Chapter 10
Search Trees

Section 10.2
AVL Trees

- Binary Search Trees can become very unbalanced, depending upon the order in which keys are inserted.
- In the worst case, when insertions are made in sorted order, the tree is no better than a linear list.

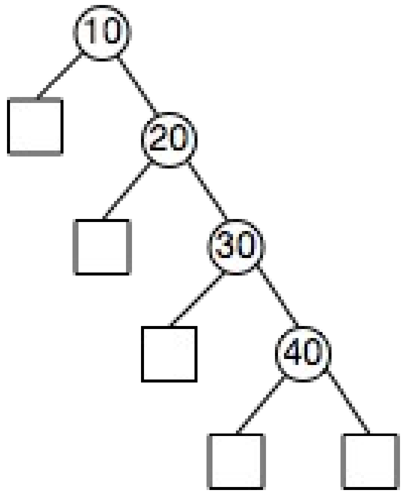


Figure 10.7. An unbalanced binary search tree with linear height.

- The tree can be rebalanced as nodes are inserted or removed.
- One such tree is the AVL tree, named after its inventors, Adel'son-Vel'skii and Landis
- The AVL tree maintains height balance, where the heights of any node's left and right subtrees differ by no more than 1.

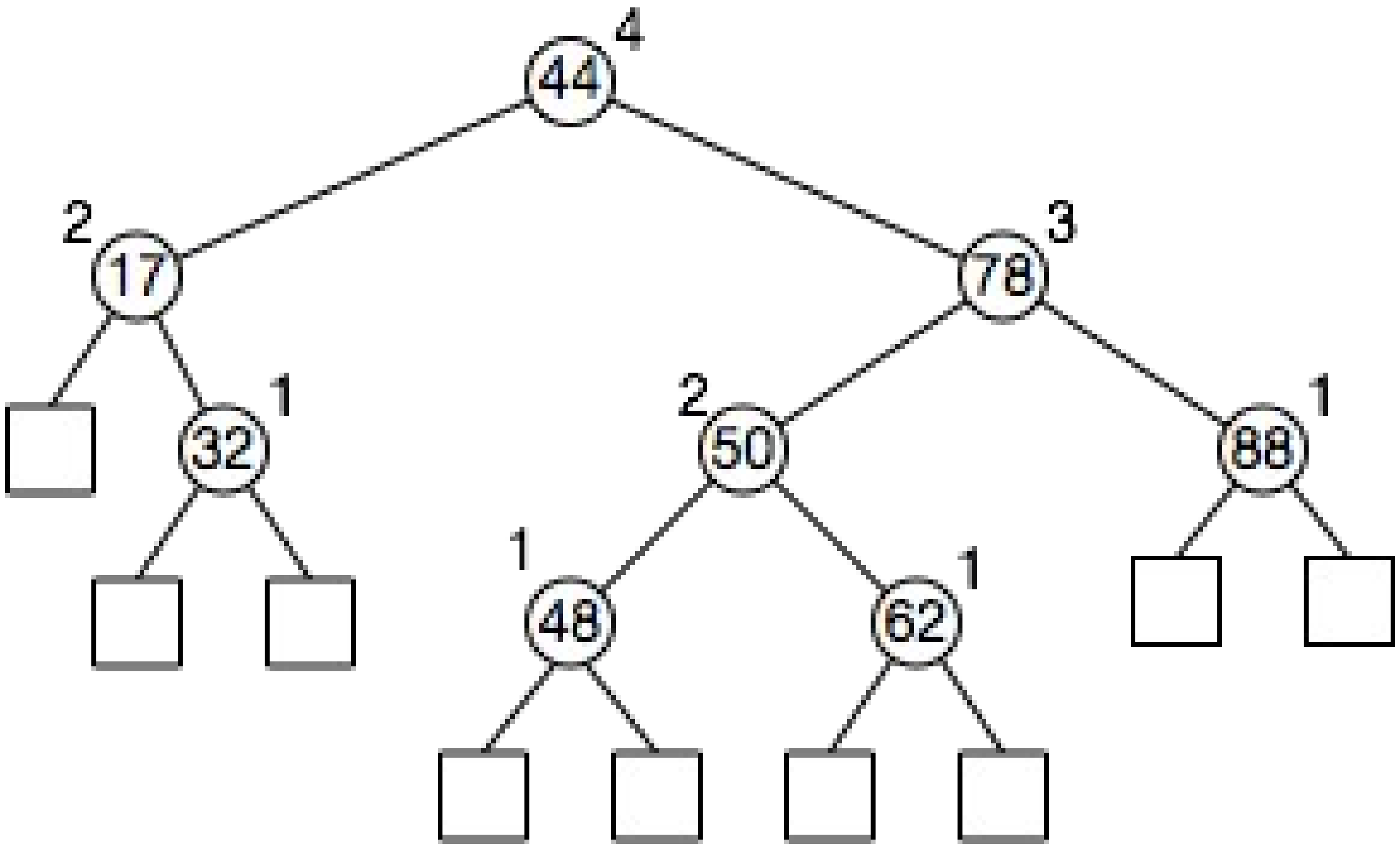
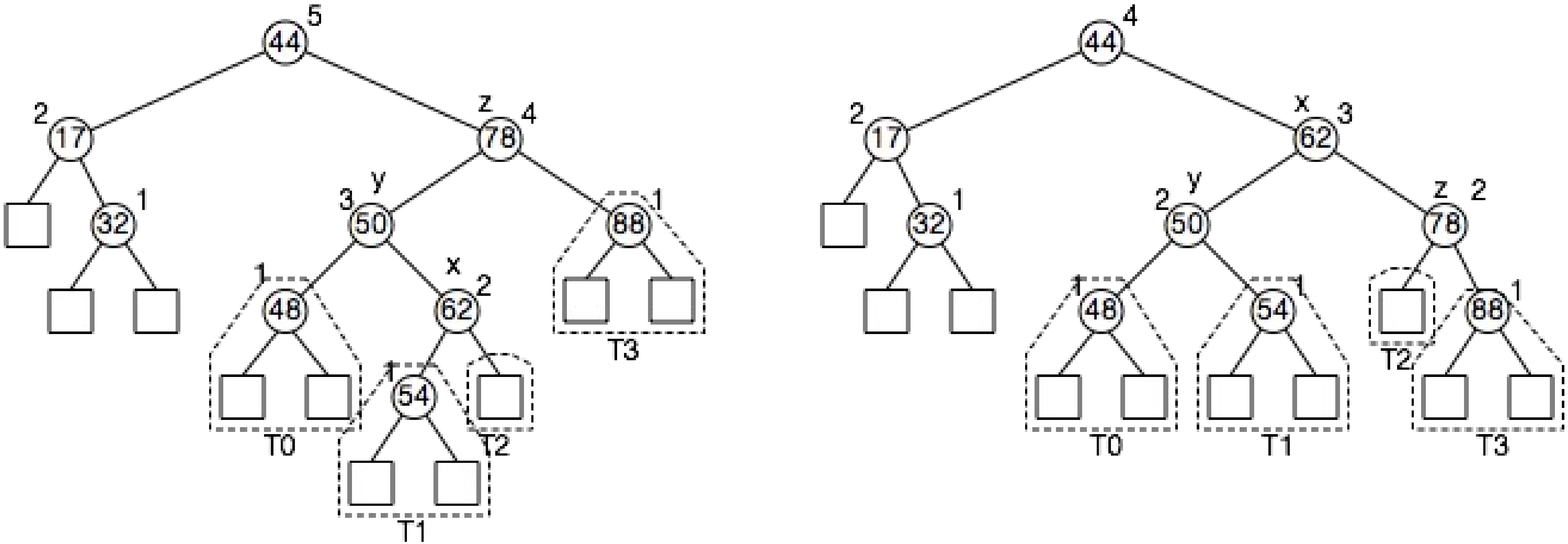


Figure 10.8. An AVL tree with node heights indicated.



