DS286 | 2016-10-14

L18: Balanced & AVL Trees

Yogesh Simmhan
simmhan@cds.iisc.ac.in

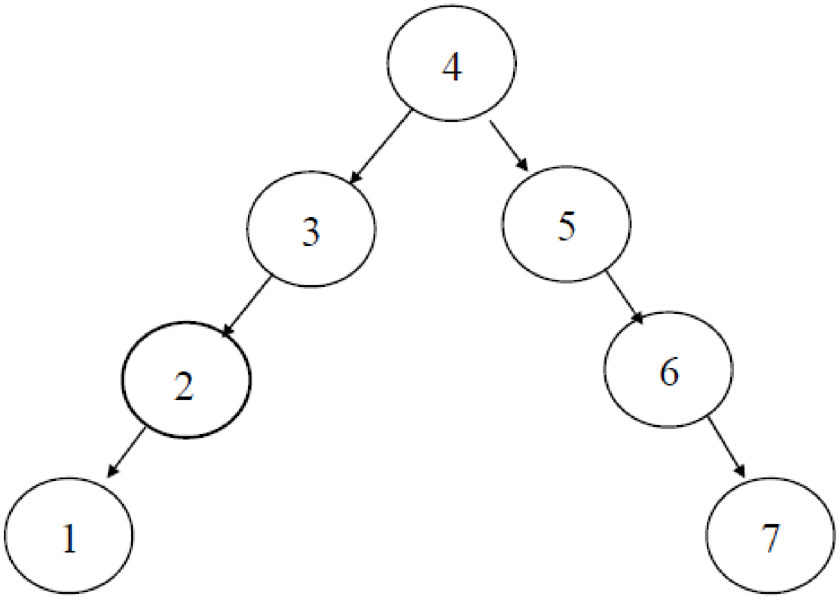
Slides courtesy Venkatesh Babu, CDS

Need for Balancing

- BST have search, insert, delete complexity of O(h)
- But h=n in worst case, i.e., operations are O(n)
- Balancing scheme to limit the height of the tree
- `Balance Factor = |height(left subtree) – height(right subtree)|`

Approaches

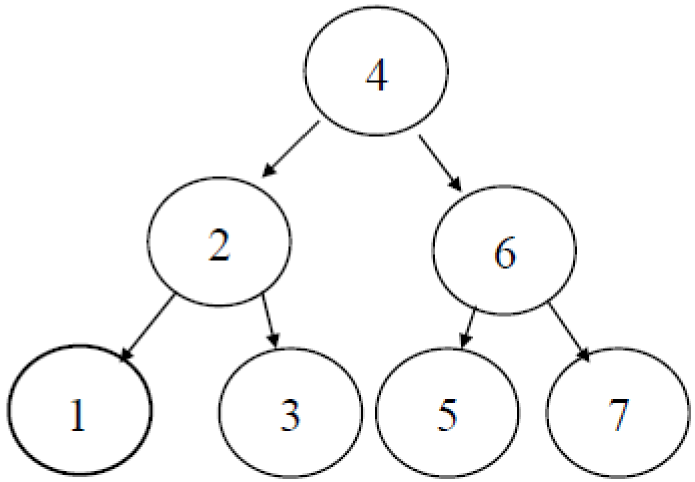
- Balance Factor of root is 0 or 1



Rule #1: Require that the left and right subtrees of the root node have the same height. We can do better.

