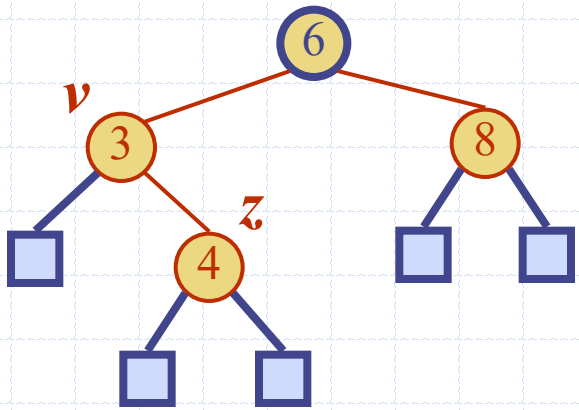
Presentation for use with the textbook Data Structures and Algorithms in Java, 6th edition, by M. T. Goodrich, R. Tamassia, and M. H. Goldwasser, Wiley, 2014

Balanced Search Trees



Reading

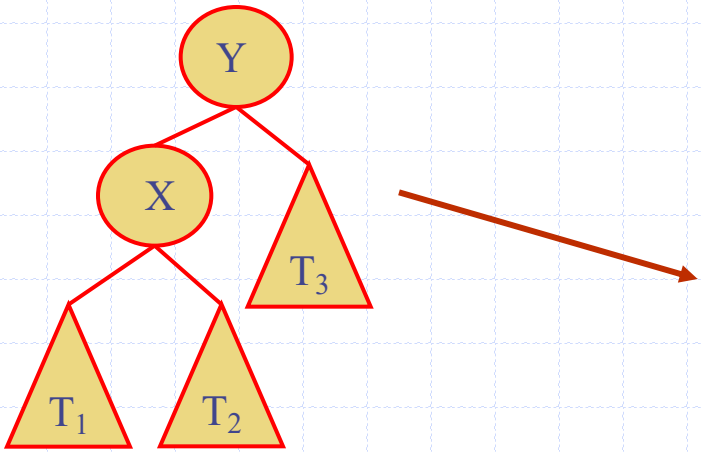
M. T. Goodrich, R. Tamassia and M. H. Goldwasser, *Data Structures and Algorithms in Java*, 6th Edition, 2014.

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

Rotation

重新平衡二叉搜索树的主要操作称为“ 旋转 ”。
在旋转操作中，我们“ 旋转 ”一个子节点使其成为其父节点的上级。

- 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.

