بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

In the name of Allah, Most Gracious, Most Merciful

CSE 4303
Data Structure

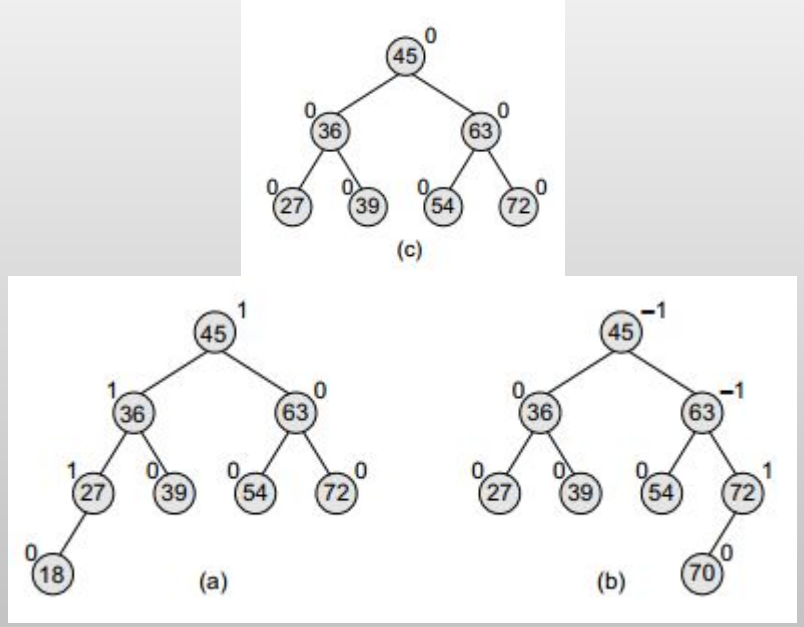
Topic: Self Balancing Tree (AVL) Tree

Asaduzzaman Herok
Lecturer | CSE | IUT
asaduzzaman34@iut-dhaka.edu

Adelson-Velsky and E.M. Landis Tree (AVL)

Self-balancing binary search tree invented by G.M. Adelson-Velsky and E.M. Landis in 1962. Also known as a height-balanced tree.

- The heights of the two sub-trees of a node may differ by at most one.
- The height of the tree is limited to O(log n).
- An additional variable called the BalanceFactor.



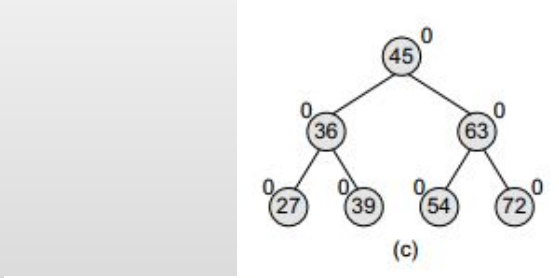
Balance factor
= Height (left sub-tree) – Height (right sub-tree)

Since the height of the tree is limited to O(log(n)), the time needed to search an element in the tree is greatly reduced.

AVL Tree (Balance Factor)

Balance factor = Height (left sub-tree) – Height (right sub-tree)

- **Left-heavy tree:** If the balance factor of a node is 1, then it means that the left sub-tree of the tree is one level higher than that of the right sub-tree.
- **Perfectly balanced Tree:** If the balance factor of a node is 0, then it means that the height of the left sub-tree (longest path in the left sub-tree) is equal to the height of the right sub-tree.
- **Right-heavy tree:** If the balance factor of a node is –1, then it means that the left sub-tree of the tree is one level lower than that of the right sub-tree.



Operations on AvL Trees

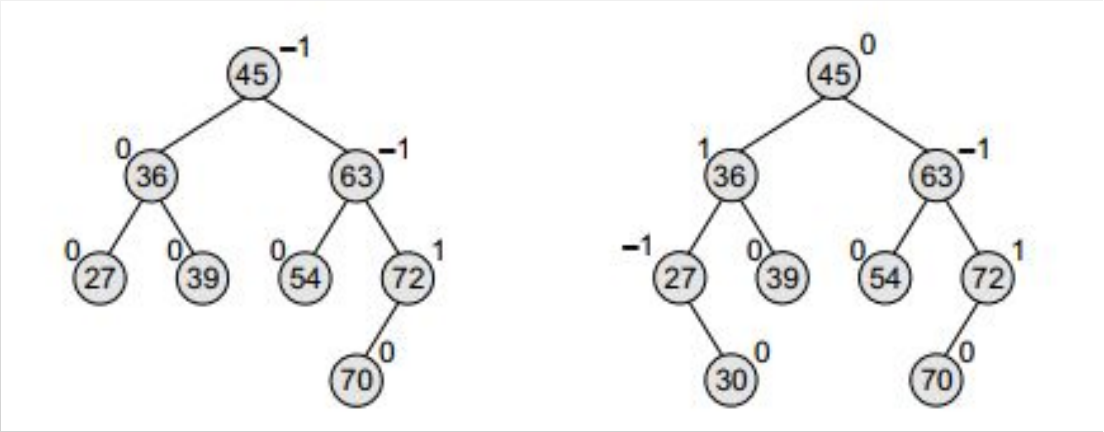
- Searching for a Node in an AVL Tree
  - Exactly the same way as it is performed in a binary search tree.
  - The search operation takes O(log n)
- Inserting a New Node in an AVL Tree
  - Done in the same way as it is done in a binary search tree.
  - The new node is always inserted as the leaf node.
  - Rotation is done to restore the balance of the tree.
  - If balance factor is in between -1 and 1 than no rotation is required
- Deleting a Node from an AVL Tree
  - Is similar to that of binary search trees.
  - Deletion may disturb the balance of the tree
  - Rotation is done to rebalance the tree if tree loses the AVLness

