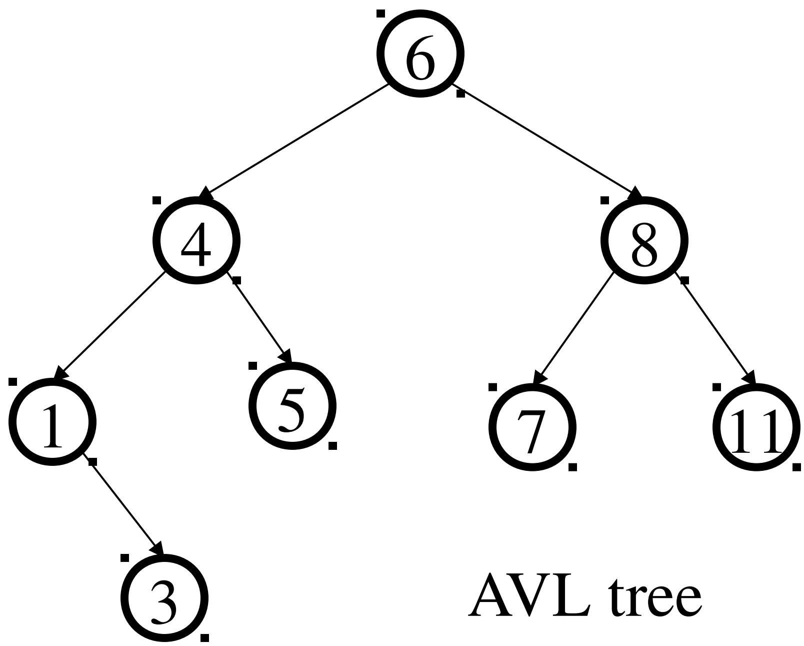
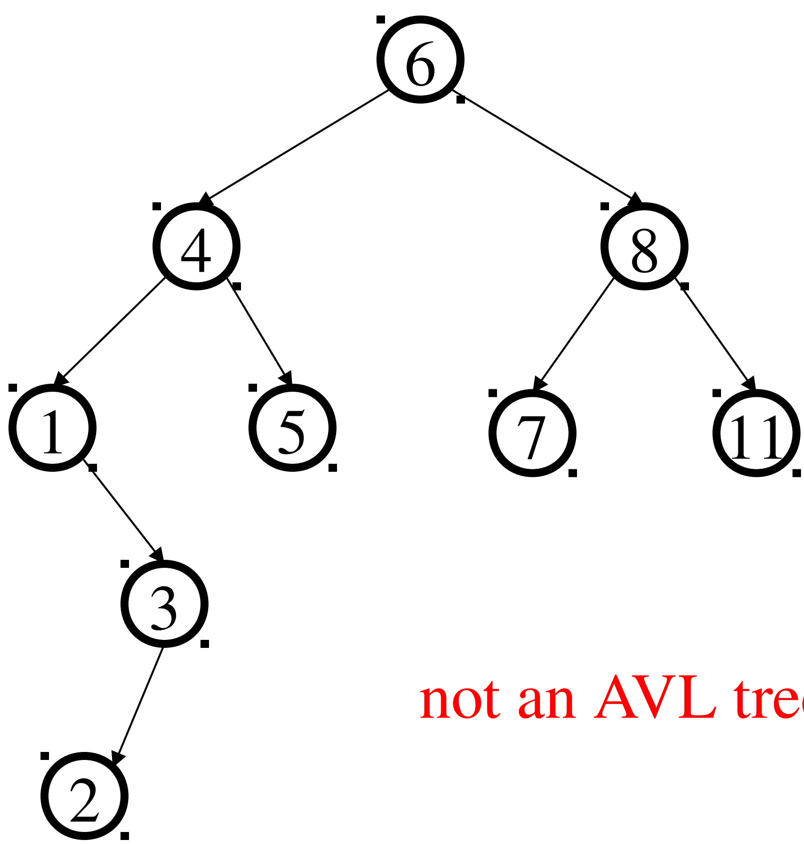
AVL Trees

- 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



not an AVL tree

AVL trees: find, insert

- AVL tree find is the same as BST find.
- AVL insert: same as BST insert, 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?