Approaches

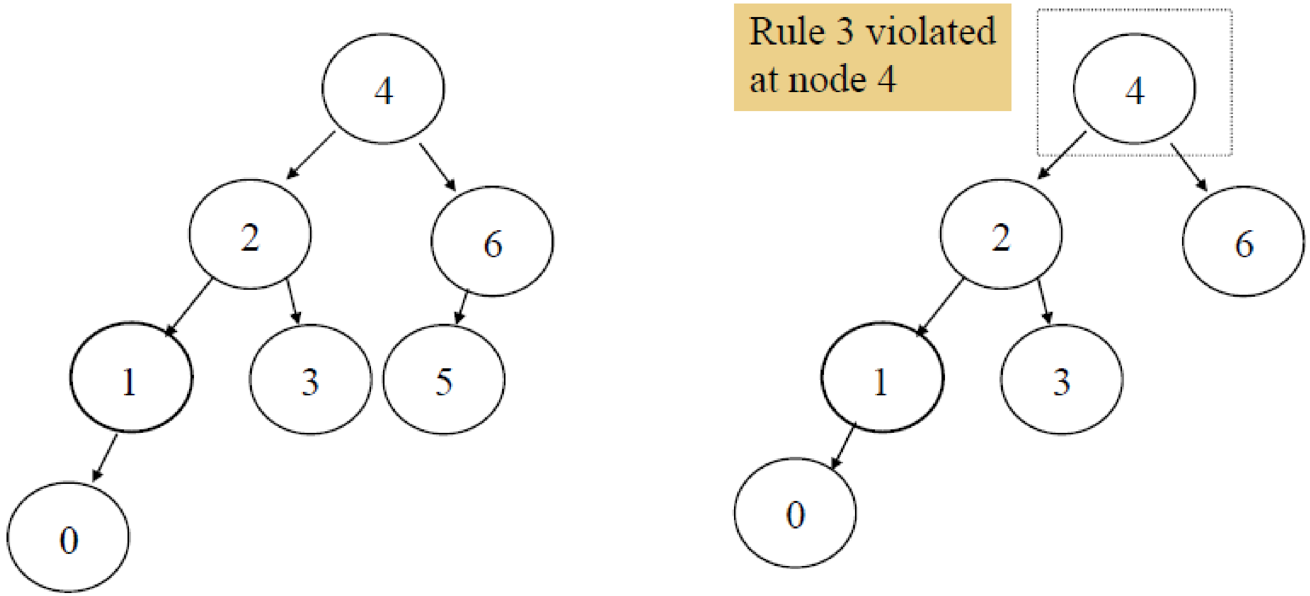
- Balance factor at every node is 0



Rule #2: Require that every node have left and right subtrees of the same height. Too restrictive.

Approaches

- Balance factor at every node is 0 or 1 … **AVL Tree**

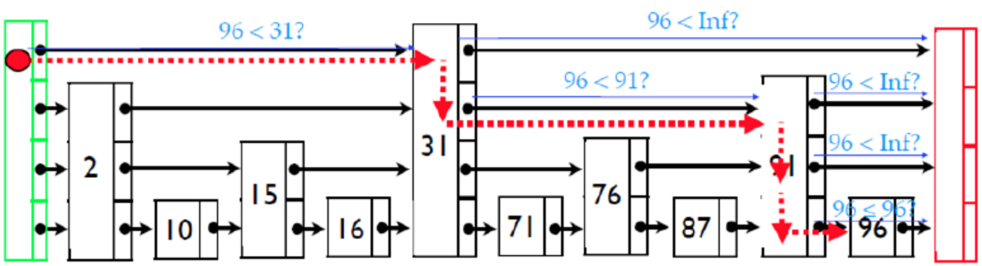


Rule #3: Require that, for every node, the height of the left and right subtrees can differ by most one. The example on the left satisfies rule #3, while the one on the right does not. Why? This rule is a nice compromise between too lax and too restrictive.