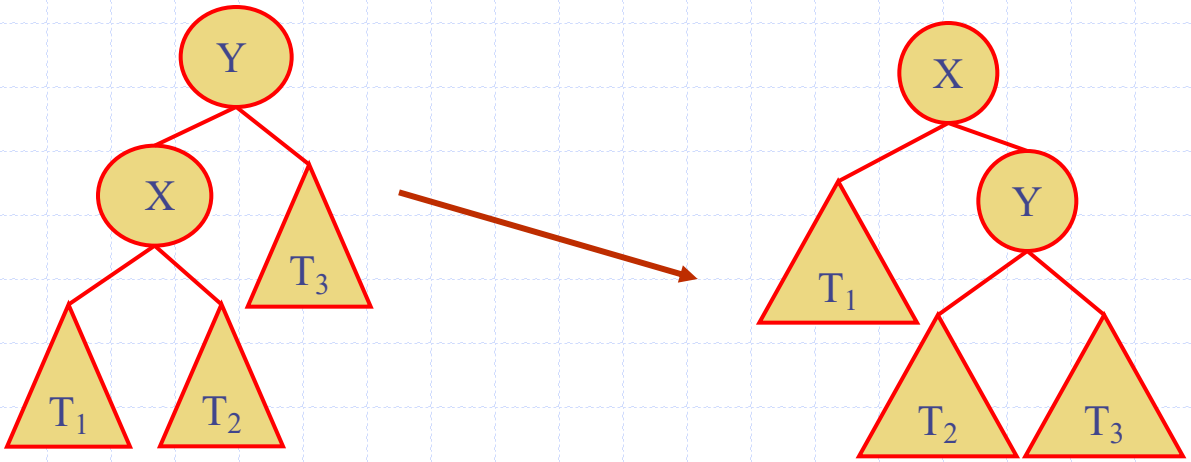


旋转的目标：将X这个左子节点“ 旋转 ”成它父节点Y的上方（变为新的根），
同时保持**二叉搜索树的有序性规则**：左子树所有值 < 根节点 < 右子树所有值

Rotation

满足 BST 的有序性条件：T1 < X < T2 < Y < T3

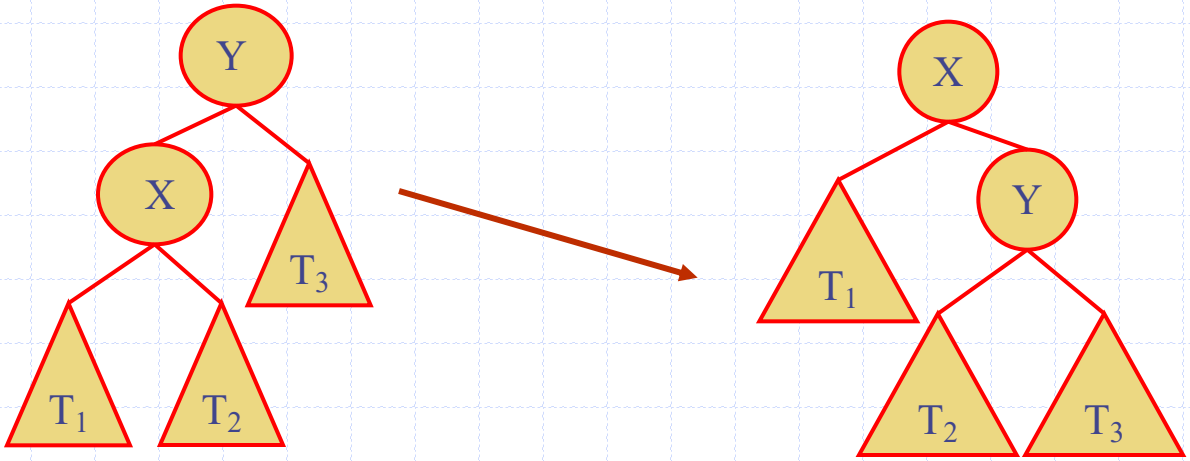
- The primary operation to rebalance a binary search tree is known as a ***rotation***.
- During a rotation, we “rotate” a child to be above its parent.