(a) Unbalanced after 54 inserted.

(b) Rebalanced.

Figure 10.9. Insertion into an AVL tree.

Trinode Restructuring

- a, b, c is the inorder listing of nodes x, y, z (child, parent, grandparent)
- T0, T1, T2, T3 are the left-to-right subtrees of x, y, and z

1. Replace the subtree rooted at **z** with a subtree rooted at **b**
2. Make **a** the left child of **b**
3. Make T0 and T1 the left and right children of **a**
4. Make **c** the right child of **b**
5. Make T2 and T3 the left and right children of **c**

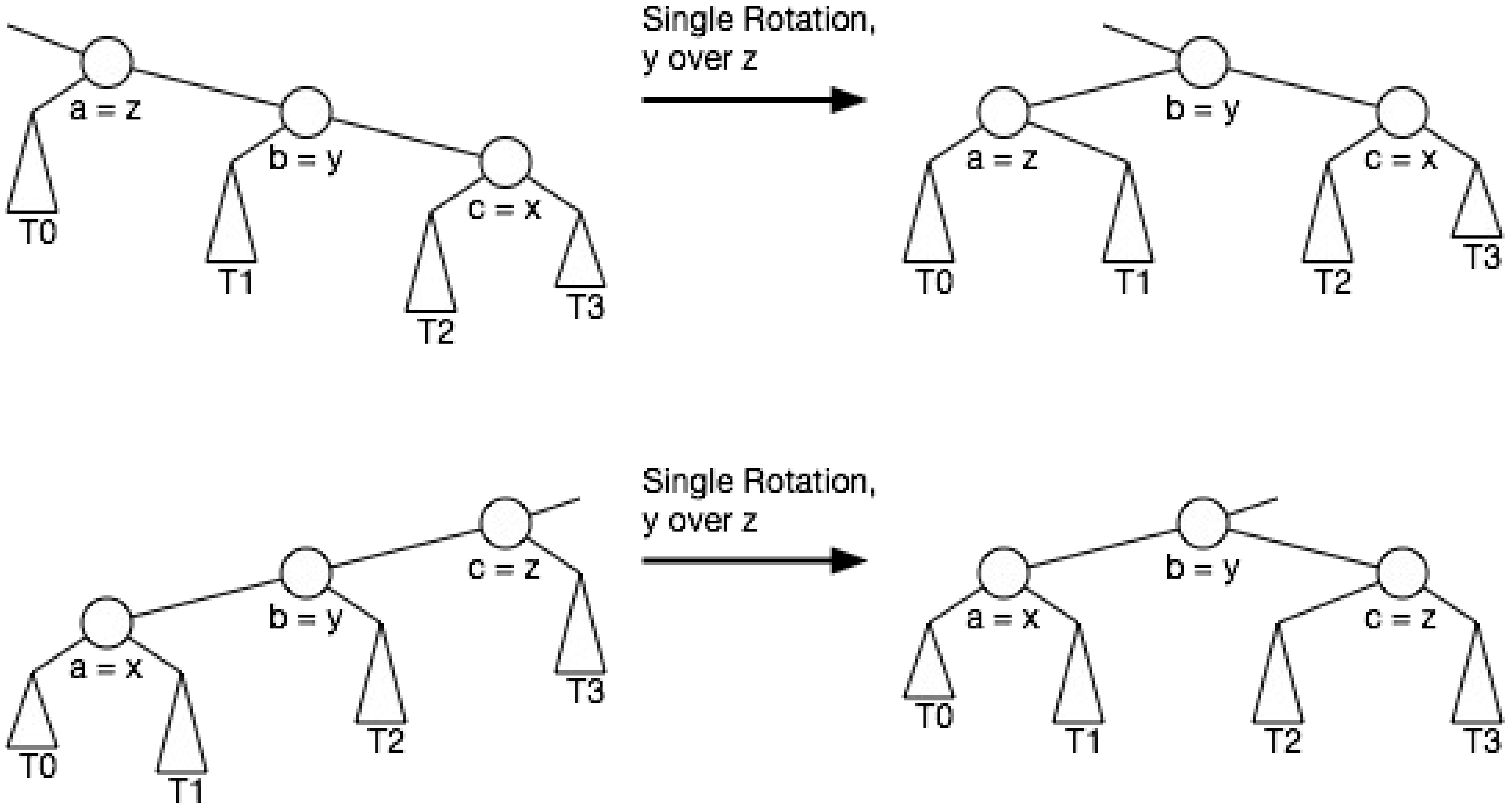


Figure 10.10 a & b. Trinode restructuring.