Insertion in AvL Trees

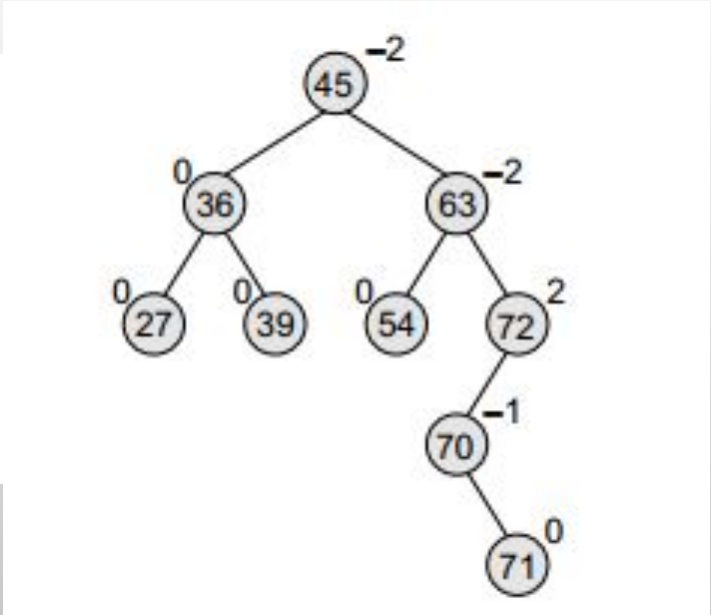
- The balance factor of the newly inserted node is always zero.
- But nodes on path from root to newly inserted node may get unbalanced by balance factor.
- 3 cases in insertion:
  - Initially, the node was either left- or right-heavy and after insertion, it becomes balanced.
  - Initially, the node was balanced and after insertion, it becomes either left- or right-heavy.
  - Initially, the node was heavy (either left or right) and the new node has been inserted in the heavy sub-tree, thereby creating an unbalanced sub-tree.

Critical Node: The unbalanced node after insertion.

Insertion in AvL Trees



Insertion of 30 with no balancing requirement.



Insertion of 71 with balancing requirement.

Insertion in AvL Trees

The four categories of rotations are:

- **LL rotation** The new node is inserted in the left sub-tree of the left sub-tree of the critical node.
- **RR rotation** The new node is inserted in the right sub-tree of the right sub-tree of the critical node.
- **LR rotation** The new node is inserted in the right sub-tree of the left sub-tree of the critical node.
- **RL rotation** The new node is inserted in the left sub-tree of the right sub-tree of the critical node

Insertion in AvL Trees (LL Rotation)

While rotation, node B becomes the root, with T1 and A as its left and right child. T2 and T3 become the left and right sub-trees of A.

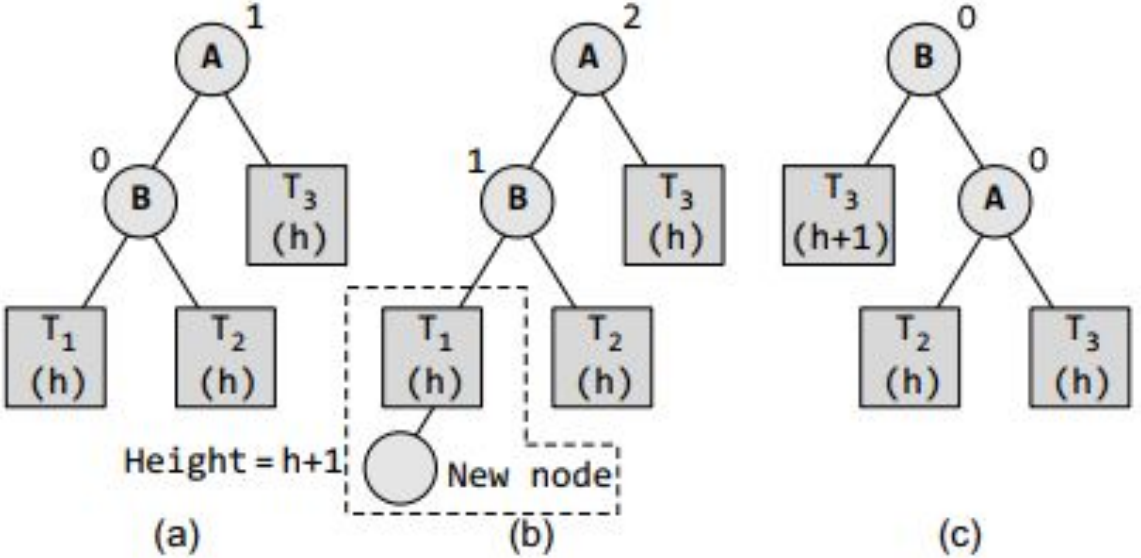


Figure 10.40 LL rotation in an AVL tree

Insertion in AvL Trees (LL Rotation)

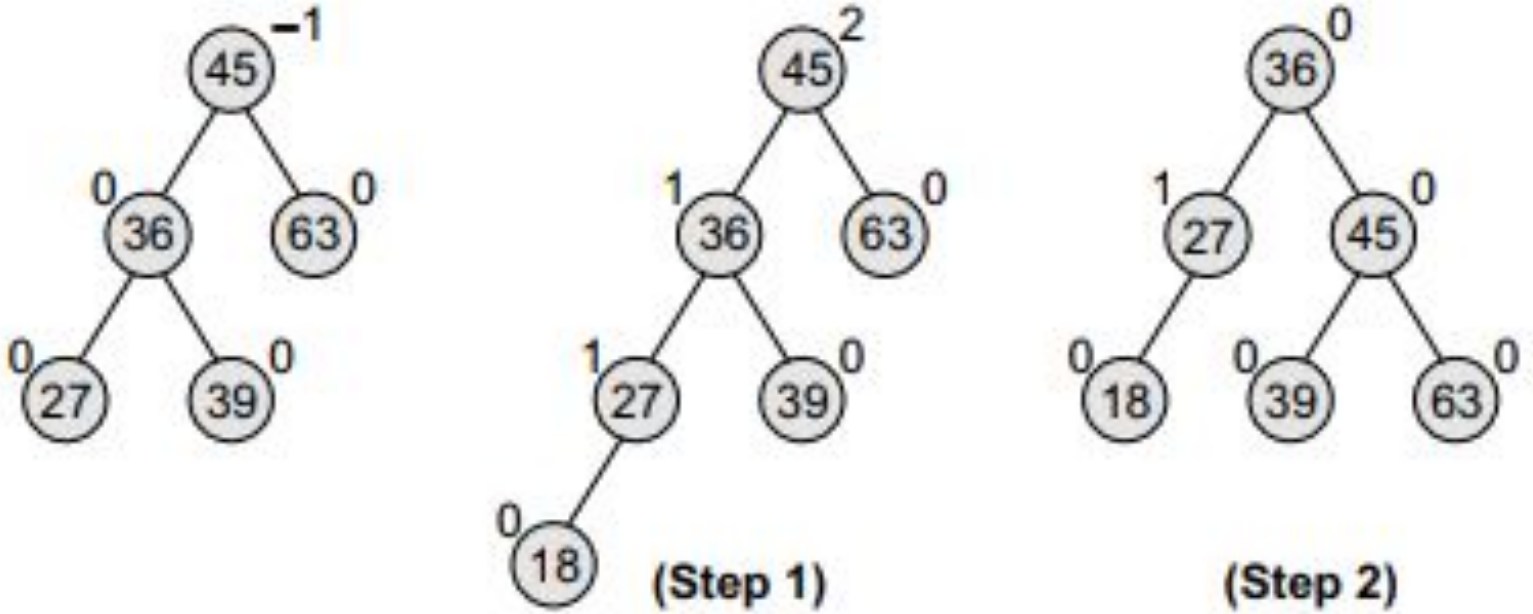


Figure 10.41 AVL tree

Insertion in AvL Trees (RR Rotation)

While rotation, node B becomes the root, with A and T 3 as its left and right child. T1 and T2 become the left and right sub-trees of A.

