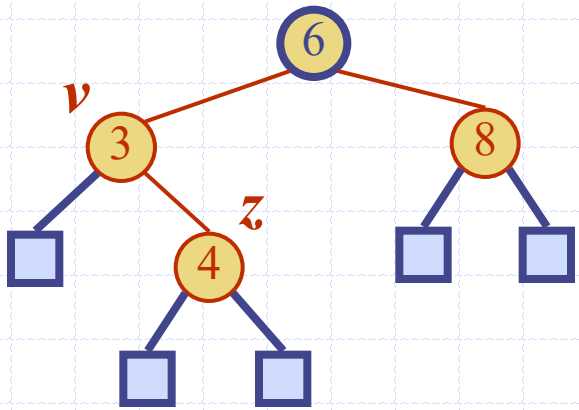
Presentation for use with the textbook Data Structures and Algorithms in Java, 6th edition, by M. T. Goodrich, R. Tamassia, and M. H. Goldwasser, Wiley, 2014

Balanced Search Trees



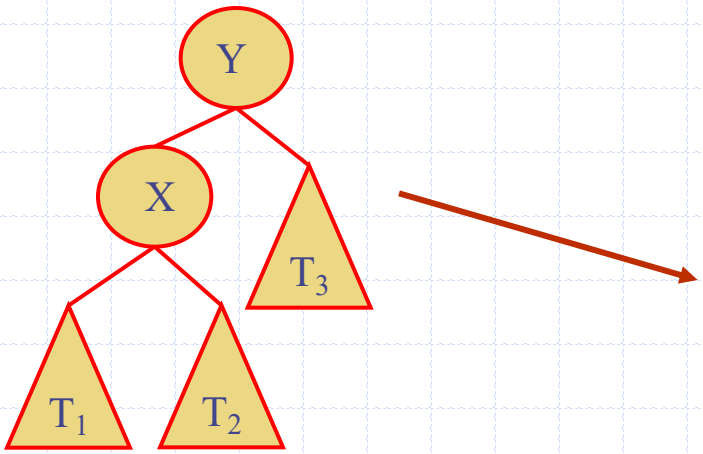
Reading

M. T. Goodrich, R. Tamassia and M. H. Goldwasser, *Data Structures and Algorithms in Java*, 6th Edition, 2014.

- **Chapter 11. Search Tree Structures**
- **Sections 11.1-11.2**
- **pp. 423-442**

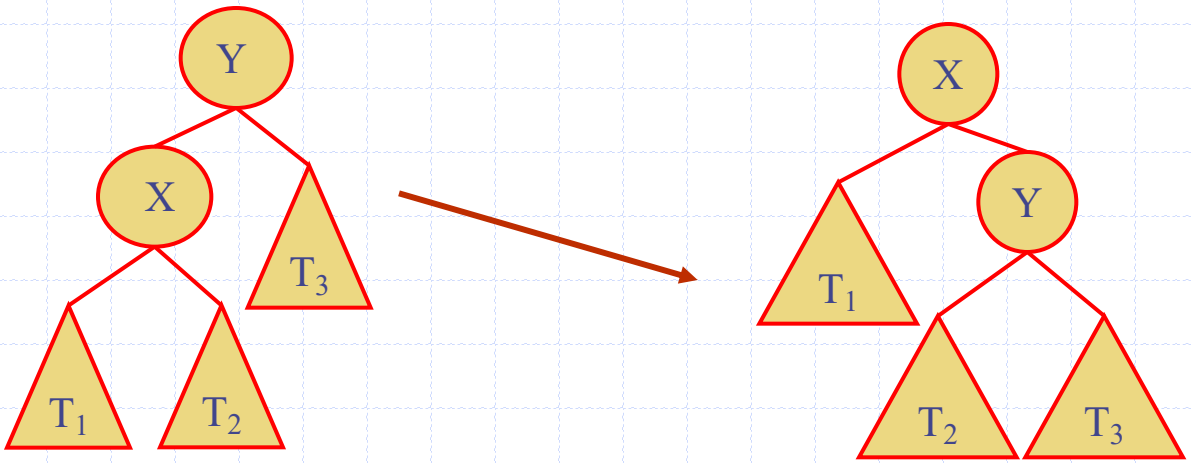
Rotation

- The primary operation to rebalance a binary search tree is known as a ***rotation***.
- During a rotation, we "rotate" a child to be above its parent.