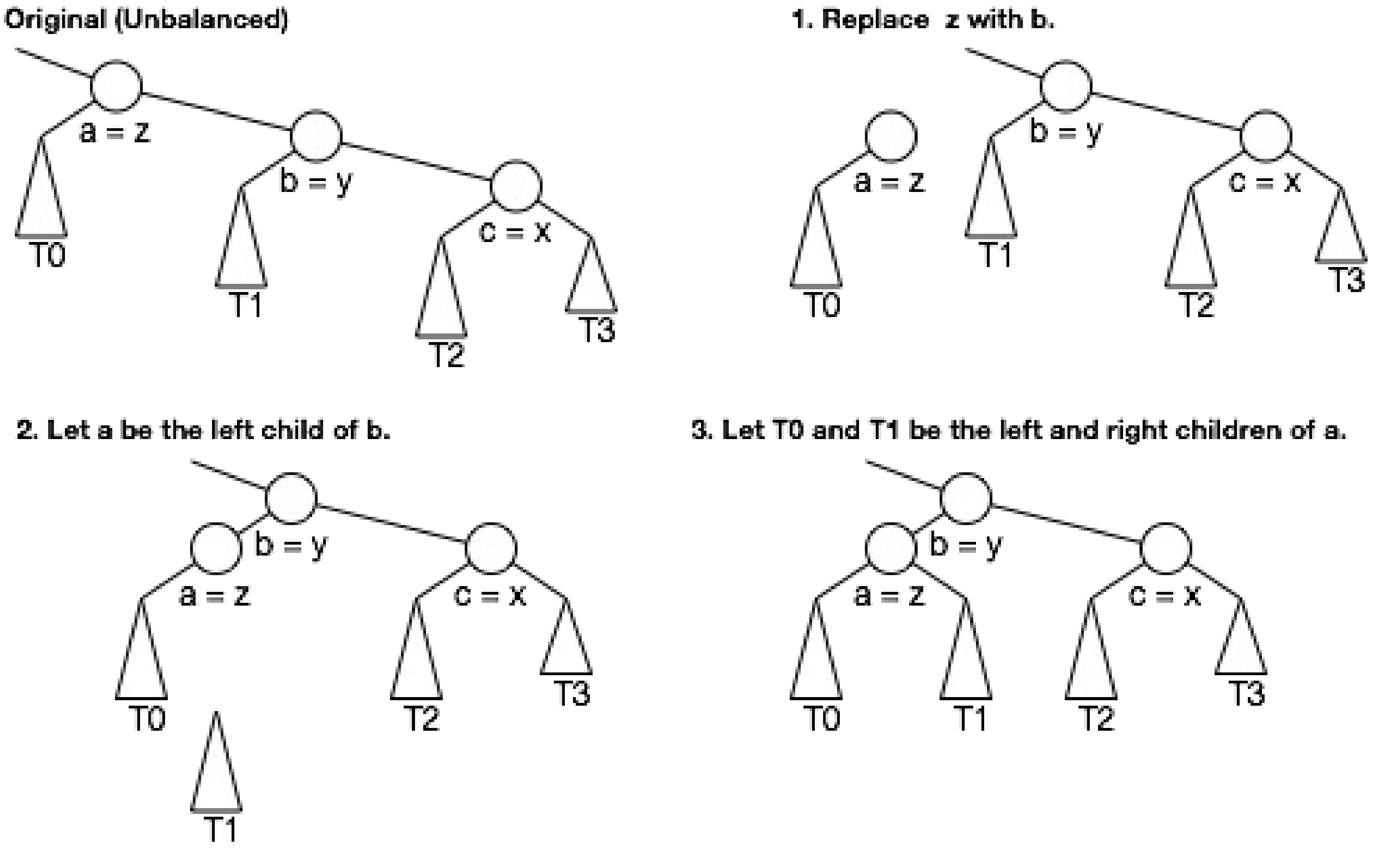


Figure 10.10a. Steps 1-3.

4. Let c be the right child of b—already true.

5. Let T2 and T3 be the left and right children of c—already true.

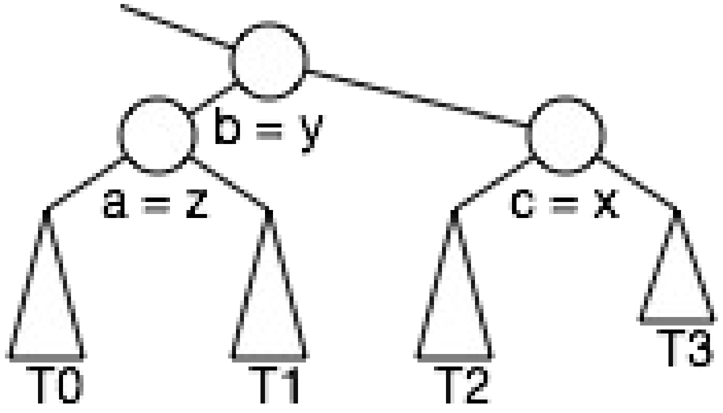


Figure 10.10a. Steps 4-5.