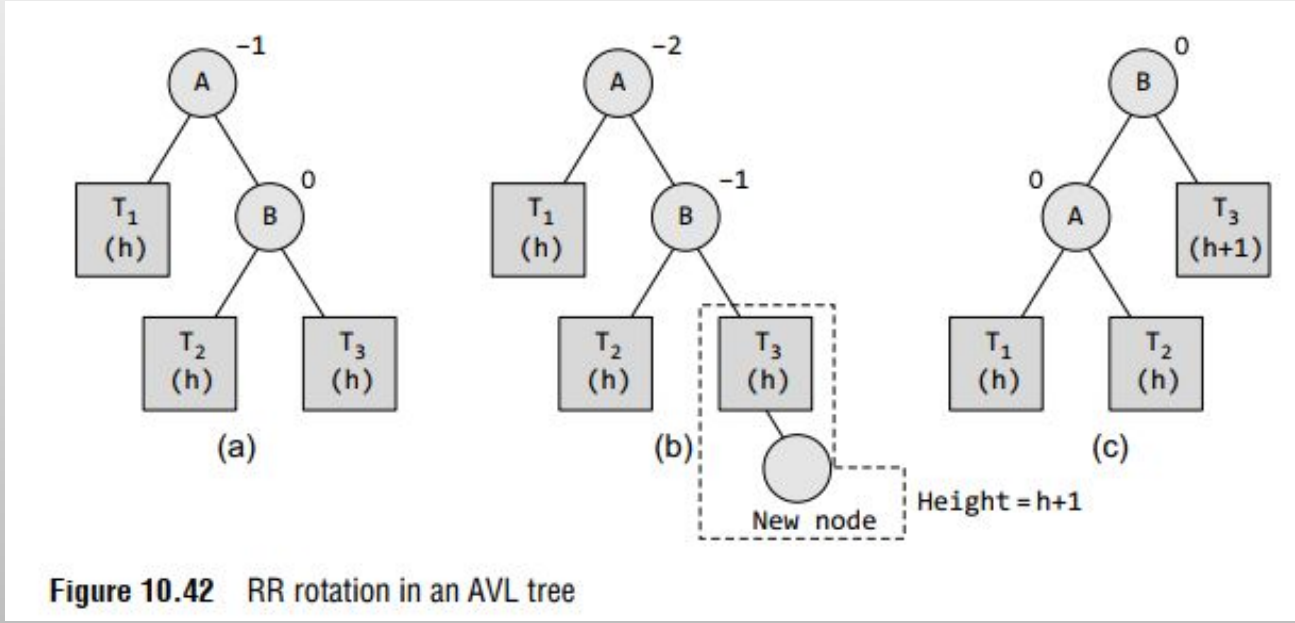
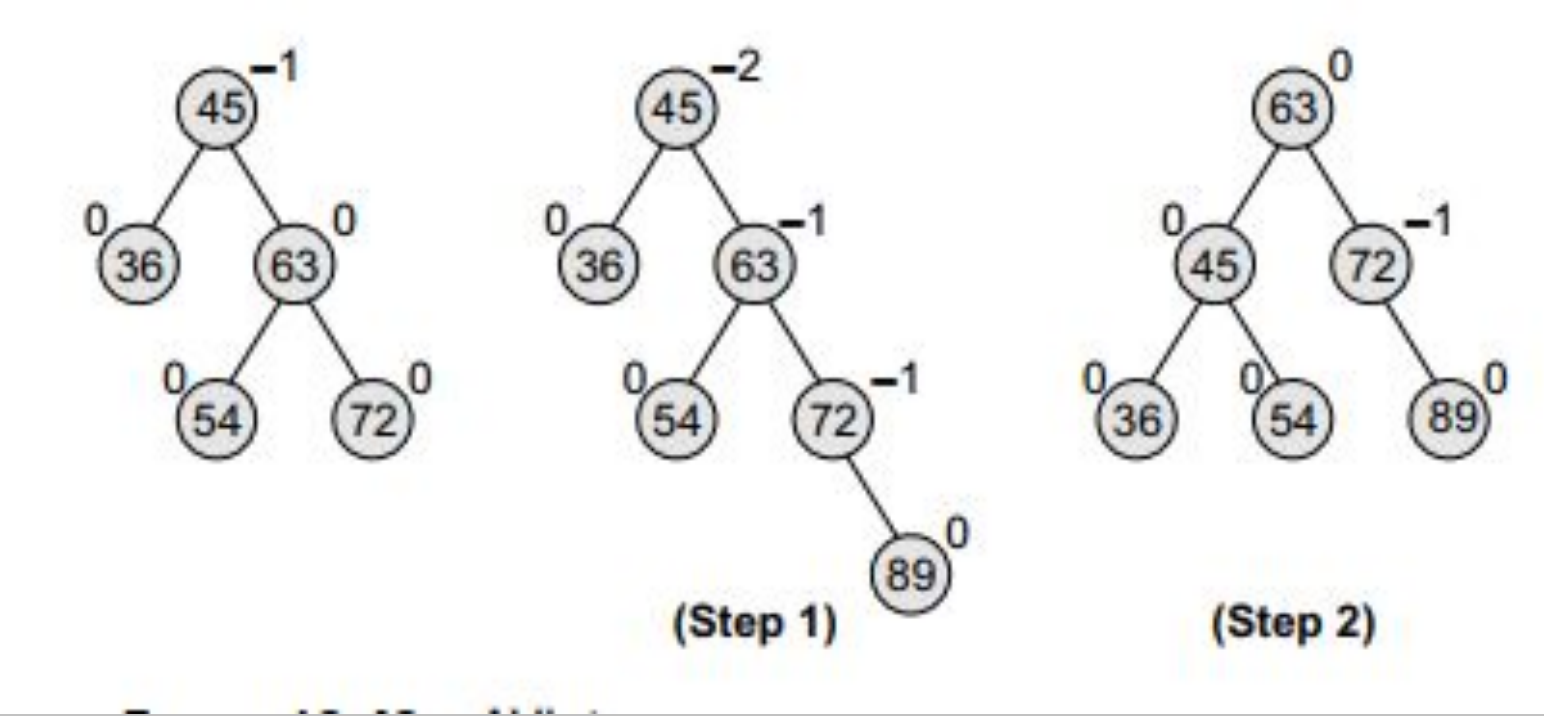