Rotation

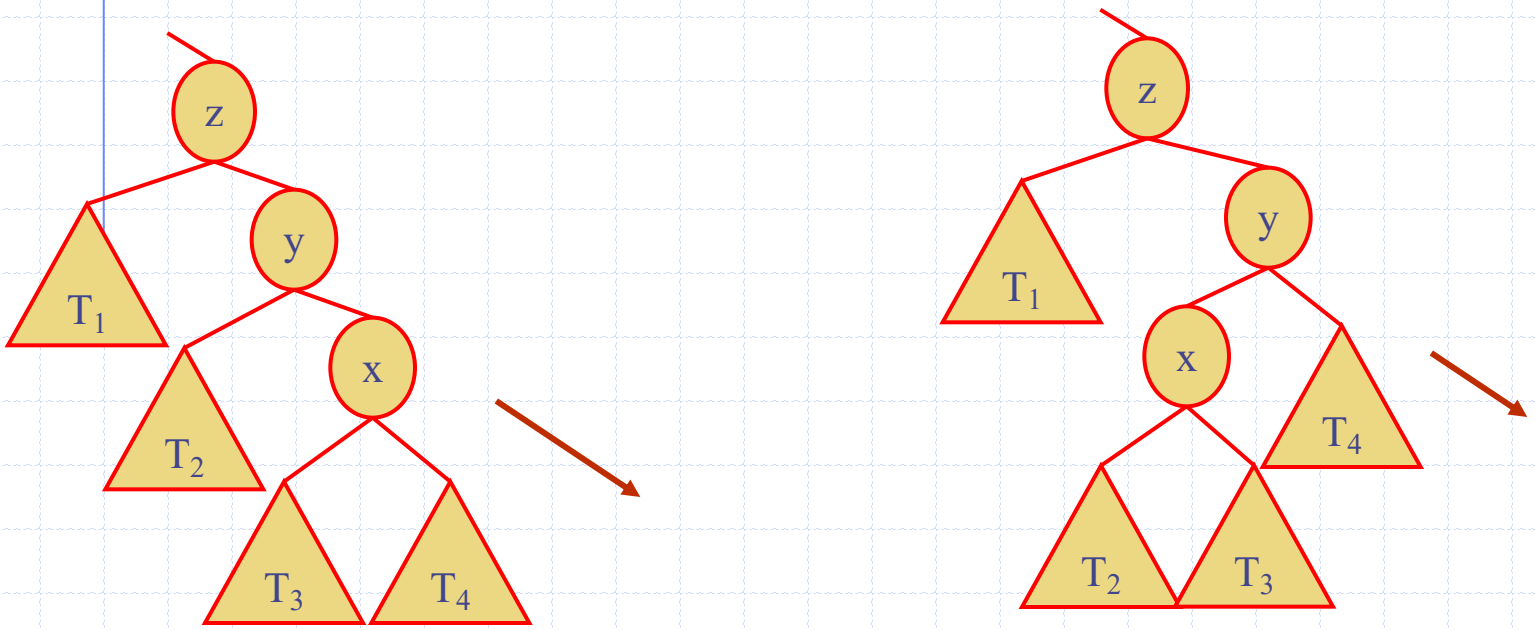
旋转操作的目的：在不破坏二叉搜索树结构的前提下，**改变树的形状**，使其更平衡。
为什么需要旋转？为了**避免高度不平衡的树结构**，比如一直往一边倾斜的树，这样会导致查找效率退化为 O(n)。

- 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

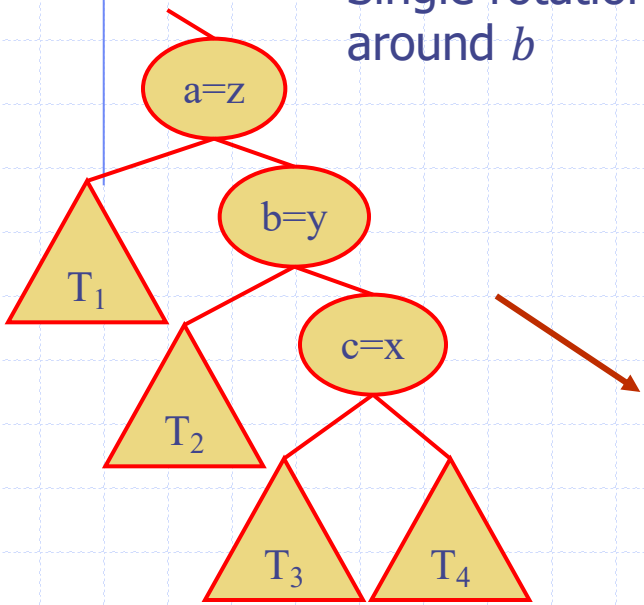
三节点重构是指对三个关键节点（z, y, x）及其四个子树（T, T, T, T）进行**局部调整**，使整棵树保持平衡。通常出现在以下场景：
在AVL树中，当插入新节点导致**某个祖先节点高度差超过1**，就会发生这种重构。
它结合了**旋转**操作，但更广义、更规范化。