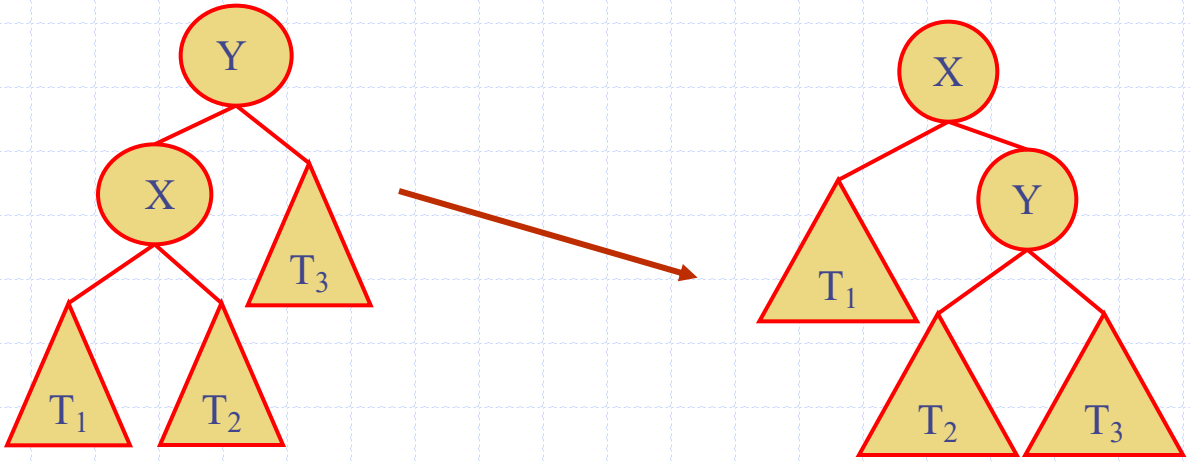
Rotation

- The primary operation to rebalance a binary search tree is known as a ***rotation***.
- During a rotation, we "rotate" a child to be above its parent.