Figure 10.42 RR rotation in an AVL tree

Insertion in AvL Trees (RR Rotation)



Insertion in AvL Trees (LR Rotation)

While rotation, node C becomes the root, with B and A as its left and right children. Node B has T1 and T2 as its left and right sub-trees and T3 and T 4 become the left and right sub-trees of node A

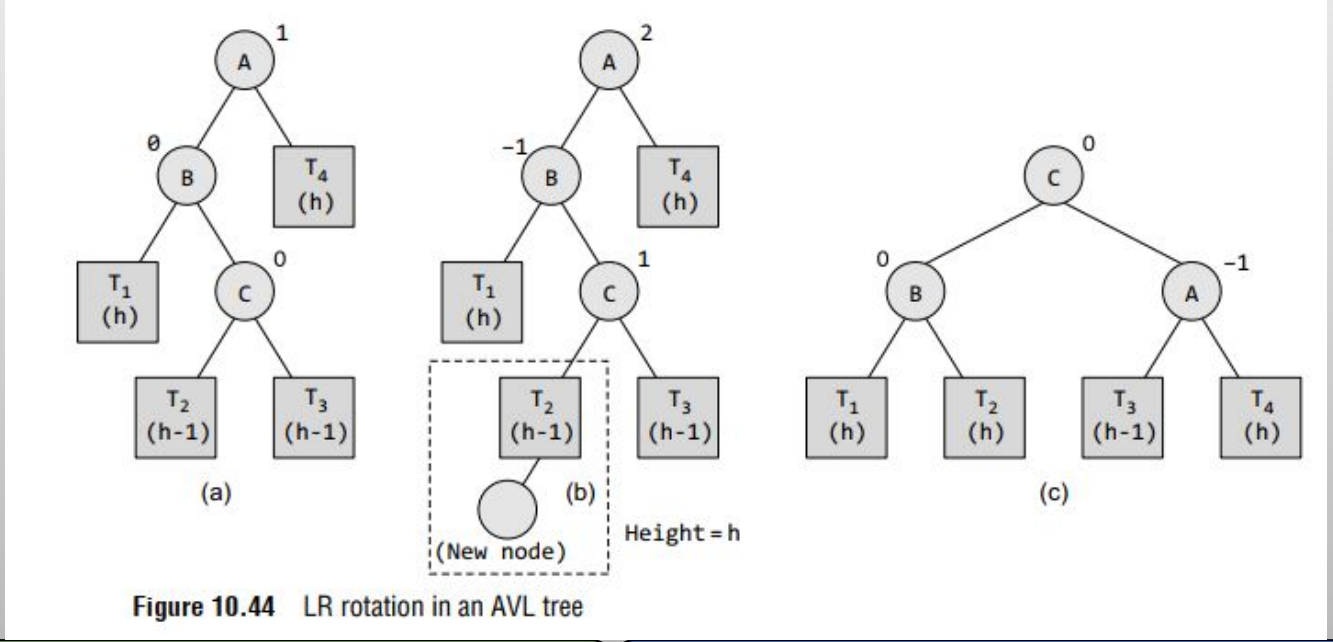
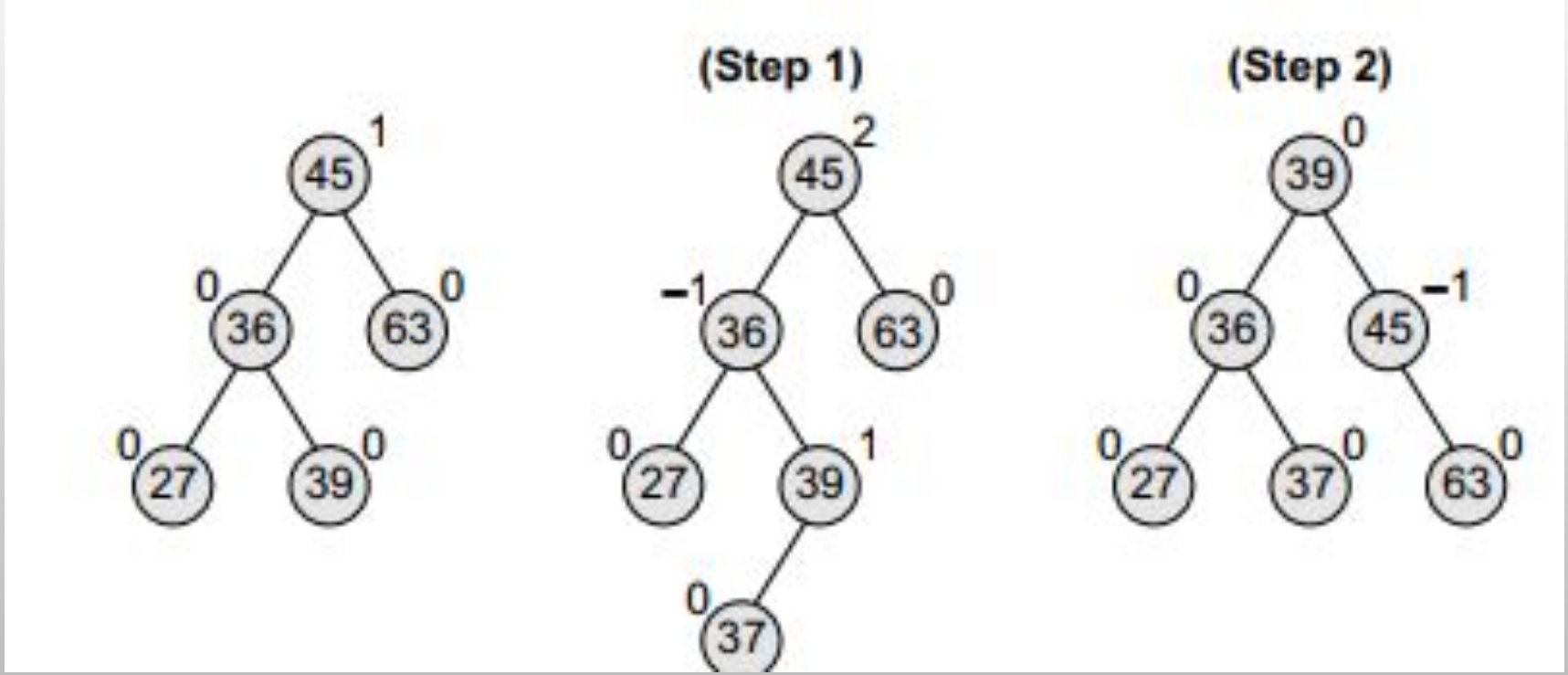