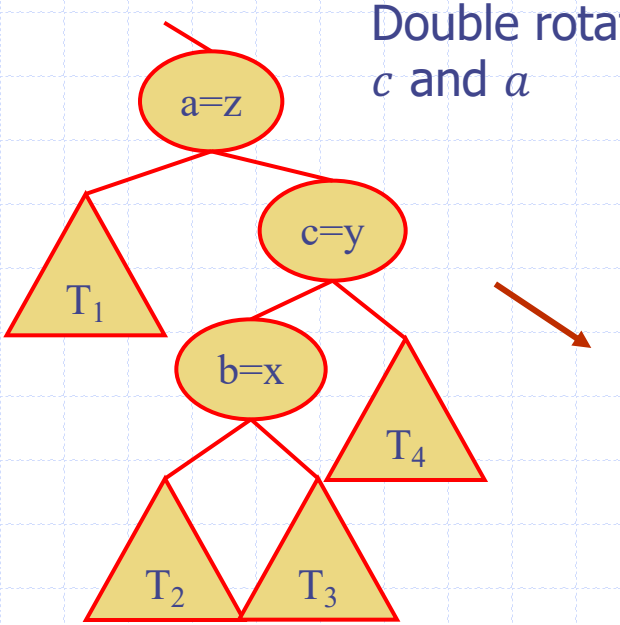
Trinode Restructuring

- 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

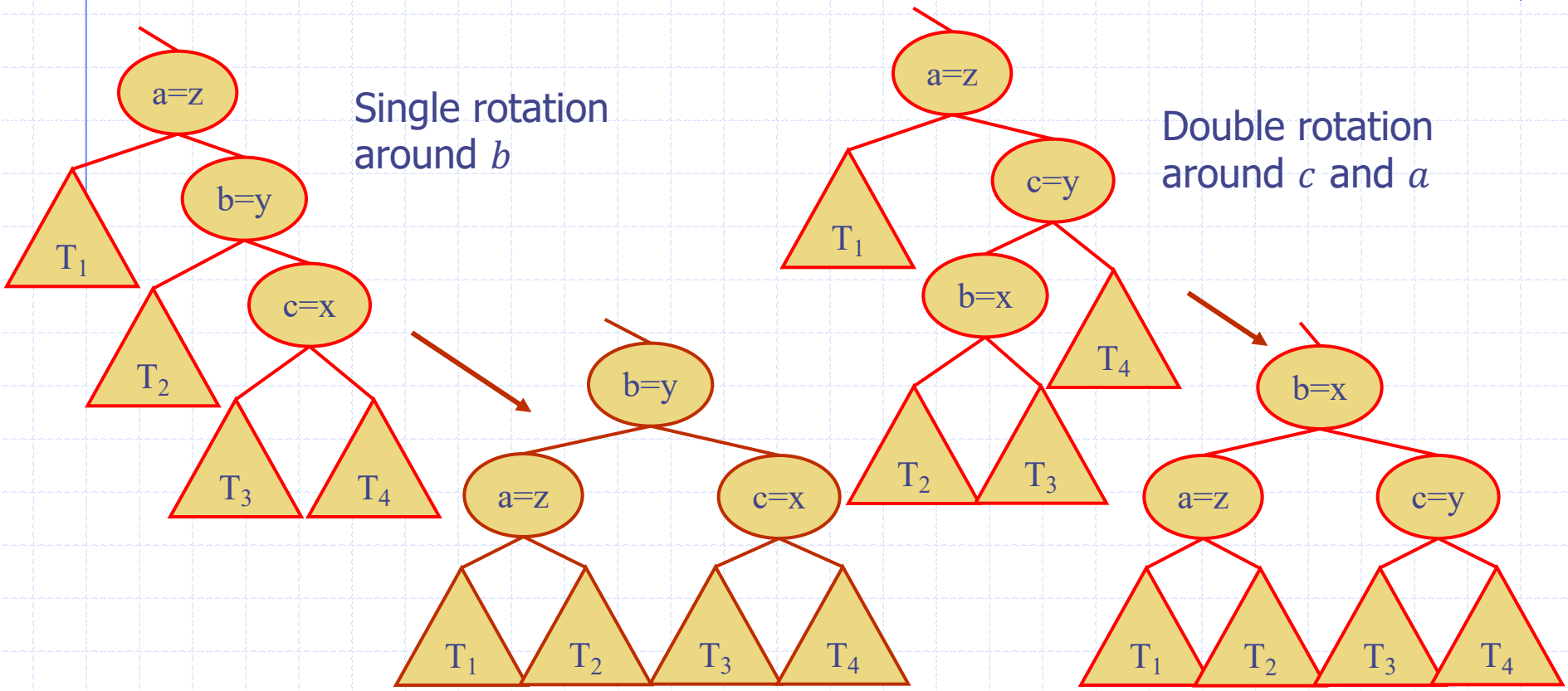


Double rotation around c and a



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.

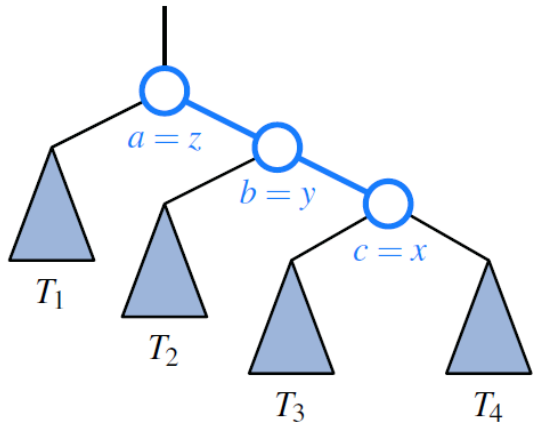