Sep 10, 2010 7

Approaches

- Skip Lists (!)
  - Probabilistic, expect same # of nodes in each level
  - Simpler than tree balancing



- "Self Balancing Trees"
- Red-Black Tree
  - Nodes colored red or black. Root is black.
  - Red node must have black children.
  - Number of black nodes from root to leaf is same
- AVL Trees (today)
- B-Trees (next class)

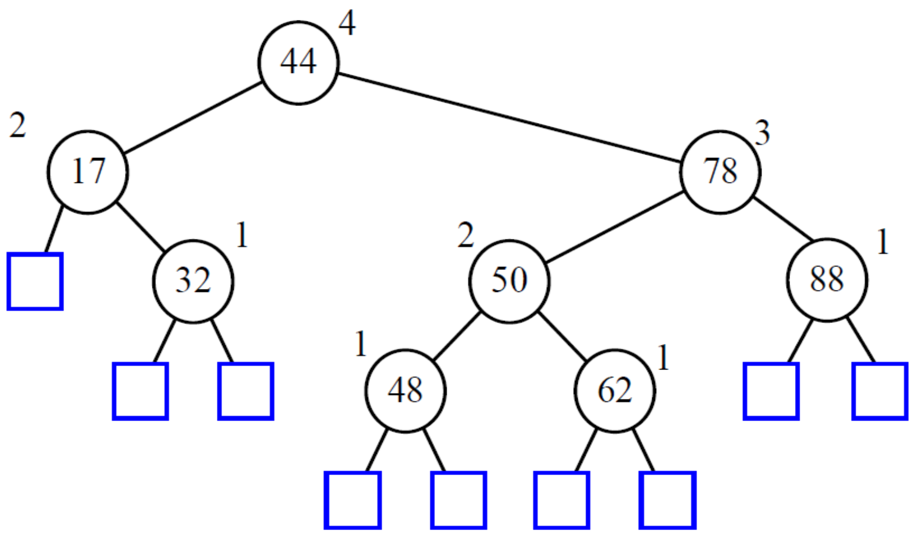
AVL Trees

- Height Balanced Tree
- Georgy **A**delson-**V**elsky and Evgenii **L**andis
  - Soviet Scientists
  - "An algorithm for the organization of information“, 1962
- O(log(n)) insert, delete, search complexity

https://en.wikipedia.org/wiki/AVL_tree

Height Balancing

- Balance Factor = |height(left subtree) – height(right subtree)|
- AVT trees have a balance factor of 1
- Store the heights of subtrees at each level



https://www.cs.purdue.edu/homes/ayg/CS251/

Height of an AVL Tree

- $h=O(\log n)$ for an AVL tree with n nodes
- $N(h)$ is the minimum number of nodes in an AVL tree of height h