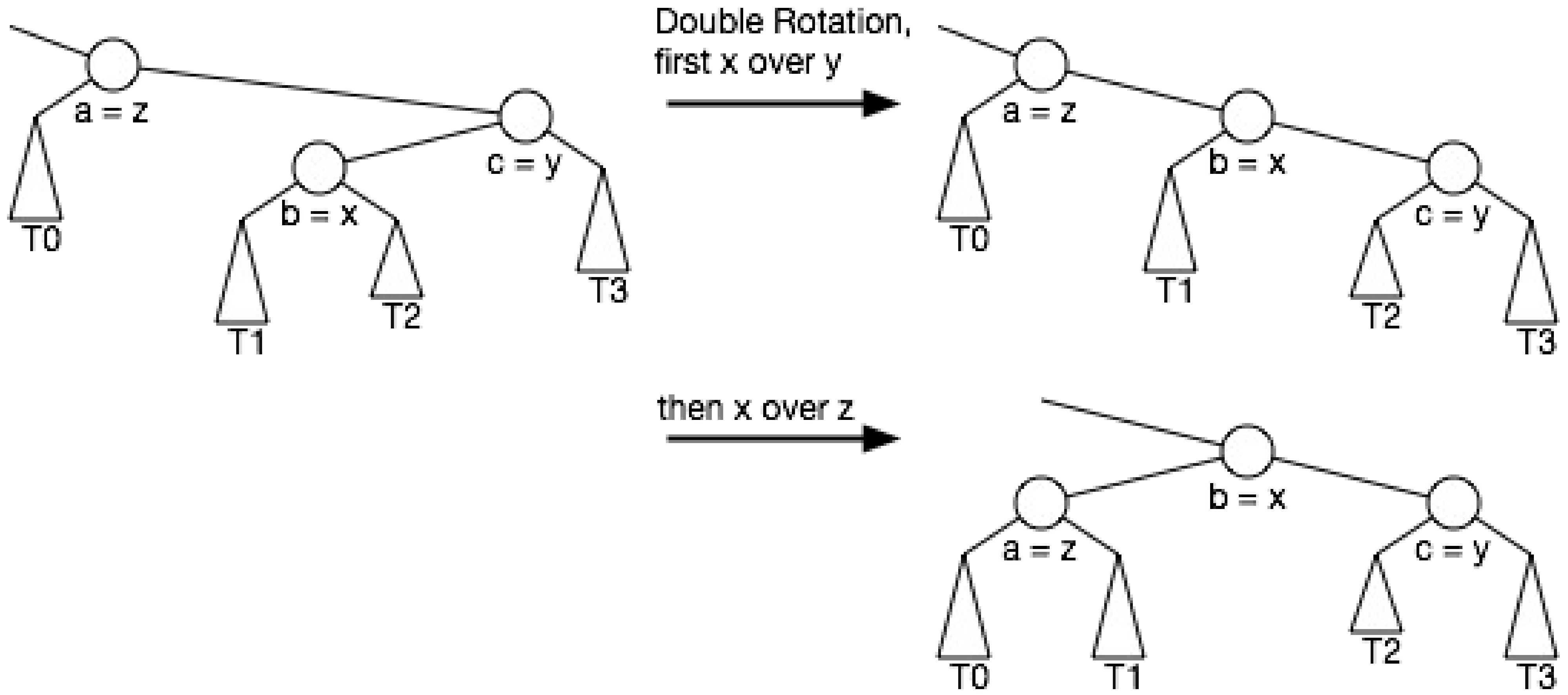


Figure 10.10c. Trinode restructuring.