3: Let a be the left child of b and let T_1 and T_2 be the left and right subtrees of a, respectively.

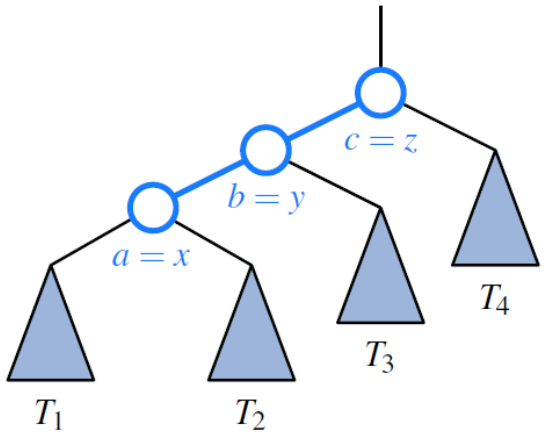
4: Let c be the right child of b and let T_3 and T_4 be the left and right subtrees of c, respectively.

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

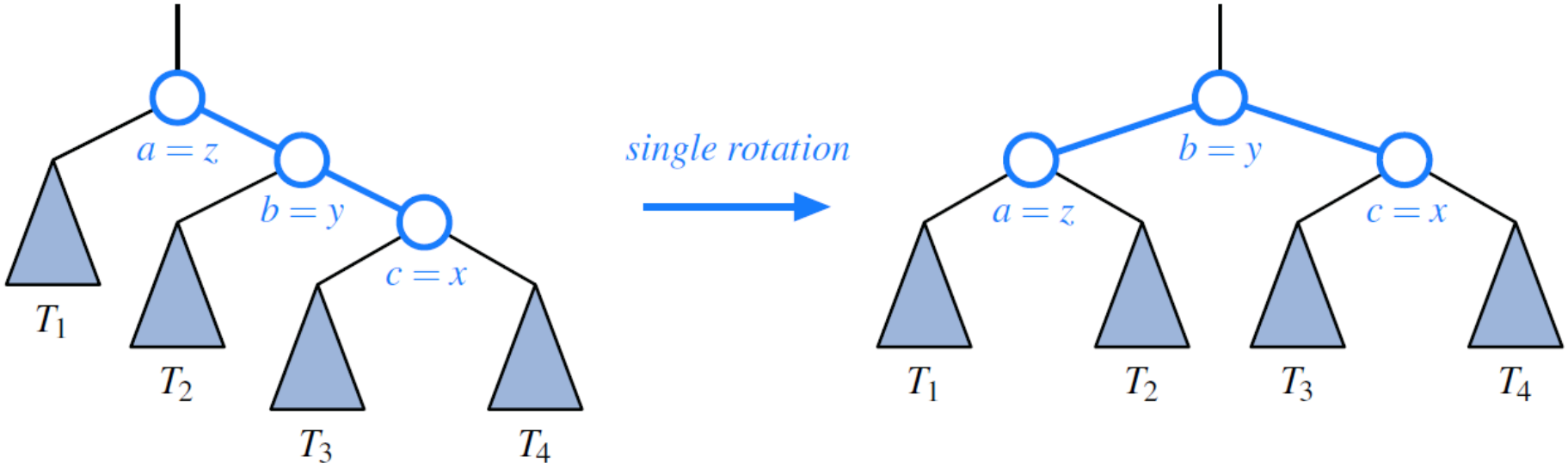
Restructuring (as Single Rotations)