$$N(1) = 1, N(2) = 2$$

- Since the heights of a node's children differ at most by 1

$$N(h) = 1 + N(h-1) + N(h-2)$$

- Since $N(h-1) > N(h-2)$

$$N(h) > 2.N(h-2)$$

$$N(h) > 4.N(h-4)$$

$$N(h) > 2^i.N(h-2i)$$

$$N(h) > 2^{h/2}$$

$$h<2.\log(N(h))$$

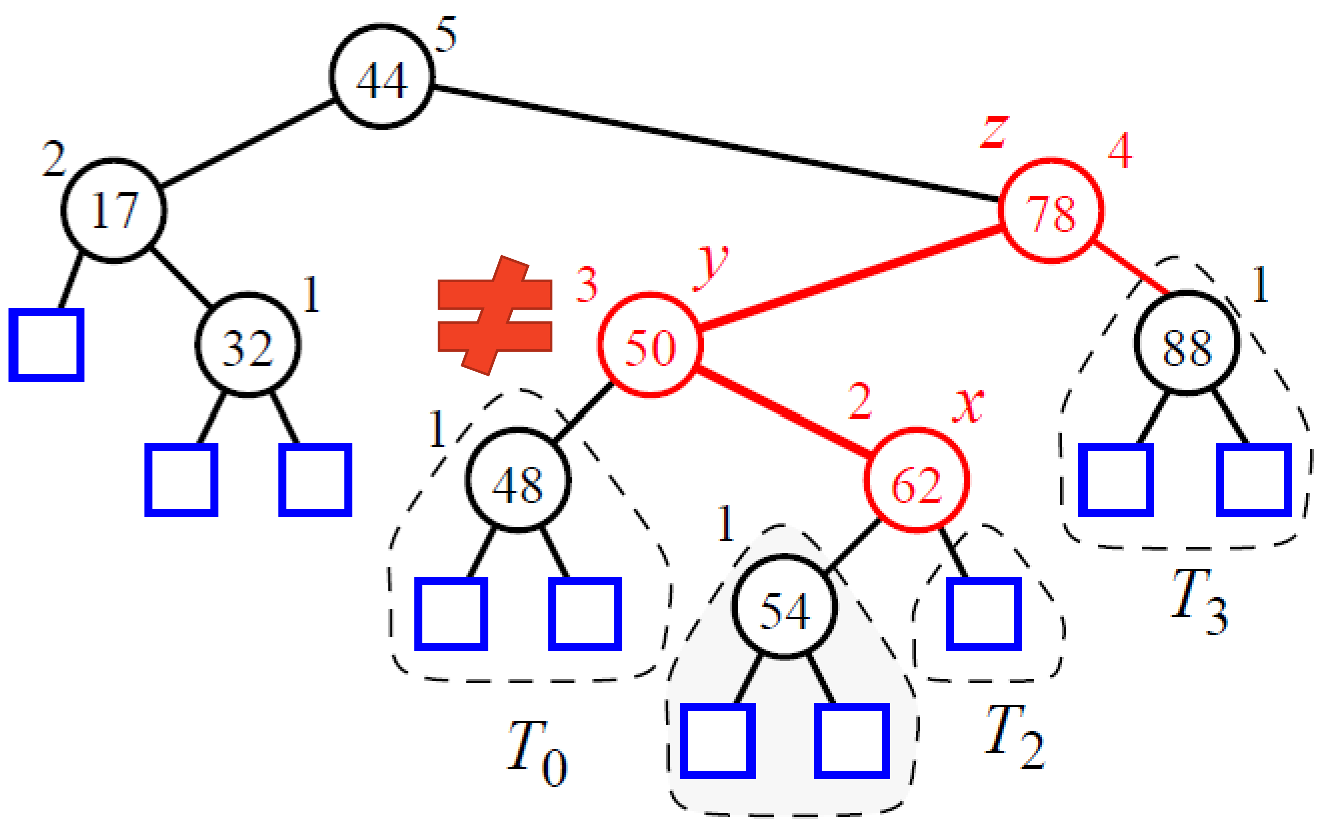
- So, h = O(log(n))

https://www.cs.purdue.edu/homes/ayg/CS251/

Repairing AVL Trees

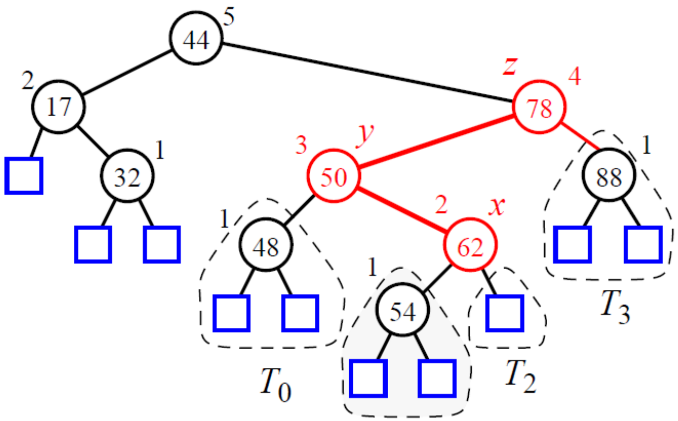
- Insertions or deletions can cause the height balance property to be violated
  - Balance factor may become 2
  - *Not more than that for a single insert/delete*
- Repair the tree when imbalance is encountered
  - Go up the tree to parent or grandparent
  - “Rotate the nodes” to restore balanced property