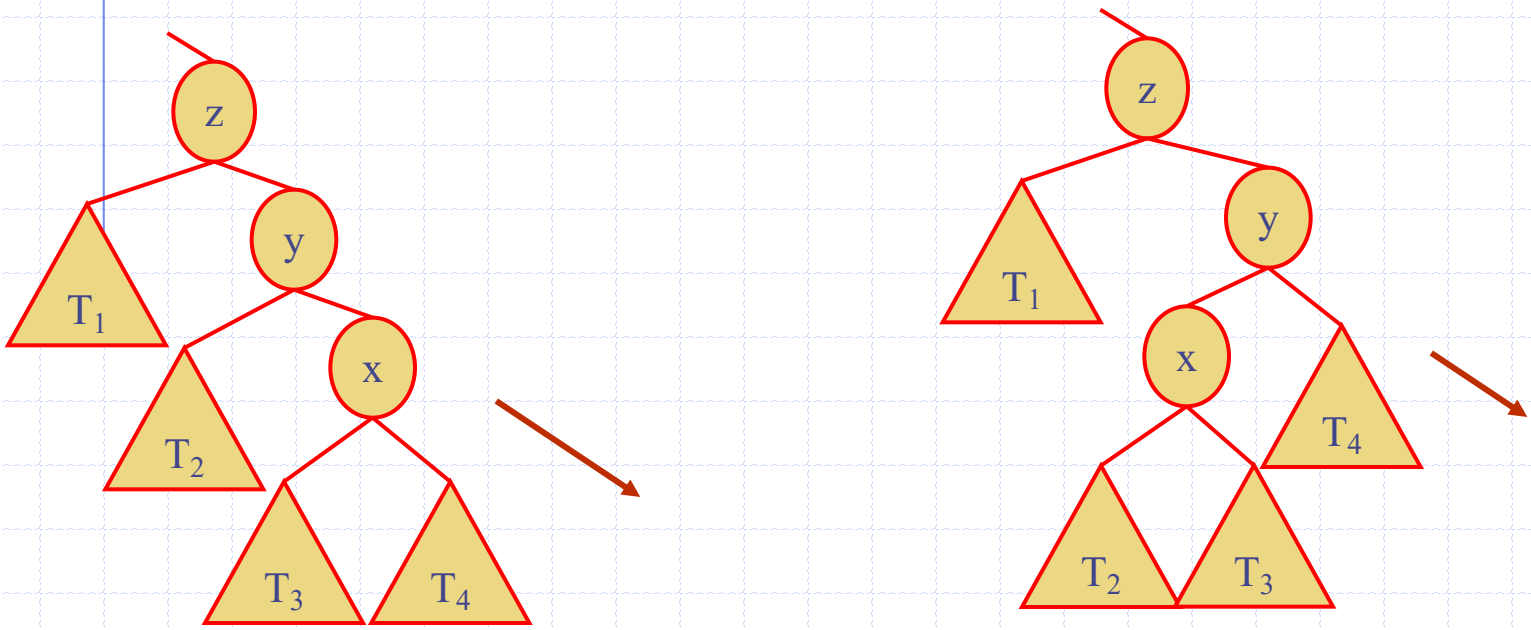


Rotation

- In the context of a tree-balancing algorithm, a rotation allows the ***shape*** of a tree to be modified while maintaining the search-tree property.
- This operation can be used to ***avoid highly unbalanced tree configurations***.
- Check how the depth of each node in subtree T1 and T3 was changed by the "rotation".



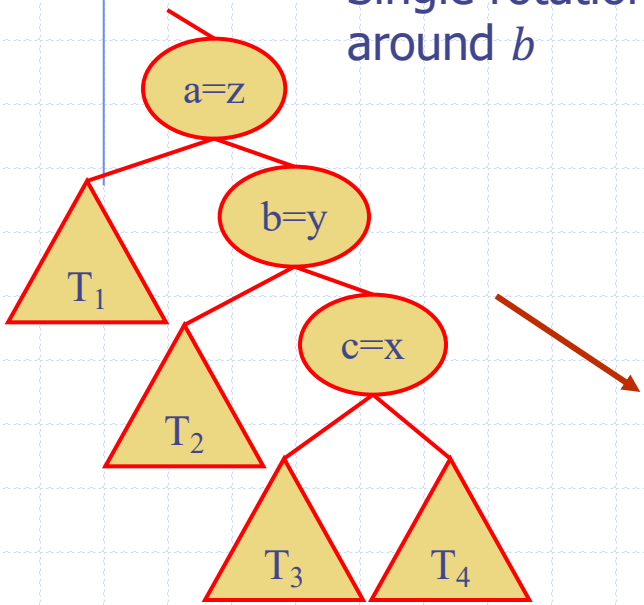
Trinode Restructuring



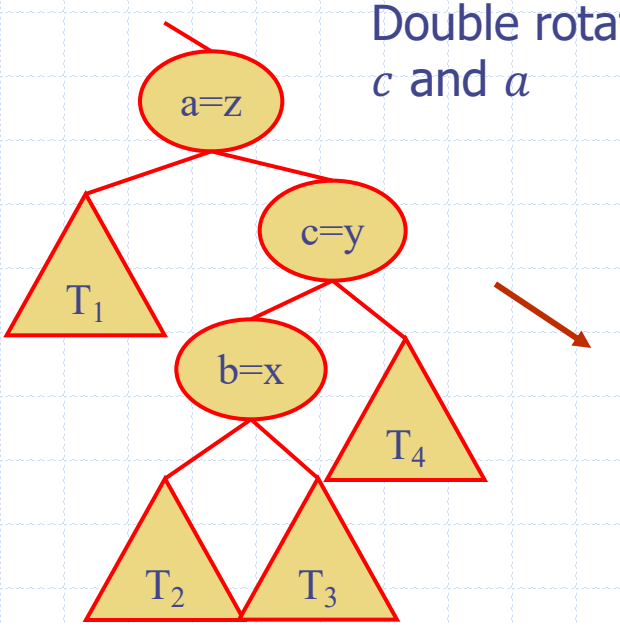
Trinode Restructuring

- Let (a, b, c) be the inorder listing of x, y, z
- Perform the rotations needed to make b the topmost node of the three

