Binary Trees (cont.)

Instructor: Dr. Sunho Lim (Ph.D., Assistant Professor)

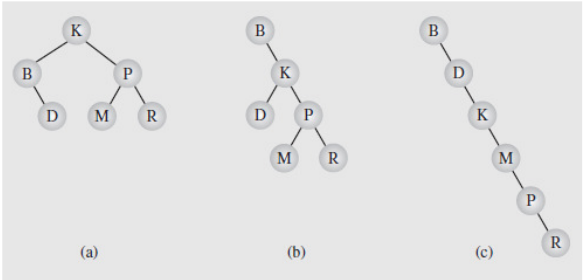
Lecture 10

sunho.lim@ttu.edu

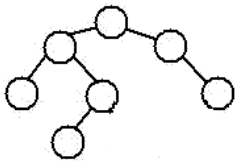
Adapted partially from Data Structures and Algorithms in C++, Adam Drozdek, 4th Edition, Cengage Learning; and Algorithms and Data Structures, Douglas Wilhelm Harder, Mmath

Balancing a Tree

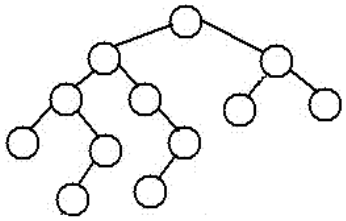
- In favor of trees,
  - represent hierarchical data particularly well
  - searching trees is much faster than searching lists
    - depends on the **structure of the tree**
    - e.g., skewed trees search no better than linear lists



Balancing a Tree



A height-balanced Tree

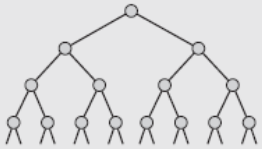


Not a height-balanced tree

- A binary tree is ***height balanced*** (or simply, ***balanced***)
  - if the difference in **height** of the subtrees of any node in the tree is **zero or one**

Balancing a Tree (cont.)

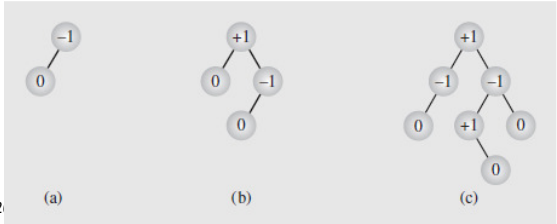
- Maximum number of nodes that can be stored in binary trees of different heights:



Height	Nodes at One Level	Nodes at All Levels
1	$2^0 = 1$	$1 = 2^1 - 1$
2	$2^1 = 2$	$3 = 2^2 - 1$
3	$2^2 = 4$	$7 = 2^3 - 1$
4	$2^3 = 8$	$15 = 2^4 - 1$
⋮		
11	$2^{10} = 1{,}024$	$2{,}047 = 2^{11} - 1$
⋮		
14	$2^{13} = 8{,}192$	$16{,}383 = 2^{14} - 1$
⋮		
h	2^{h-1}	$n = 2^h - 1$
⋮		

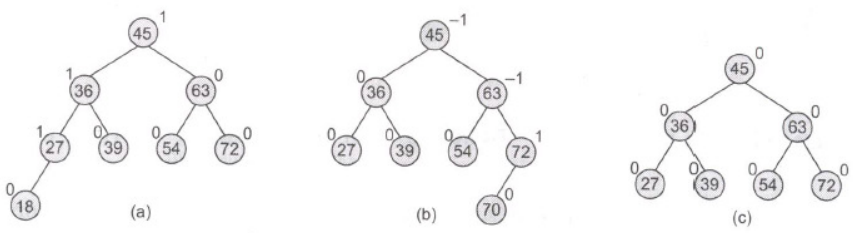
AVL Trees

- Rebalancing can be performed locally,
  - if the insertions or deletions impact only a **portion** of the tree
- An ***AVL tree*** (also called an ***admissible tree***)
  - the height of the left and right subtrees differ by **at most one**
  - the ***balance factors***,
    - the difference between the height of the right and left subtrees and should be +1, 0, or -1
    - balance factor = height (right subtree) – height (left subtree)



AVL Trees (cont.)