- Single Rotations:

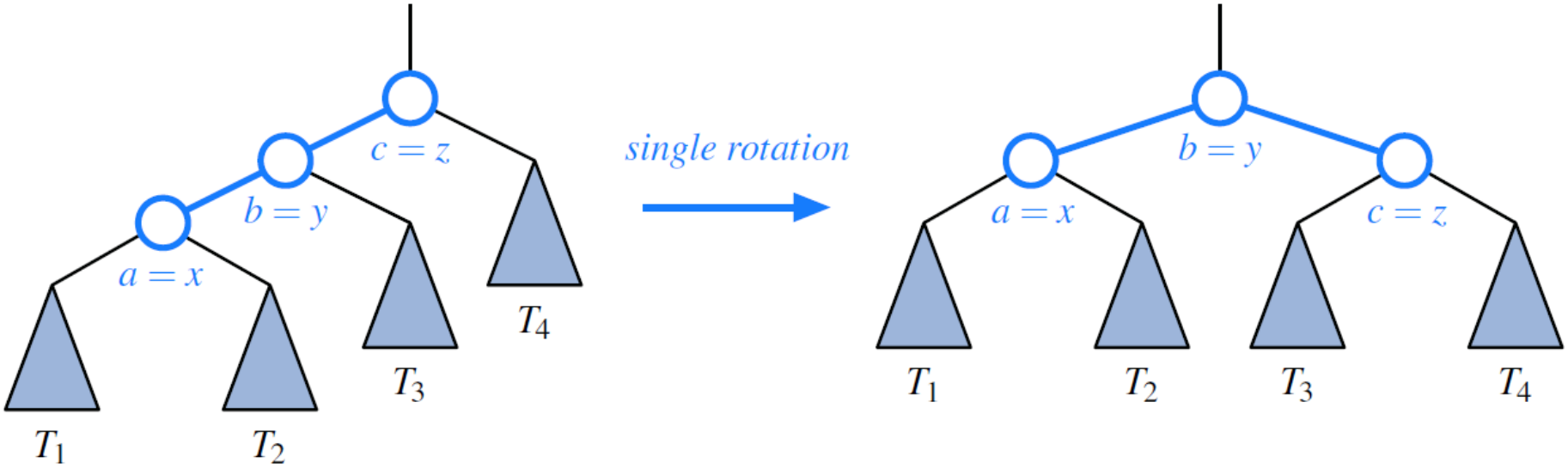




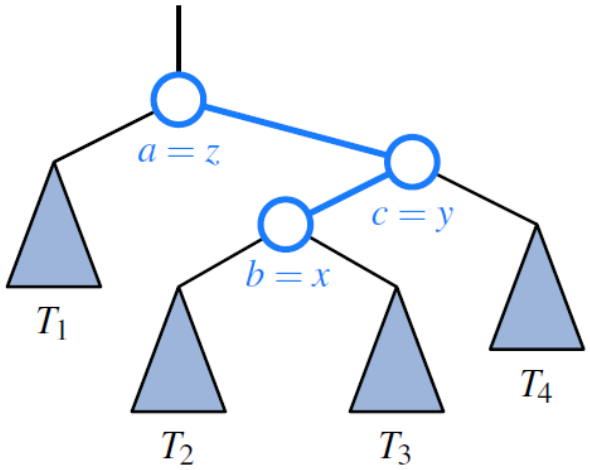
Restructuring (as Single Rotations)