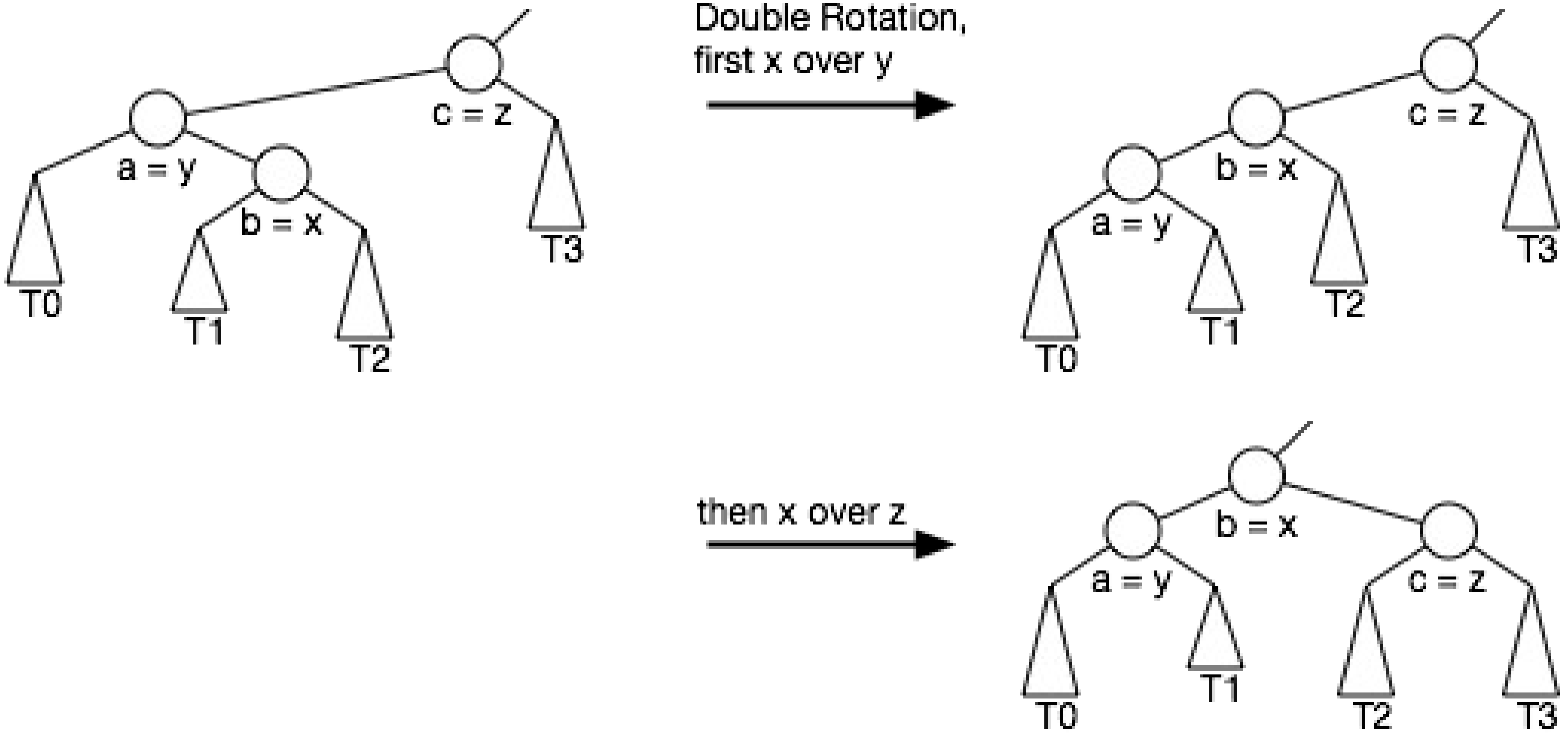
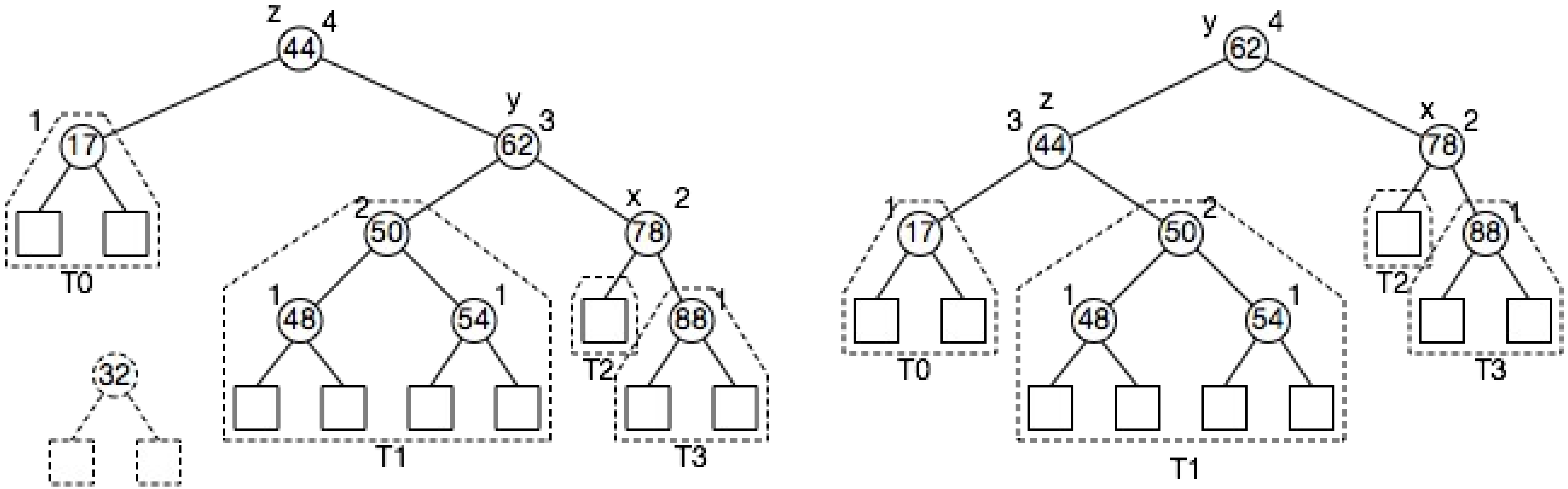


Figure 10.10d. Trinode restructuring.



(a) Unbalanced after 32 removed.

(b) Rebalanced with a single rotation.

Figure 10.11. Removal from an AVL tree.