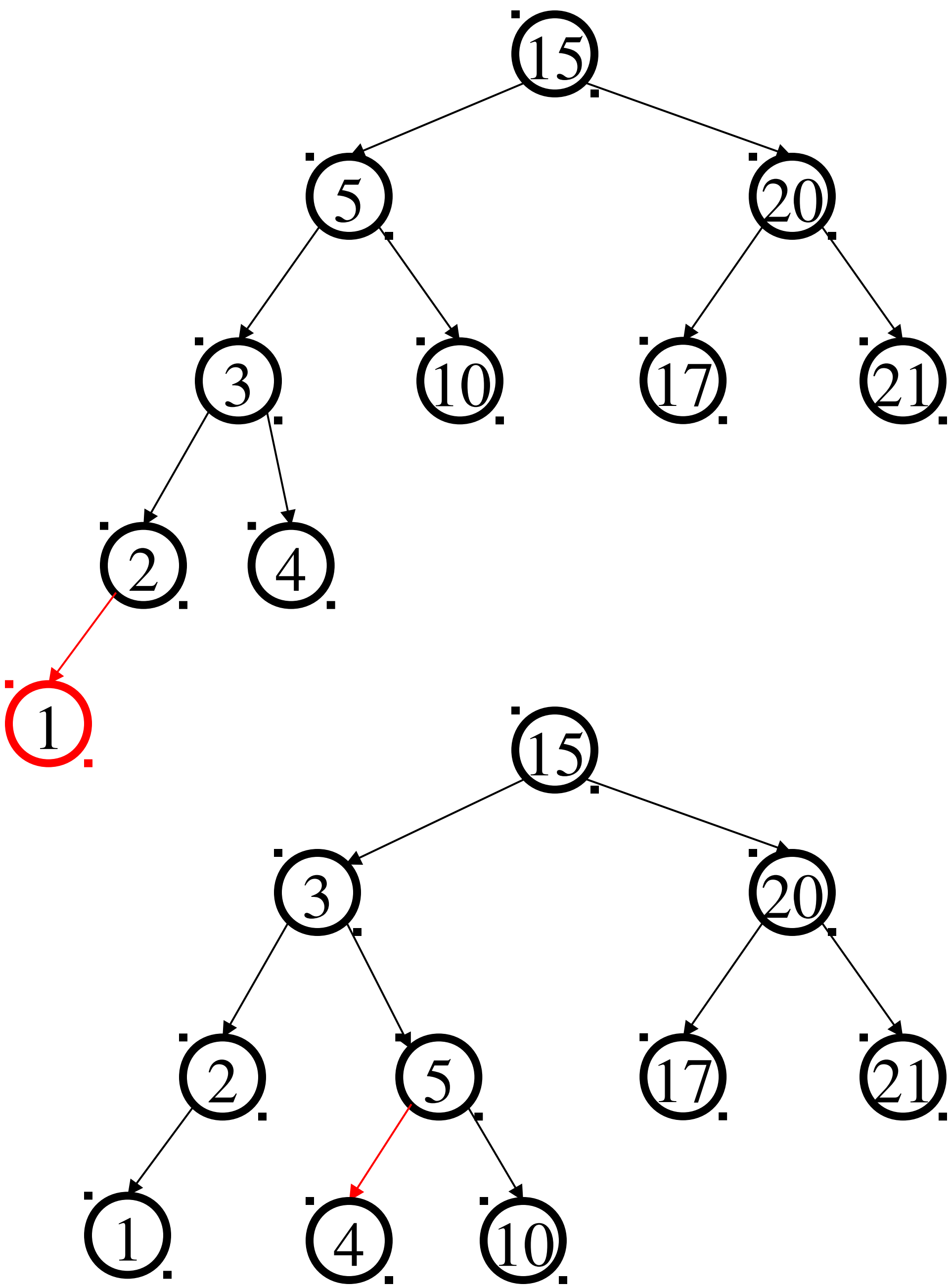
AVL tree insert

- Let x be the deepest node where an imbalance occurs.
- Four cases to consider. The insertion is in the
  1. left subtree of the left child of x.
  2. right subtree of the left child of x.
  3. left subtree of the right child of x.
  4. right subtree of the right child of x.