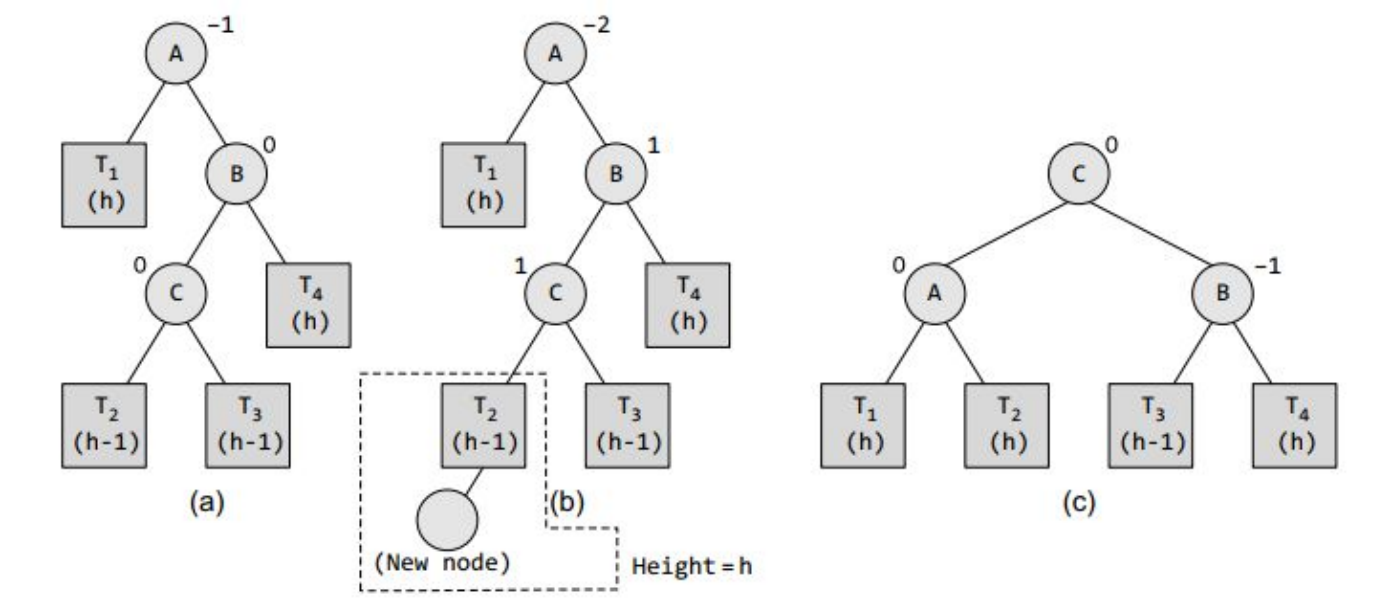
Figure 10.44 LR rotation in an AVL tree

Insertion in AvL Trees (LR Rotation)



Insertion in AvL Trees (RL Rotation)

While rotation, node C becomes the root, with A and B as its left and right children. Node A has T1 and T2 as its left and right sub-trees and T3 and T4 become the left and right sub-trees of node B.



Example 10.6 Construct an AVL tree by inserting the following elements in the given order. 63, 9, 19, 27, 18, 108, 99, 81.