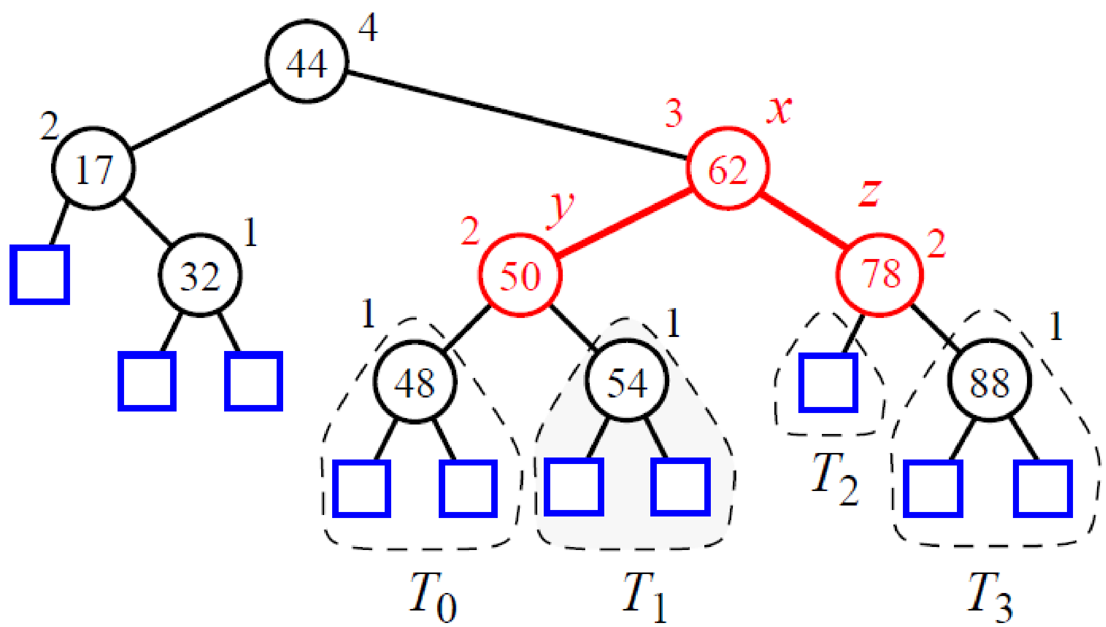
Repairing AVL Trees



https://www.cs.purdue.edu/homes/ayg/CS251/

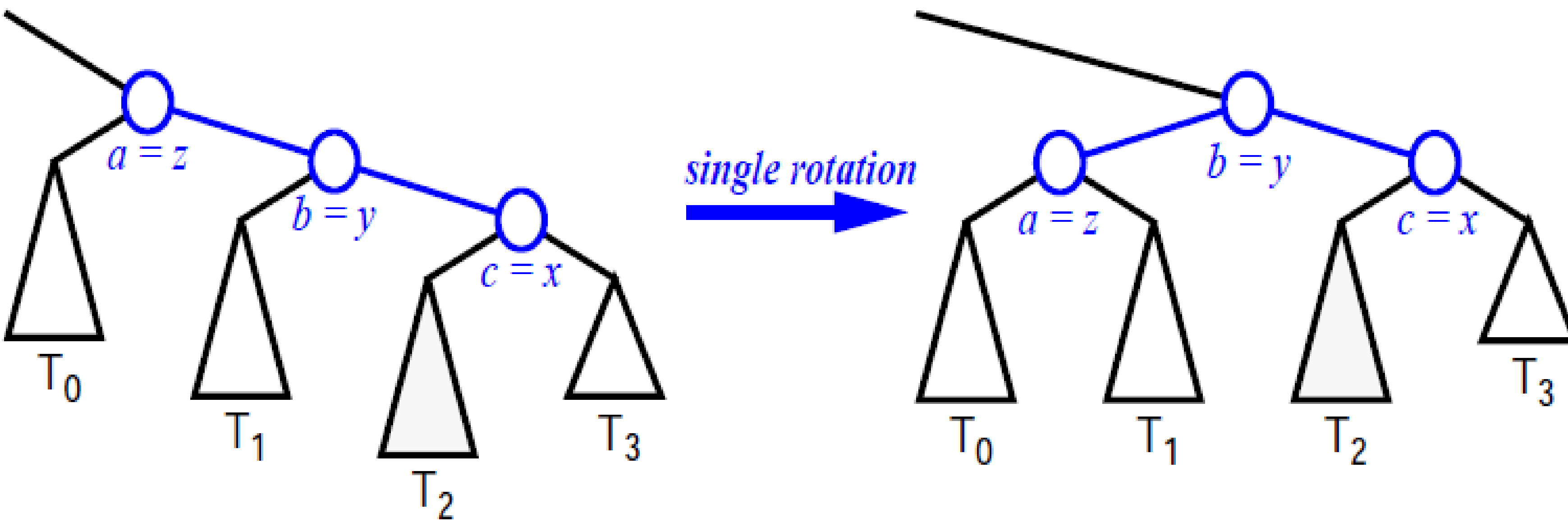
Repairing AVL Trees





https://www.cs.purdue.edu/homes/ayg/CS251/