Single rotation around b

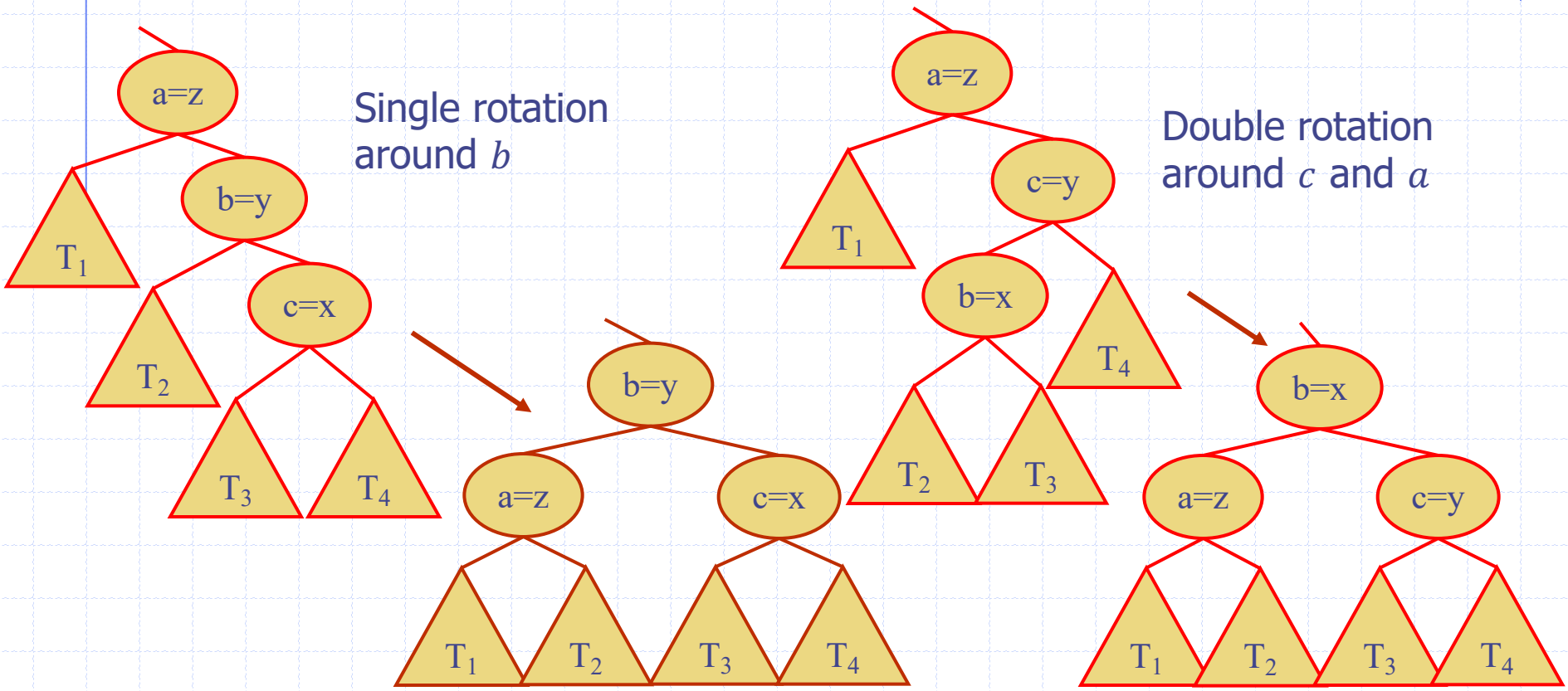


Double rotation around c and a



Trinode Restructuring

- Let (a, b, c) be the inorder listing of x, y, z
- Perform the rotations needed to make b the topmost node of the three