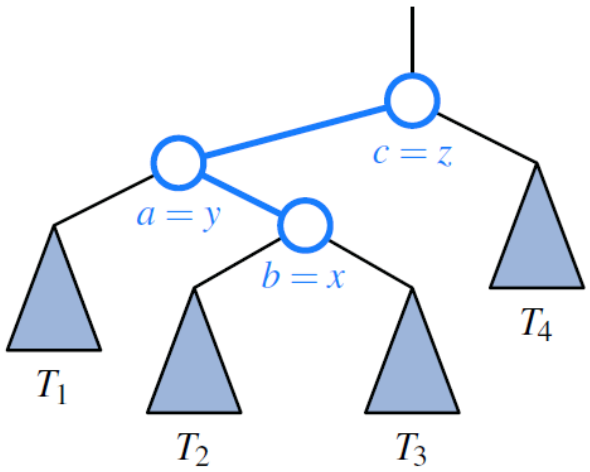


Restructuring (as Single Rotations)

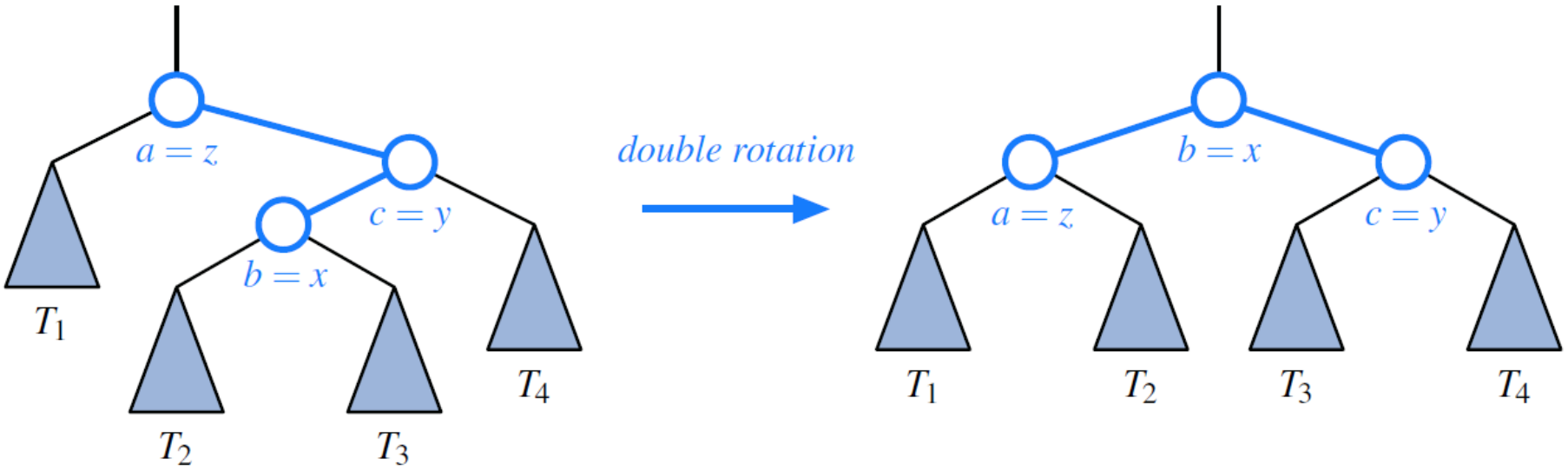


Restructuring (as Double Rotations)





Restructuring (as Double Rotations)



Restructuring (as Double Rotations)