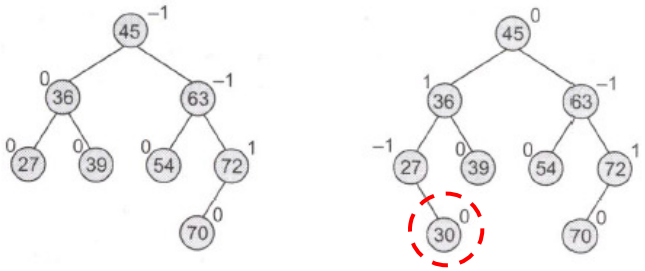
- Example of AVL trees,
  - Here, balance factor = height (**left** subtree) – height (**right** subtree)



- Searching for a node
  - exactly the same way with a binary search tree

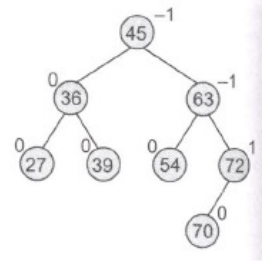
AVL Trees (cont.)

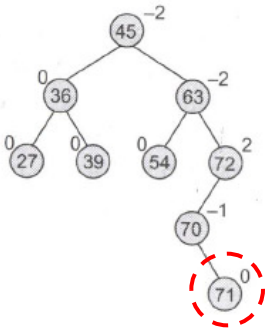
- Inserting a new node
  - if not disturb the **balance factor**
    - don't do anything
    - e.g., inserting 30



AVL Trees (cont.)

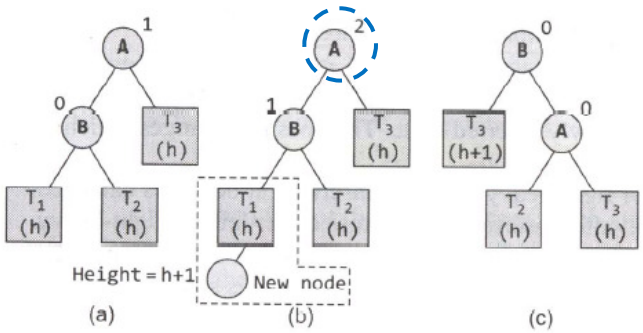
- Inserting a new node (cont.)
  - if disturb the balance factor
    - single rotation (LL & RR)
    - double rotation (LR & RL)
    - e.g., inserting 71
  - **critical node**
    - the nearest ancestor node on the path from the inserted node to the root whose balance factor is neither -1, 0, nor 1.





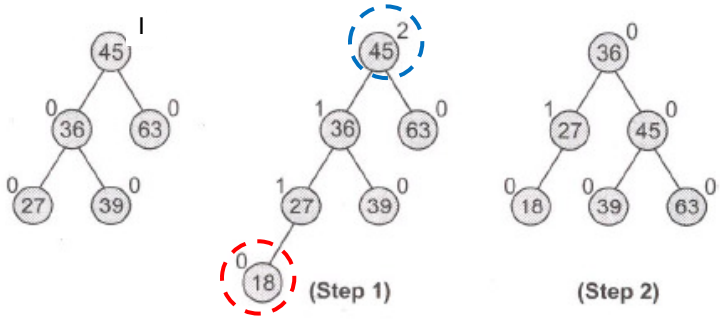
AVL Trees (cont.)

- Inserting a new node (cont.)
  - **single rotation (LL):** inserted in the **left subtree** of the **left subtree** of the critical node



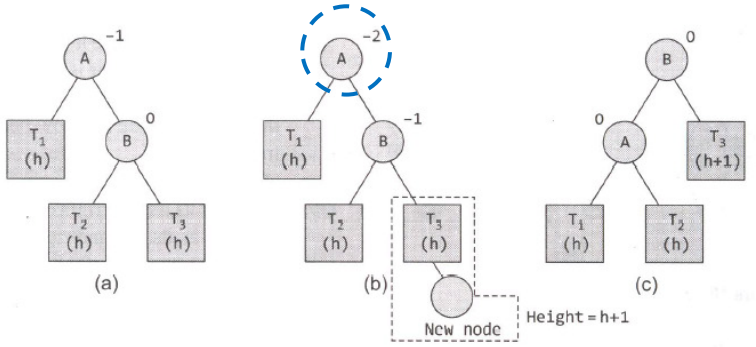
AVL Trees (cont.)