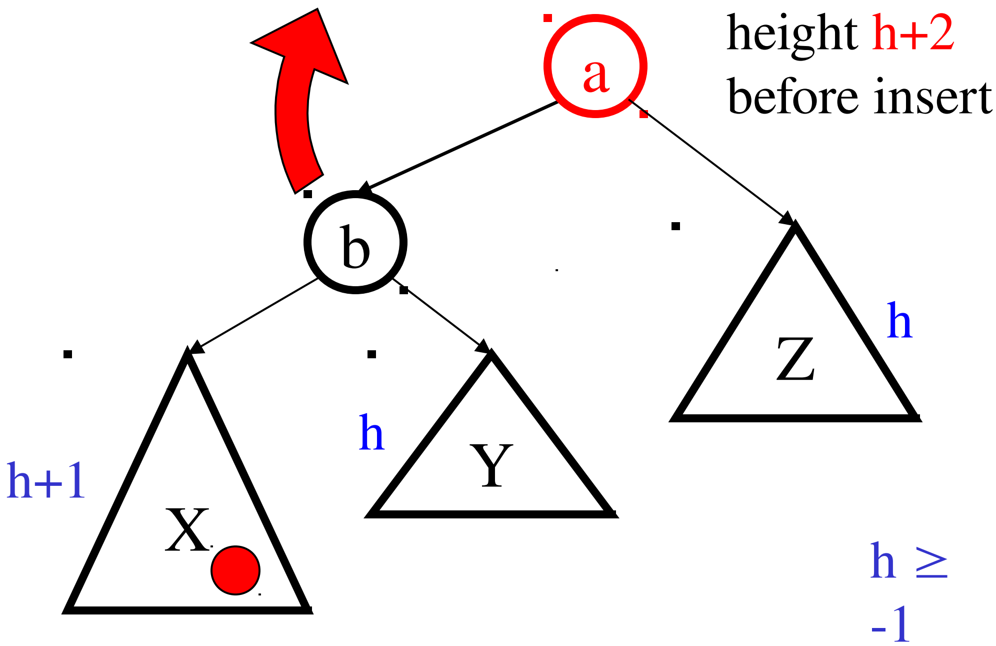
Idea: Cases 1 & 4 are solved by a single rotation.

Cases 2 & 3 are solved by a double rotation.