Restructure

Algorithm restructure(x):

Input: A position x of a binary search tree T that has both a parent y and a grandparent z

Output: Tree T after a trinode restructuring (which corresponds to a single or double rotation) involving positions x, y, and z

1: Let (a, b, c) be a left-to-right (inorder) listing of the positions x, y, and z, and let (T_1, T_2, T_3, T_4) be a left-to-right (inorder) listing of the four subtrees of x, y, and z not rooted at x, y, or z.

2: Replace the subtree rooted at z with a new subtree rooted at b.

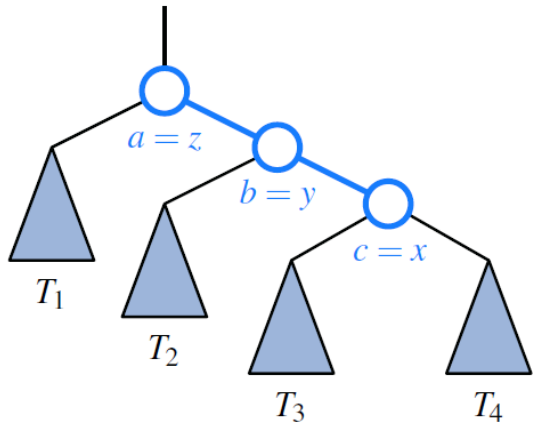
3: Let a be the left child of b and let T_1 and T_2 be the left and right subtrees of a, respectively.

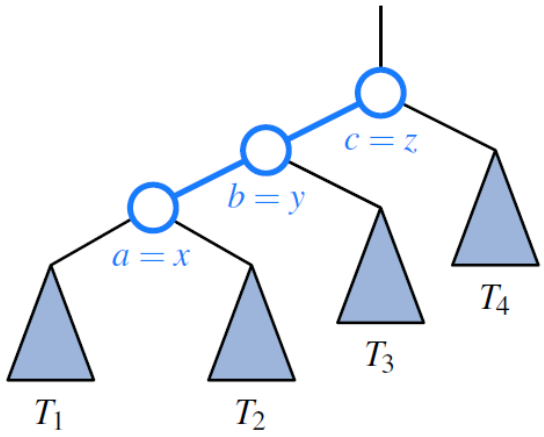
4: Let c be the right child of b and let T_3 and T_4 be the left and right subtrees of c, respectively.

Code Fragment 11.7: The trinode restructuring operation in a binary search tree.

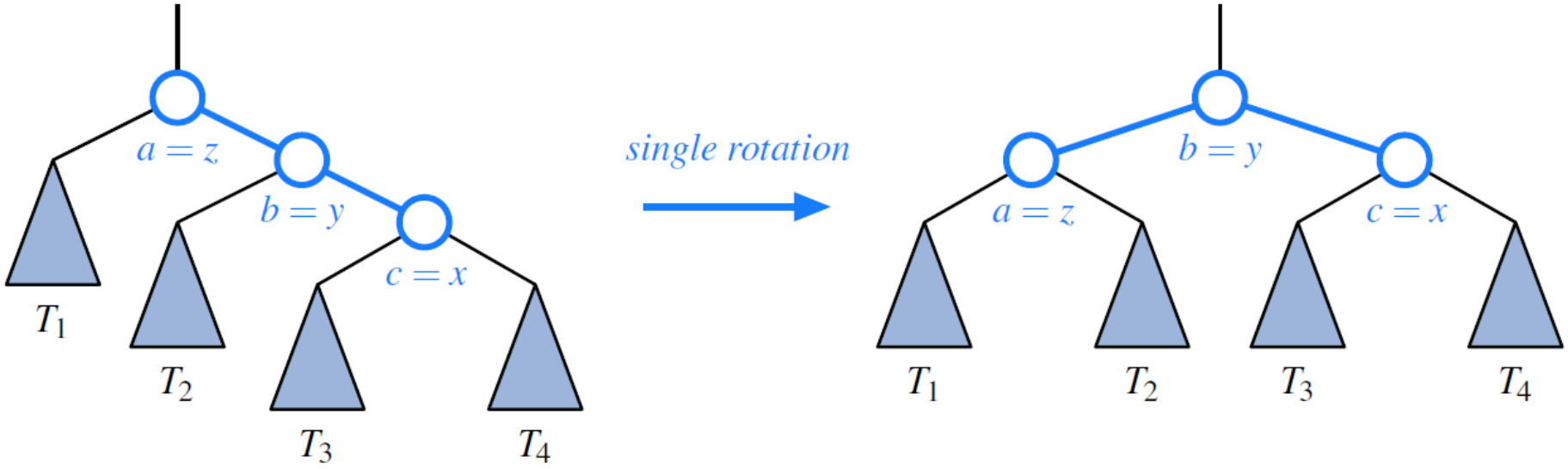
Restructuring (as Single Rotations)