- Inserting a new node (cont.)
  - **single rotation (LL):** inserted in the **left subtree** of the **left subtree** of the critical node (cont.)



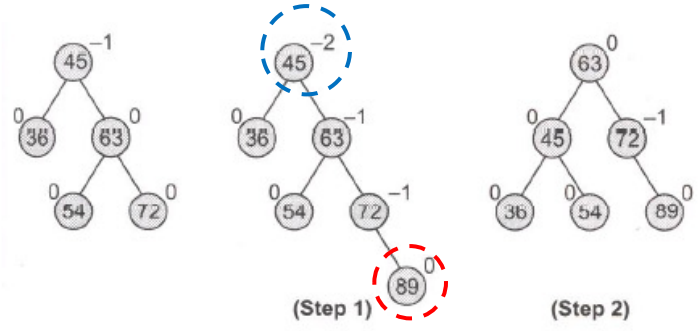
AVL Trees (cont.)

- Inserting a new node (cont.)
  - **single rotation (RR):** inserted in the **right subtree** of the **right subtree** of the critical node



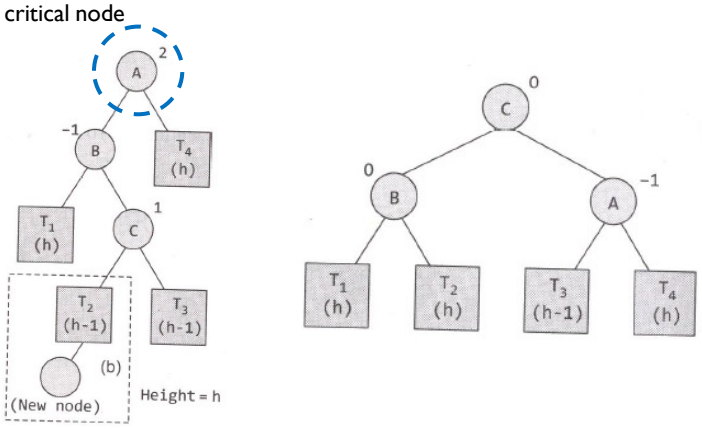
AVL Trees (cont.)

- Inserting a new node (cont.)
  - **single rotation (RR):** inserted in the **right subtree** of the **right subtree** of the critical node (cont.)



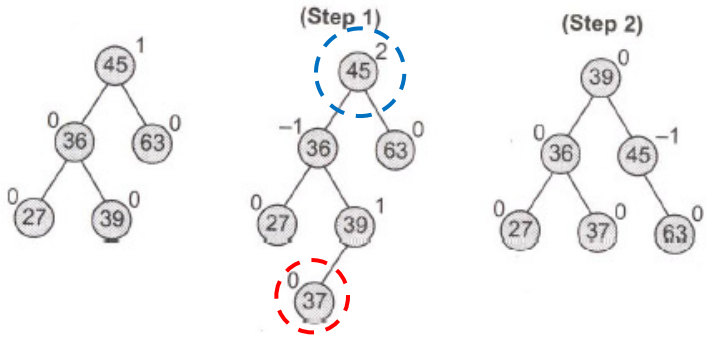
AVL Trees (cont.)

- Inserting a new node (cont.)
  - **double rotation (LR):** inserted in the **right subtree** of the **left subtree** of the critical node



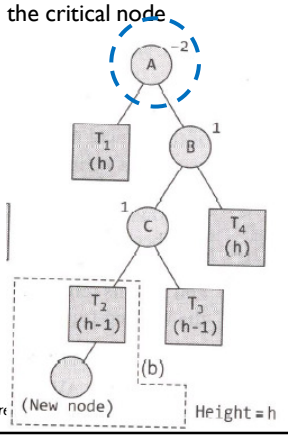
AVL Trees (cont.)