Solution

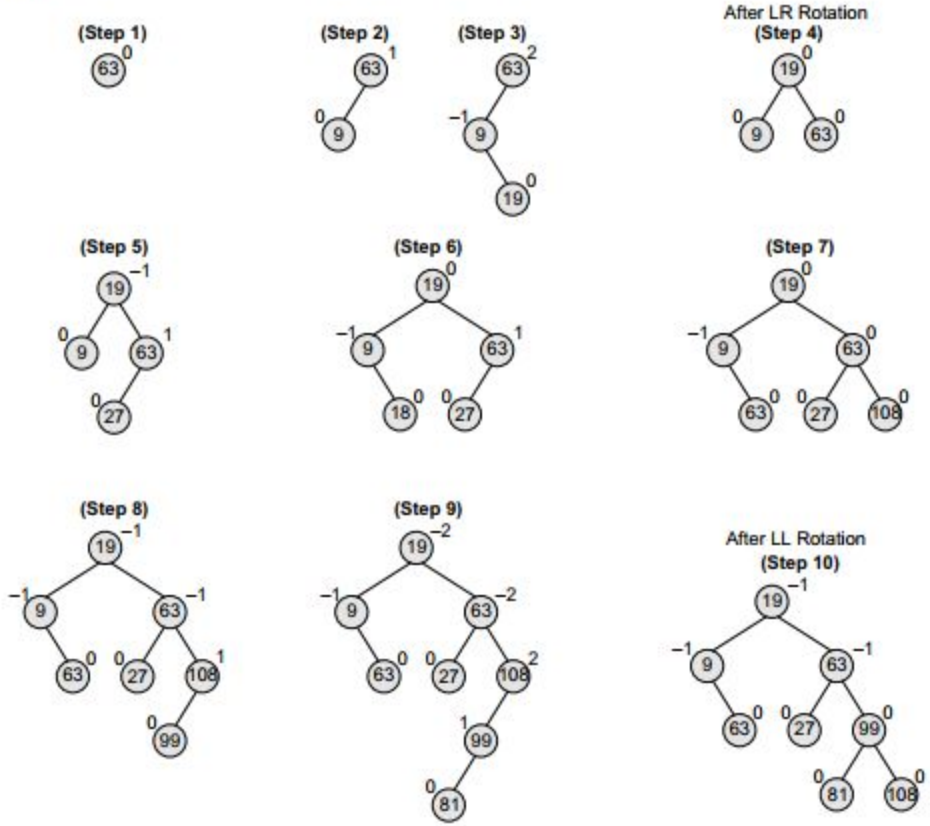
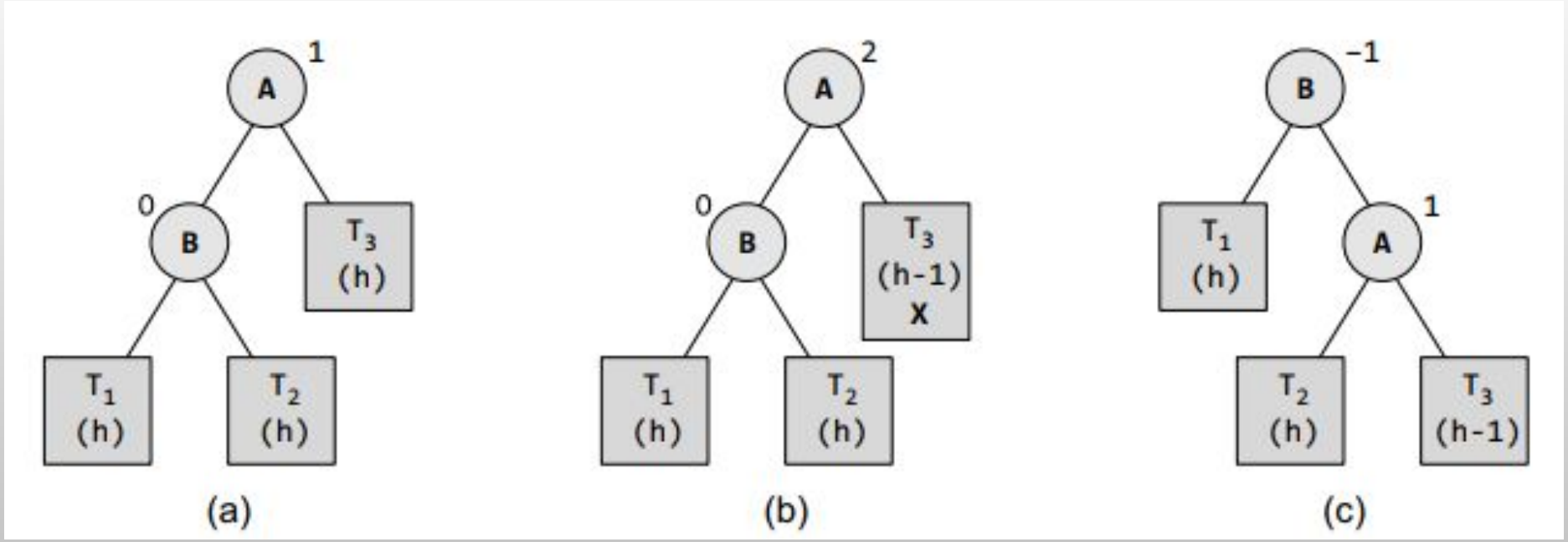
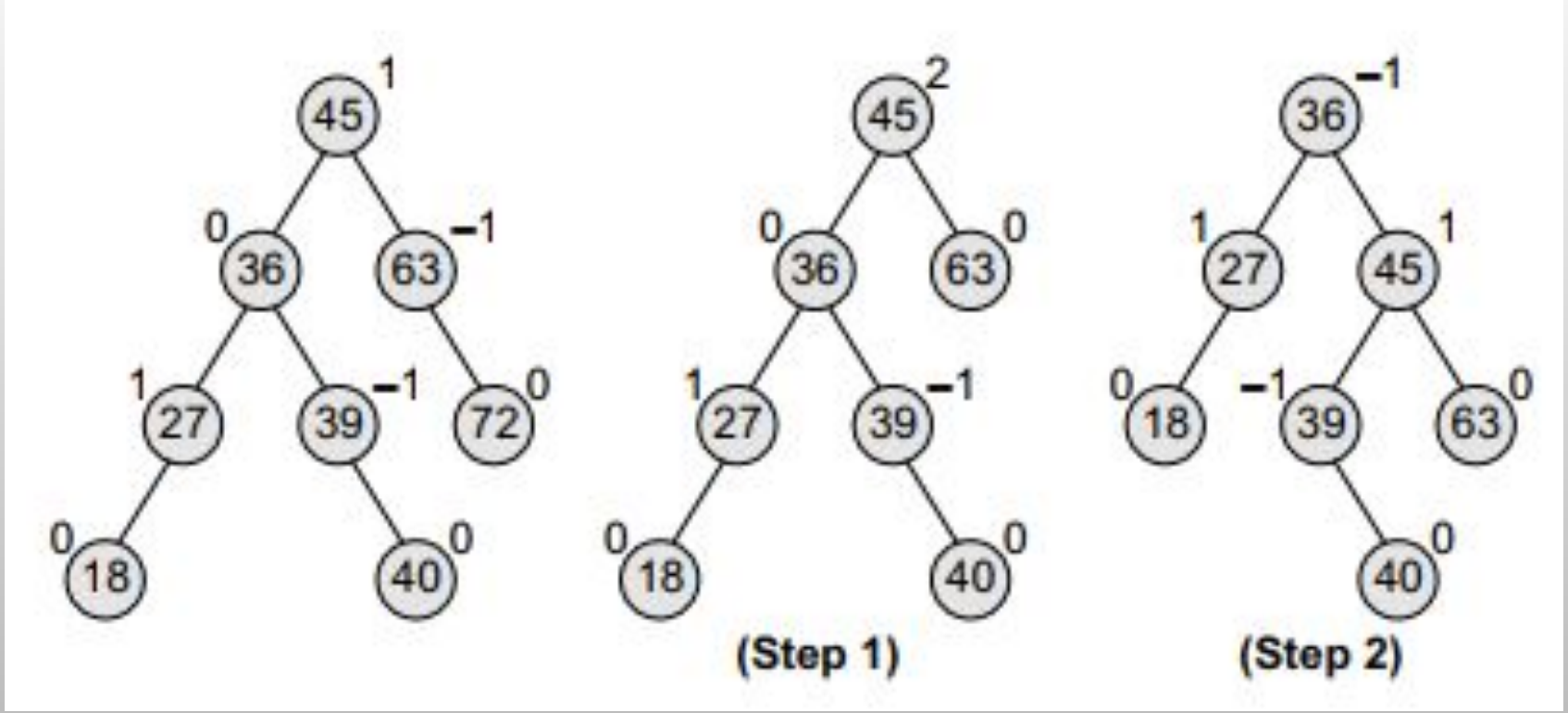