- Single Rotations:

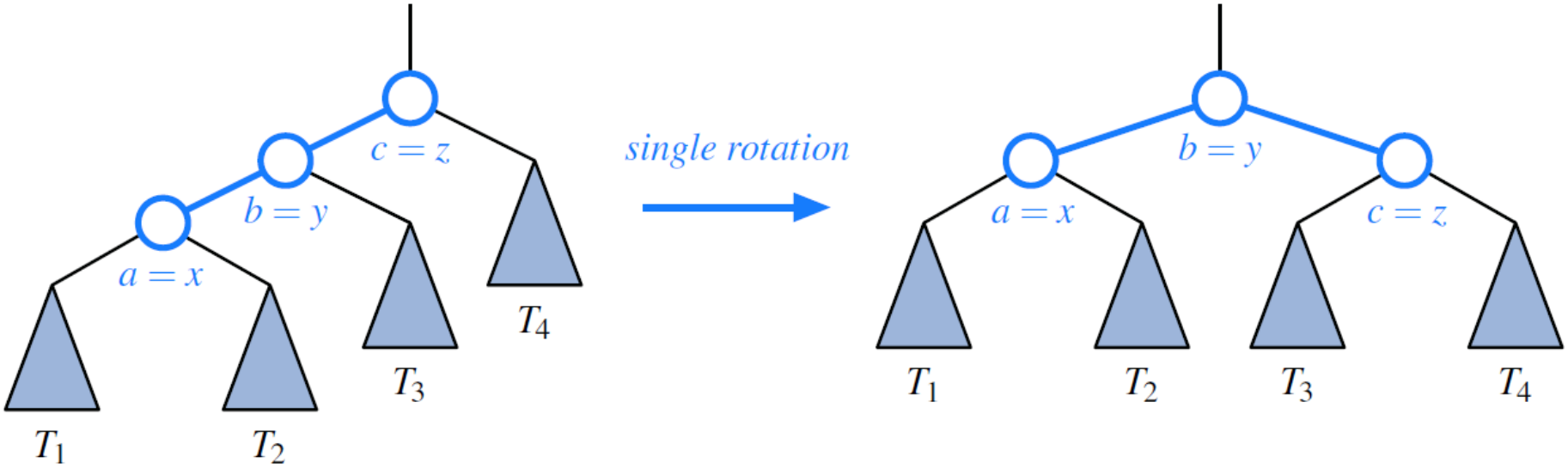




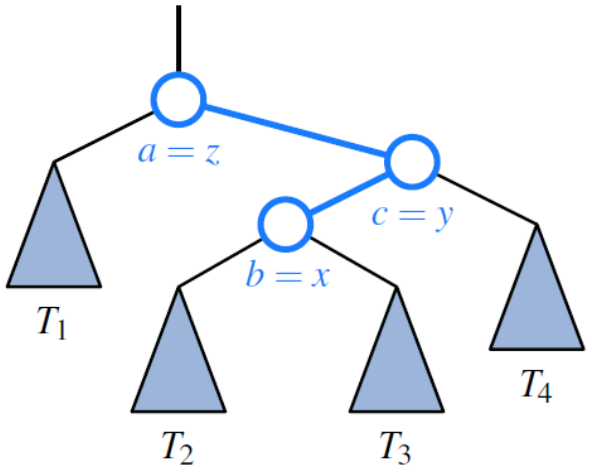
Restructuring (as Single Rotations)