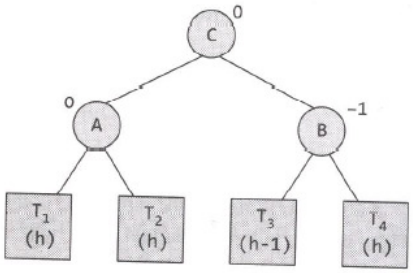
- Inserting a new node (cont.)
  - **double rotation (LR):** inserted in the **right subtree** of the **left subtree** of the critical node (cont.)



AVL Trees (cont.)

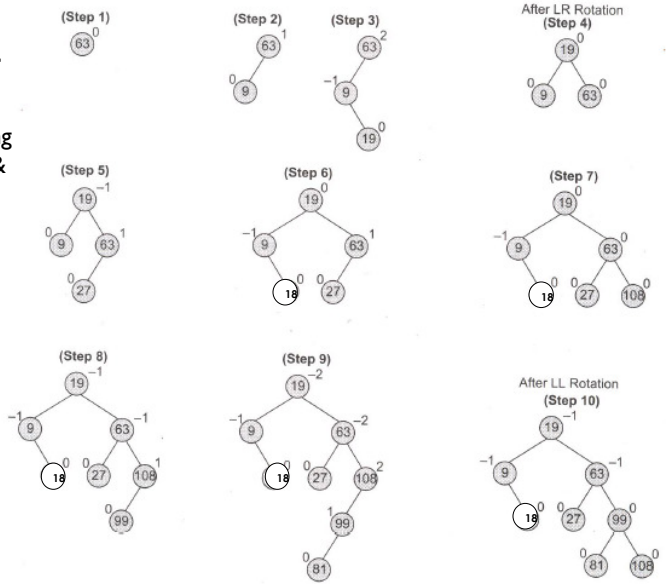
- Inserting a new node (cont.)
  - **double rotation (RL):** inserted in the **left subtree** of the **right subtree** of the critical node





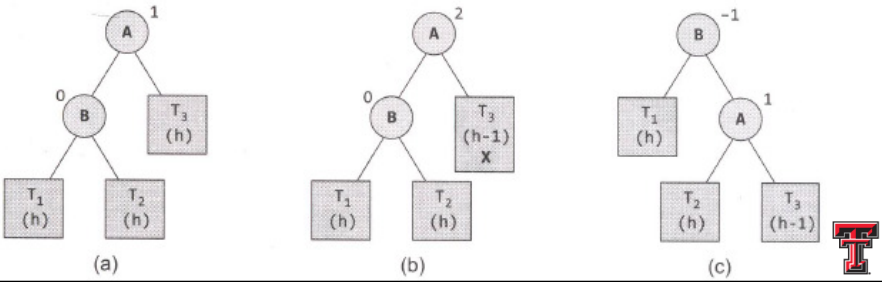
AVL Trees (cont.)

- Construct an AVL tree by inserting the following nodes in a given order & indicate any rotation
  - 63, 9, 19, 27, 18, 108, 99, 81



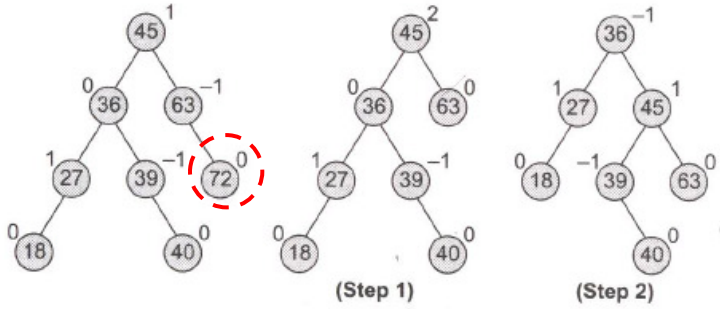
AVL Trees (cont.)