Figure 10.47 AVL tree

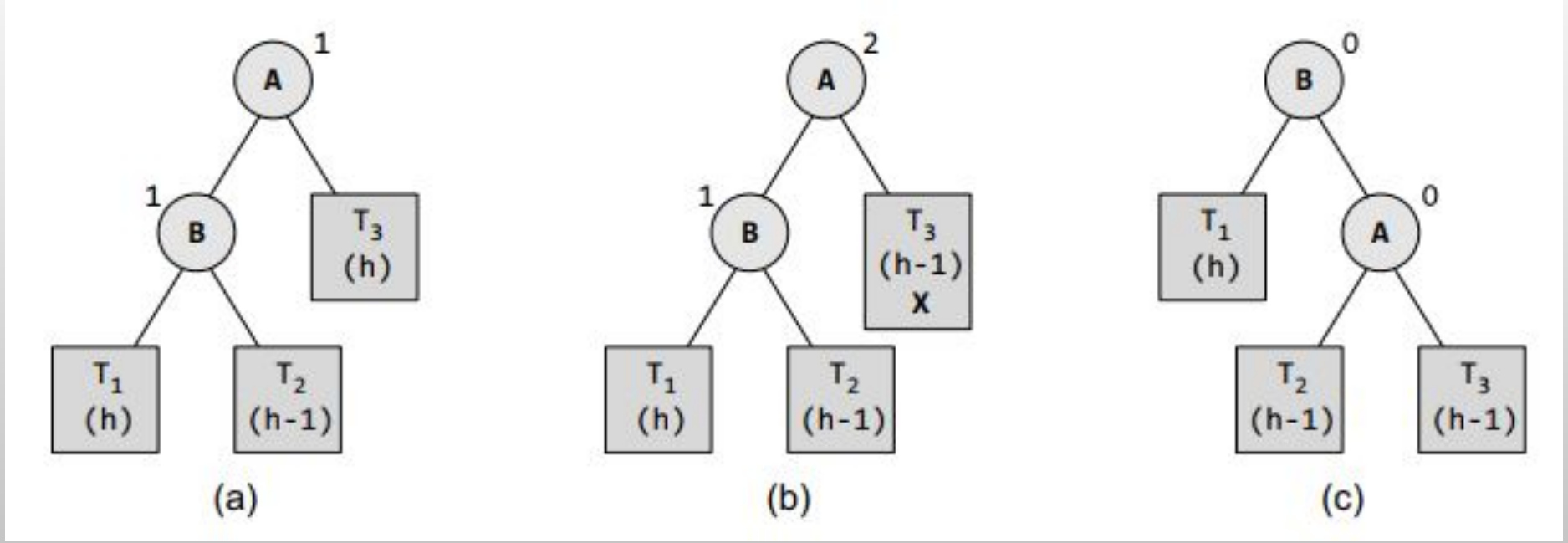
Deletion in AvL Trees (R0 Rotation)



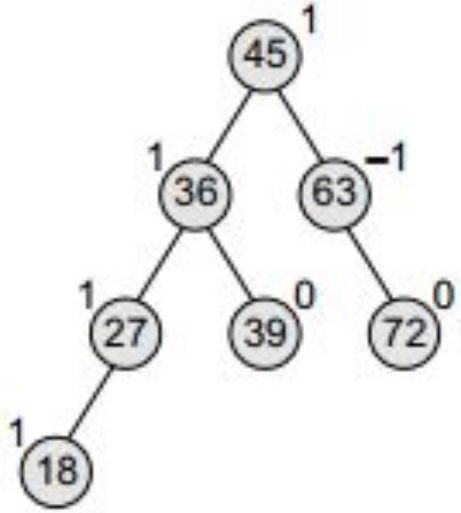
Deletion in AvL Trees (R0 Rotation)

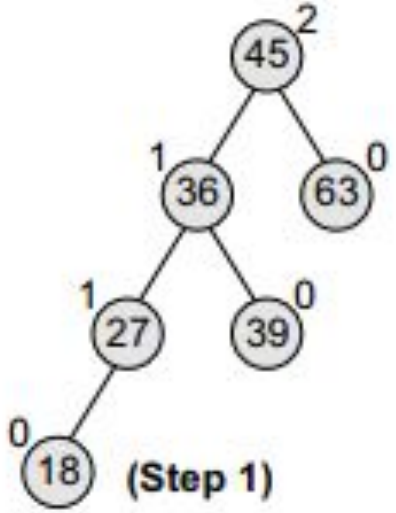


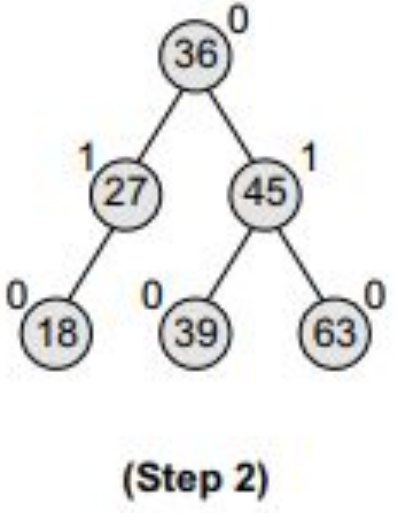
Deletion in AvL Trees (R1 Rotation)



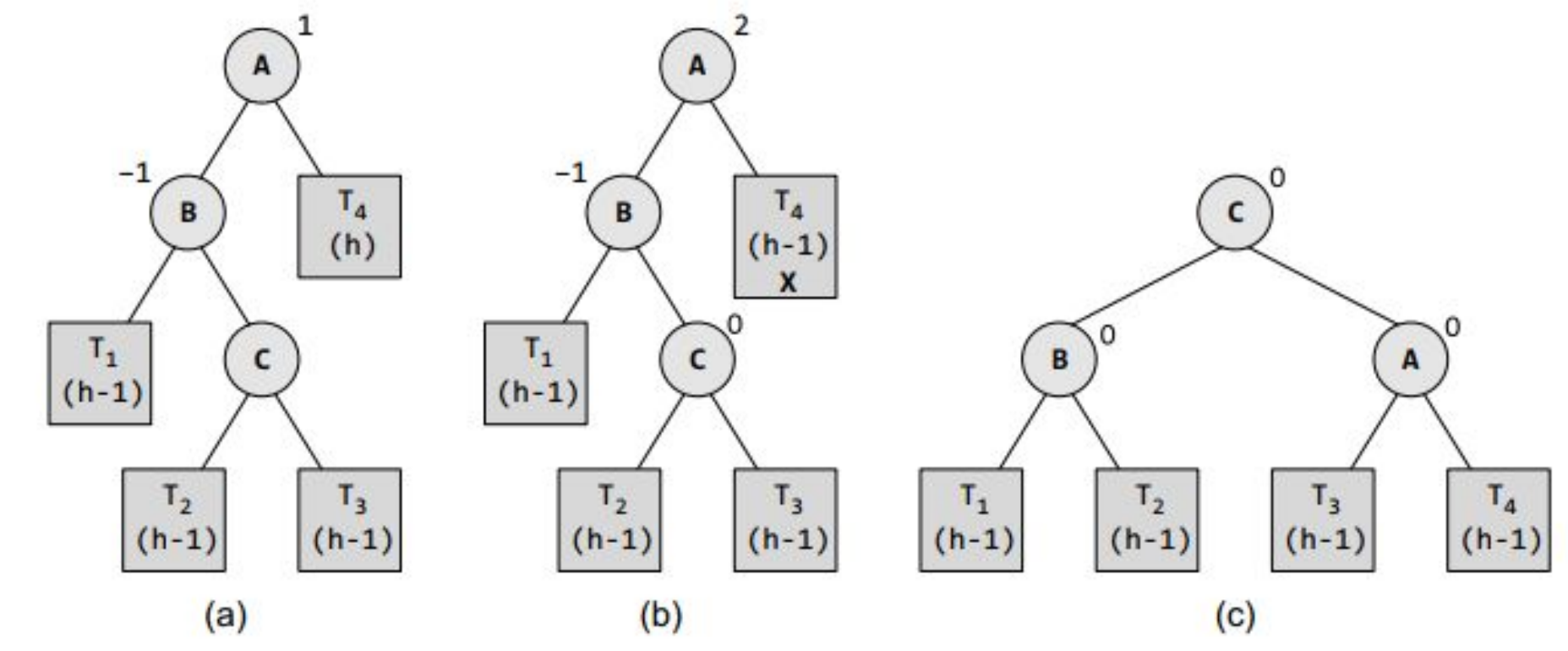
Deletion in AvL Trees (R1 Rotation)