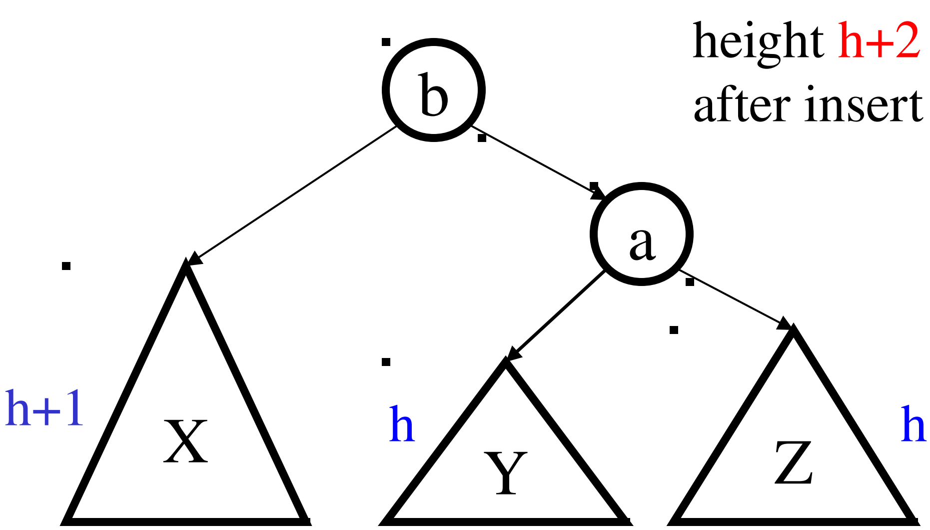
Single rotation example



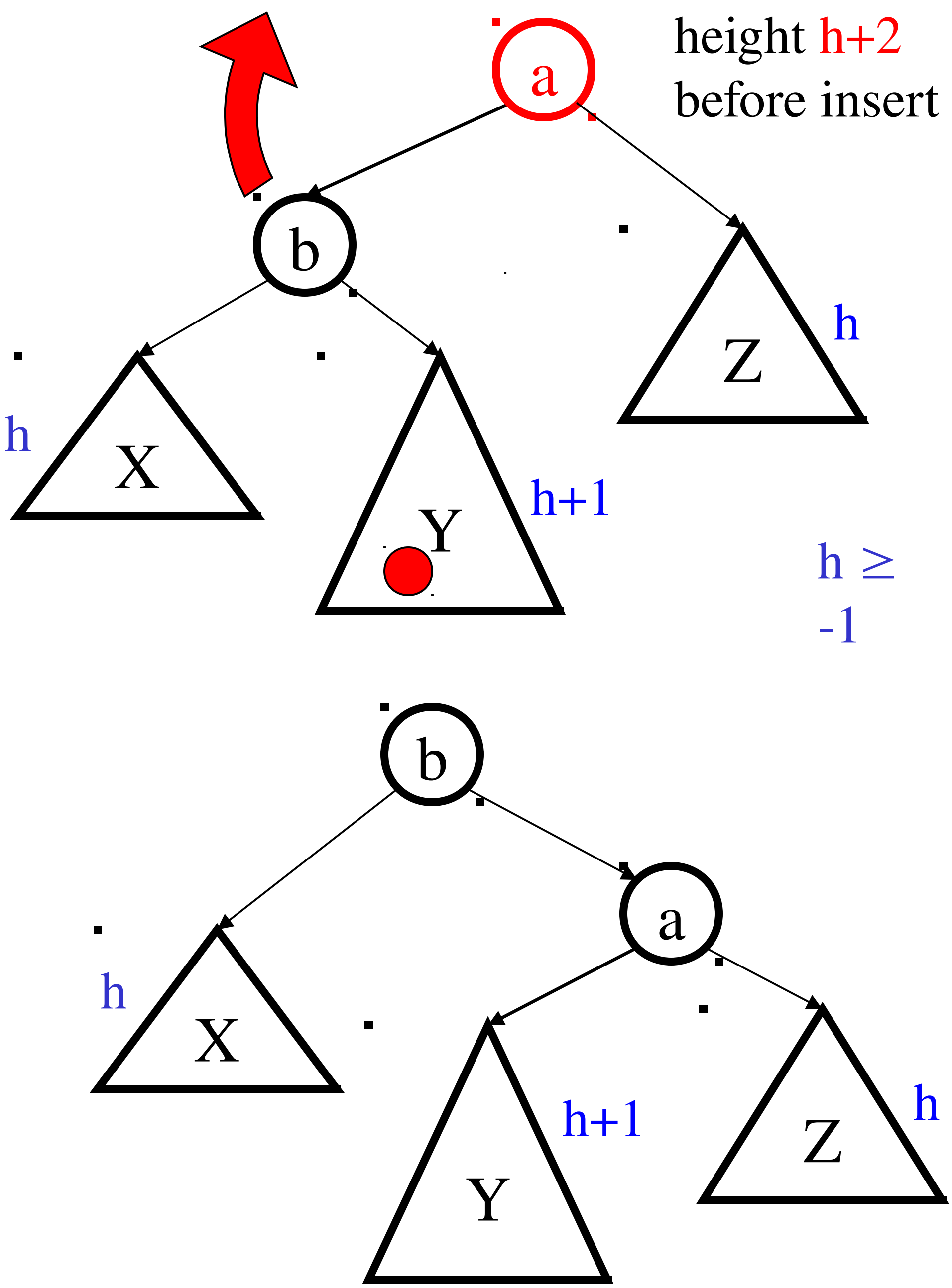
Single rotation in general



X < b < Y < a < Z

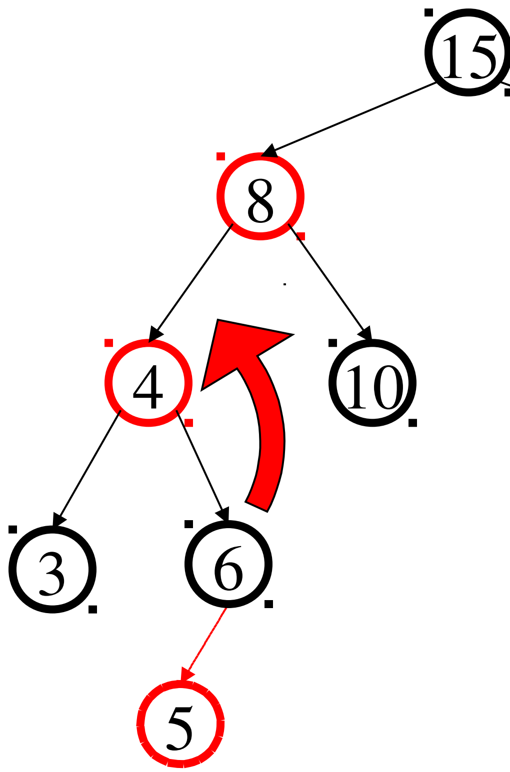


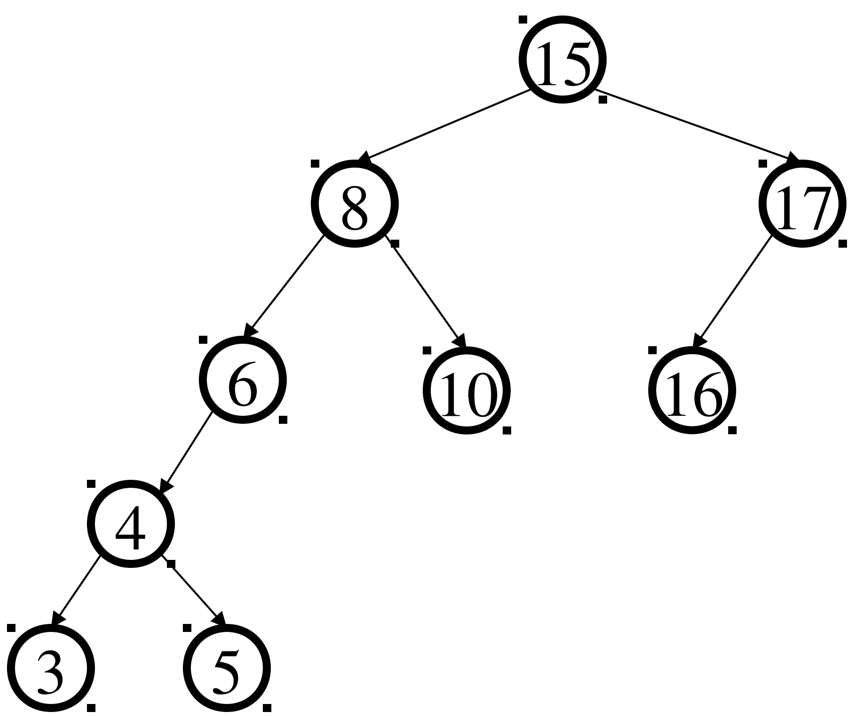
Cases 2 & 3



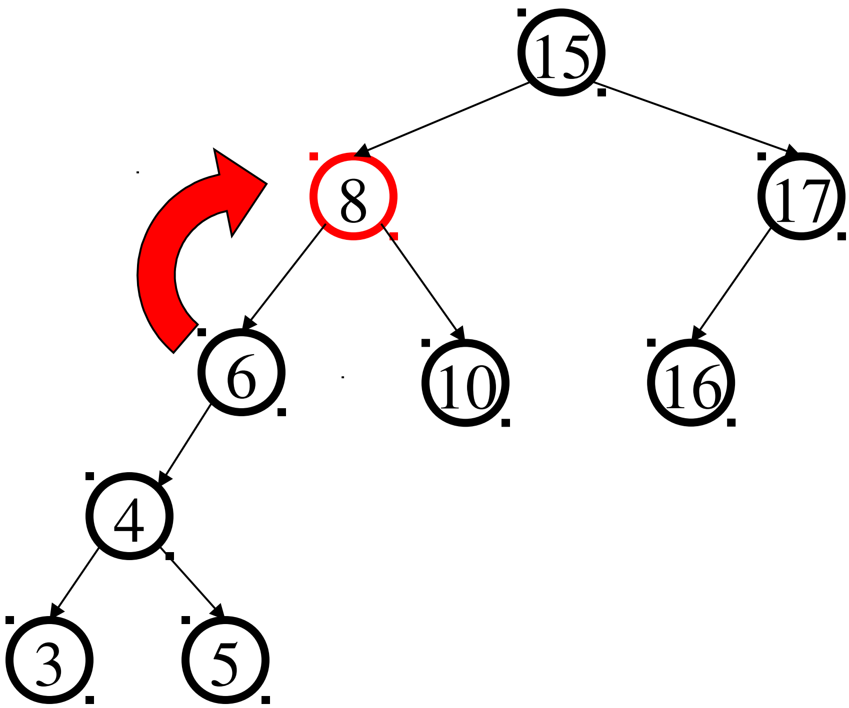
single rotation fails

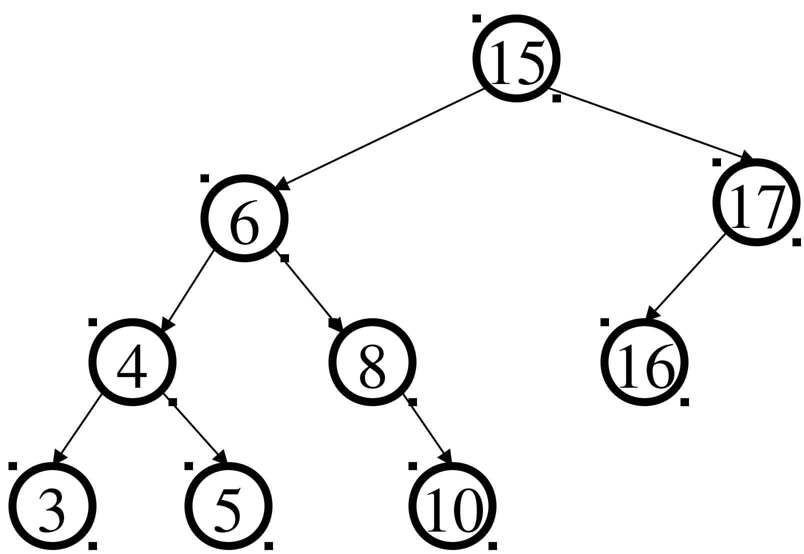
Double rotation, step 1





Double rotation, step 2

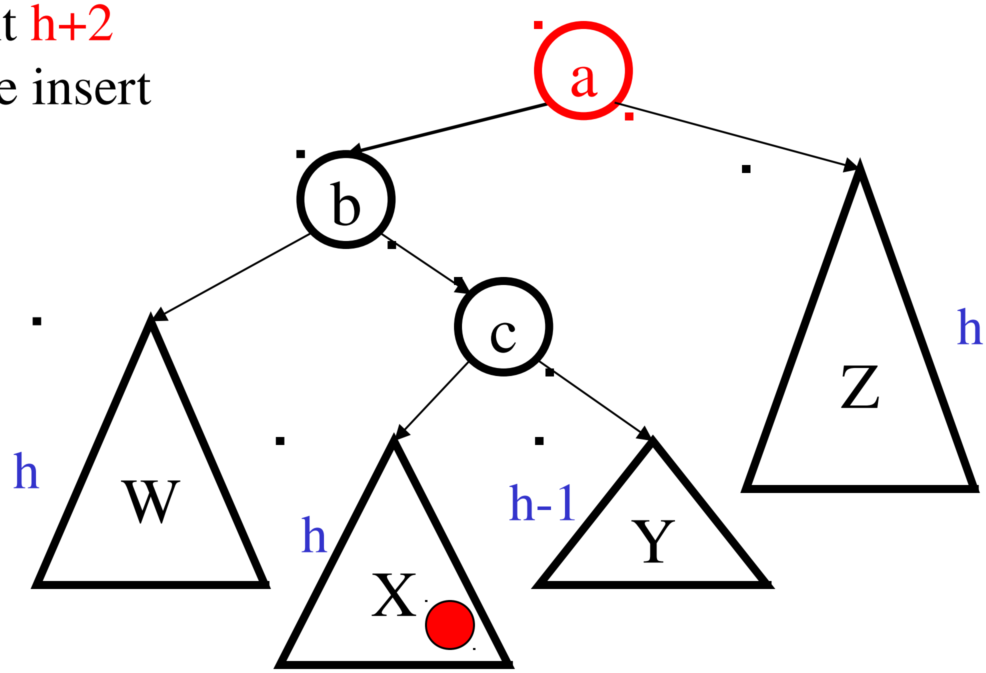