- Deleting a node
  - similar to binary search tree
  - may disturb the **balance factor**
    - **rebalance** the AVL tree
    - rotation (L & R)
- R0 rotation



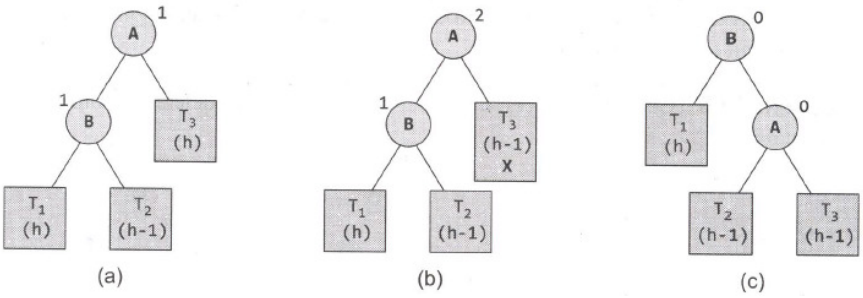
AVL Trees (cont.)

- R0 rotation: (cont.)



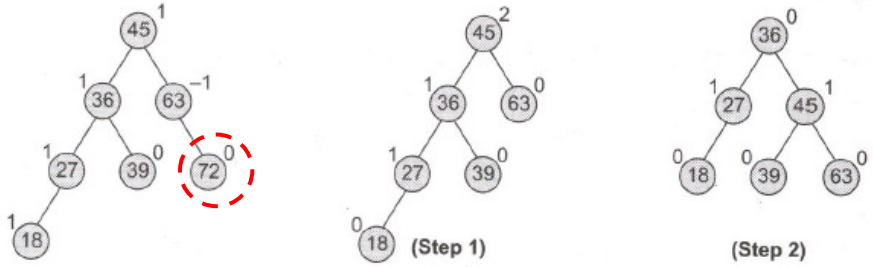
AVL Trees (cont.)

- R1 rotation:



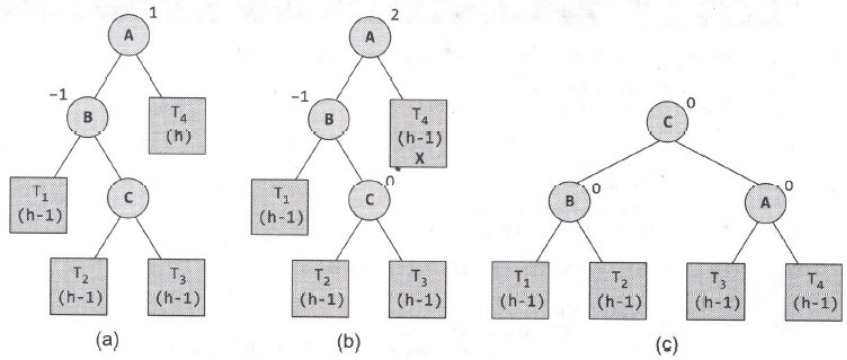
AVL Trees (cont.)

- R1 rotation: (cont.)



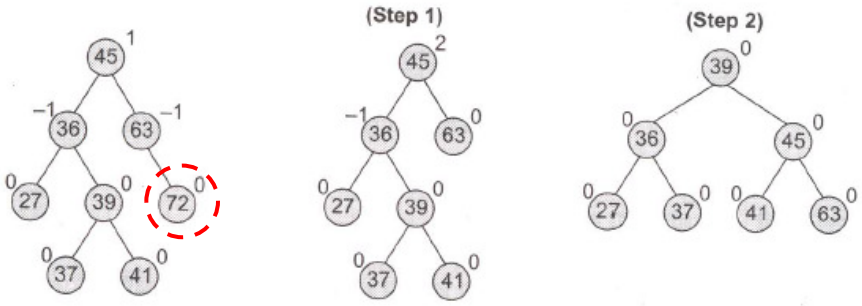
AVL Trees (cont.)

- R-1 rotation:



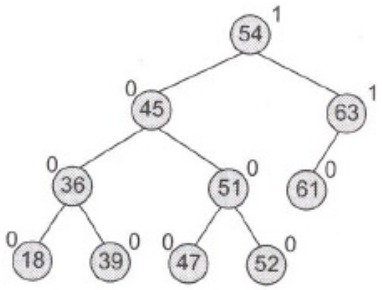
AVL Trees (cont.)

- R-1 rotation: (cont.)



AVL Trees (cont.)

- Delete nodes 52, 36, and 61 from the AVL tree given below,



AVL Trees (cont.)

- Delete nodes 52, 36, and 61 from the AVL tree given below, (cont.)