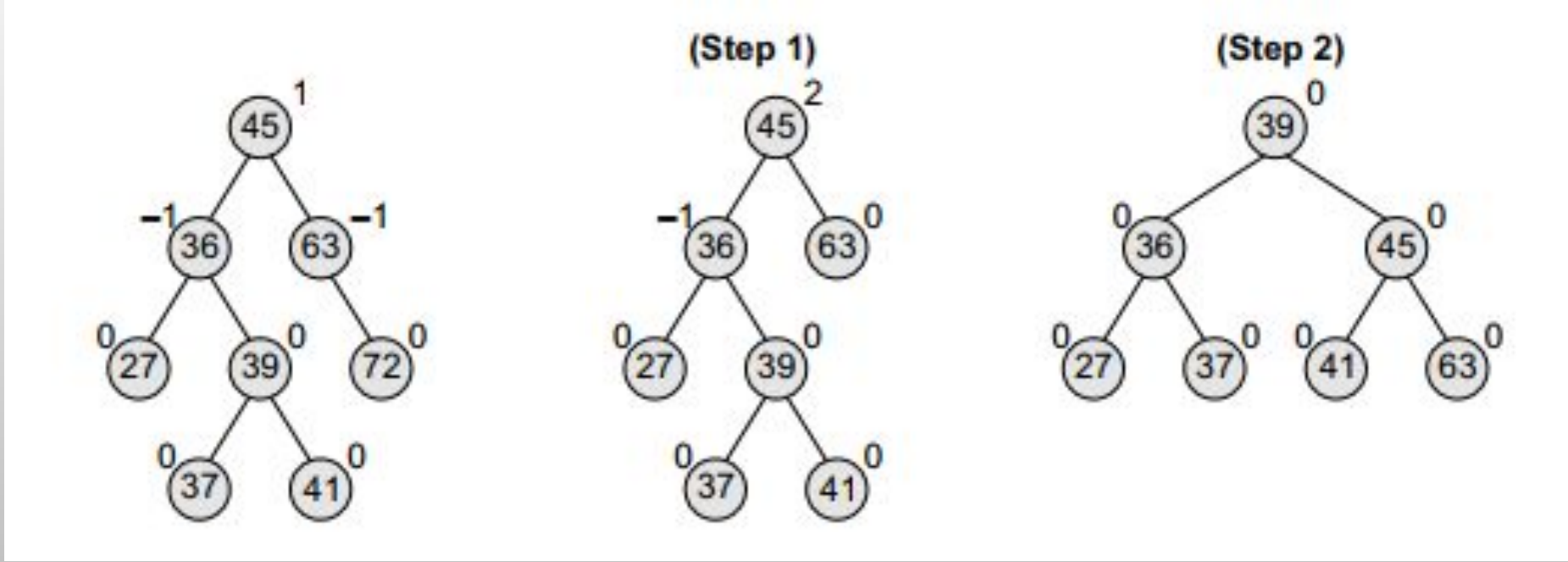




Deletion in AvL Trees (R-1 Rotation)

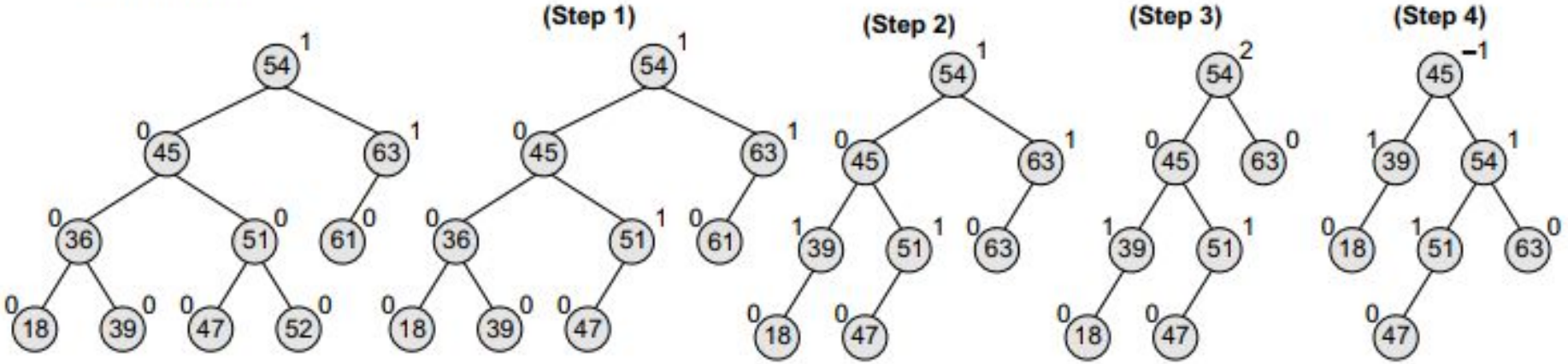


Deletion in AvL Trees (R-1 Rotation)



Example 10.10 Delete nodes 52, 36, and 61 from the AVL tree given in Fig. 10.54.

Solution



Acknowledgements

Data Structures
Using
c
Reema Thareja