Double rotation in general

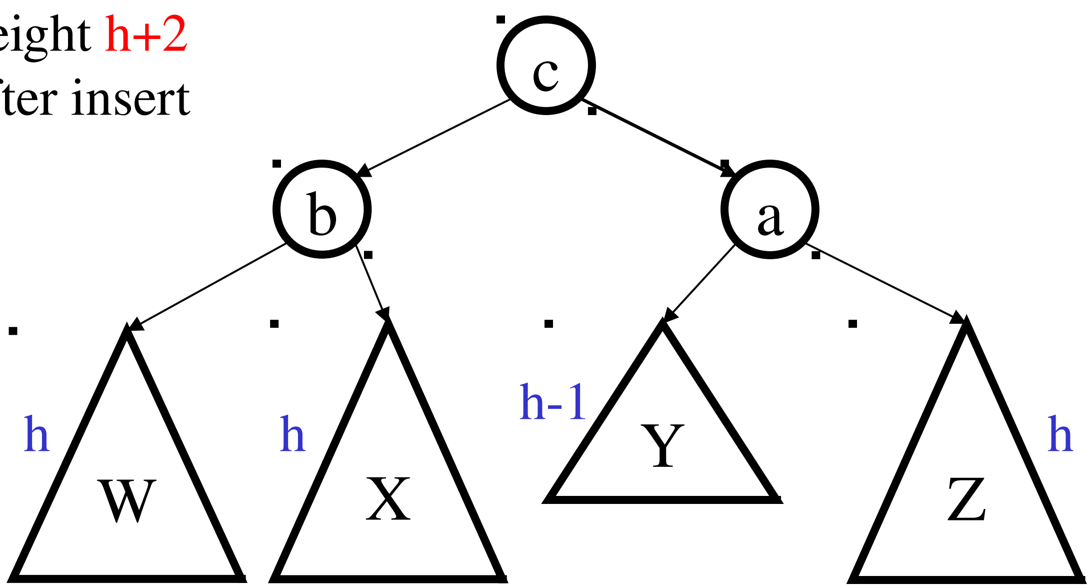
height h+2
before insert

h ≥ 0



W < b <X < c < Y < a < Z

height h+2
after insert



Depth of an AVL tree

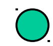
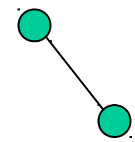
Theorem: Any AVL tree with n nodes has height less than $1.441 \log n$.

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.

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.

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

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

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

Proof: The proof is by induction on *h*.

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

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

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

Then

$w_{h+1} = 1 + w_{h-1} + w_h$

$\geq 1 + \varphi^{h-1} + \varphi^h$ (by the i.h.)

$= 1 + \varphi^{h-1} (1 + \varphi)$

$= 1 + \varphi^{h+1}$ $(1+\varphi = \varphi^2)$

$> \varphi^{h+1}$

Thus, the claim is true.

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

$n \geq w_h \geq \varphi^h$ (from the Claim)

$h \leq \log_\varphi n$

$= (\log n) / (\log \varphi)$

$< 1.441 \log n$

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.
- remove: O(log *n*) time. Left as an exercise.