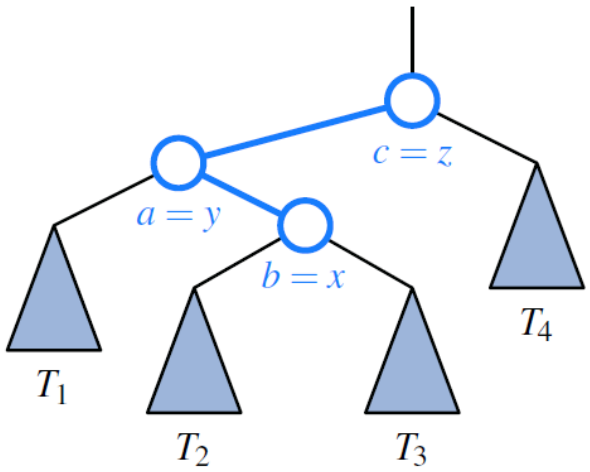


Restructuring (as Single Rotations)

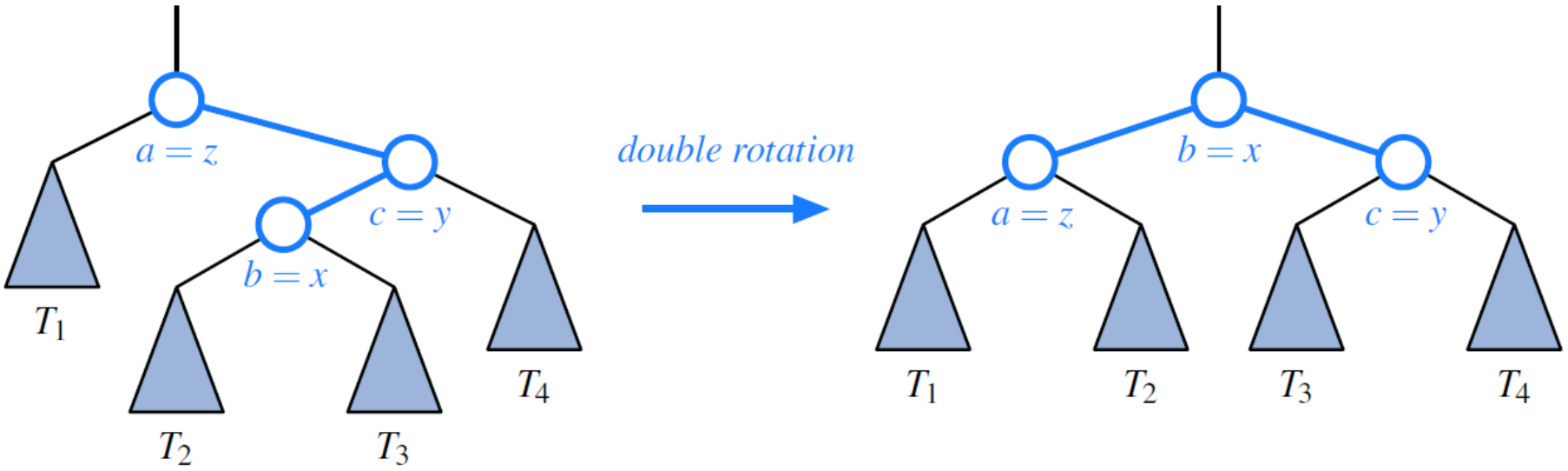


Restructuring (as Double Rotations)





Restructuring (as Double Rotations)



Restructuring (as Double Rotations)