xrightarrow{R} -1/0 \xrightarrow{R}$: L Rotation.

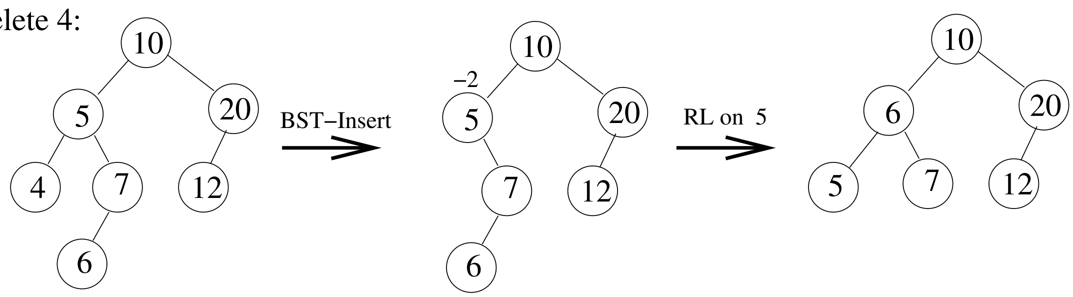
Delete 7:



Delete 10:



Delete 4:



Ex:

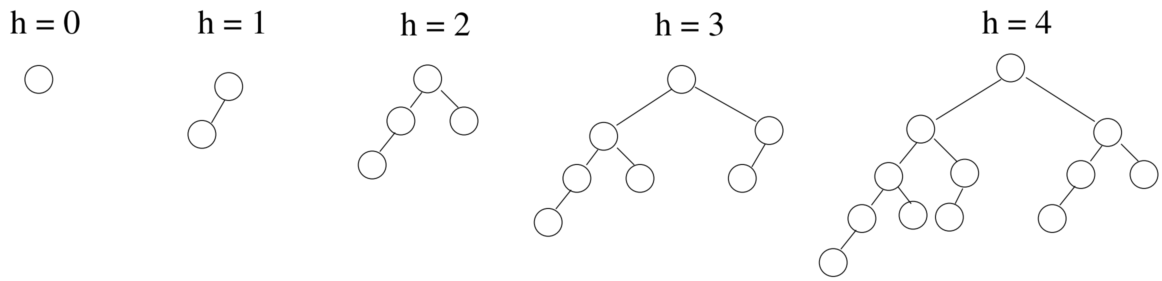
– **Height of AVL Trees:** $\Theta(\log n)$

Given an AVL tree with height h.

The maximum # of nodes: the tree is full.

$$
\begin{aligned}
&\quad n \le 2^0 + 261 + \ldots + 2^h = \textstyle\sum_{i=0}^{h} 2^i = 2^{h+1} - 1 \\
&\Longrightarrow n \le 2^{h+1} - 1 \\
&\Longrightarrow h \ge \log(n+1) - 1 \\
&\qquad = \Omega(\log n)
\end{aligned}
$$

The minimum # of nodes:



Let n_h be the minimum # of nodes of an AVL tree with height h

$$
n_h = \begin{cases} 1 & \text{if } h = 0 \\ 2 & \text{if } h = 1 \\ n_{h-1} + n_{h-2} + 1 & \text{if } h \ge 2 \end{cases}
$$

Recall the Fibonacci numbers as follows.

$$
F_h = \begin{cases} 0 & \text{if } h = 0 \\ 1 & \text{if } h = 1 \\ F_{h-1} + F_{h-2} & \text{if } h \ge 2 \end{cases}
$$

Thus, we have $n_h > F_h, \forall\, h \ge 0$.

Since $F_h = \frac{\phi^h - \overline{\phi}^h}{\sqrt{5}}$, where $\phi = 1.61803$ and $\overline{\phi} = -0.61803$.

$$
\begin{aligned}
&\quad n_h > F_h = \tfrac{\phi^h - \overline{\phi}^h}{\sqrt{5}} \simeq \tfrac{\phi^h}{\sqrt{5}} \quad \text{if } h \text{ is large} \\
&\Longrightarrow \quad n \ge n_h > \tfrac{\phi^h}{\sqrt{5}} \\
&\Longrightarrow \quad \log_\phi(\sqrt{5} \cdot n) > h \\
&\Longrightarrow \quad h < \log_\phi \sqrt{5} + \log_\phi n \\
&\qquad\quad = O(\log n)
\end{aligned}
$$

Therefore, the height of any AVL tree is $\Theta(\log n)$.