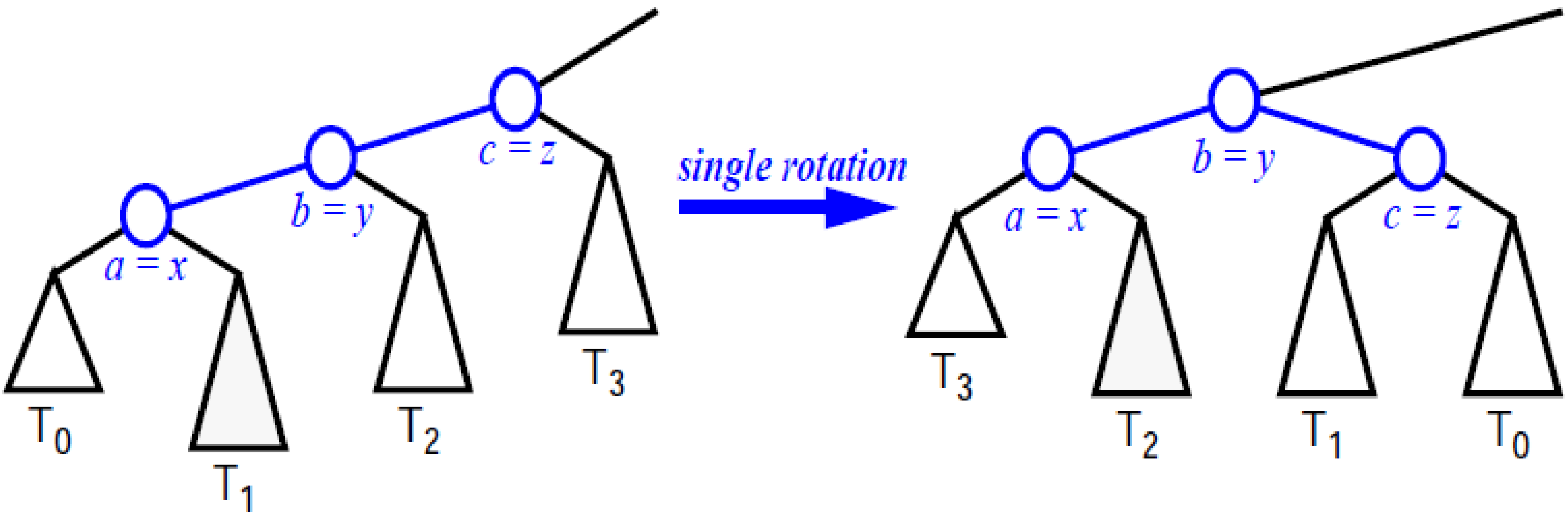
AVL Rotations: Outside Case, Single Rotation



- After single rotation, the new height of the entire subtree is exactly the same as the height of the original subtree prior to the insertion of the new data item

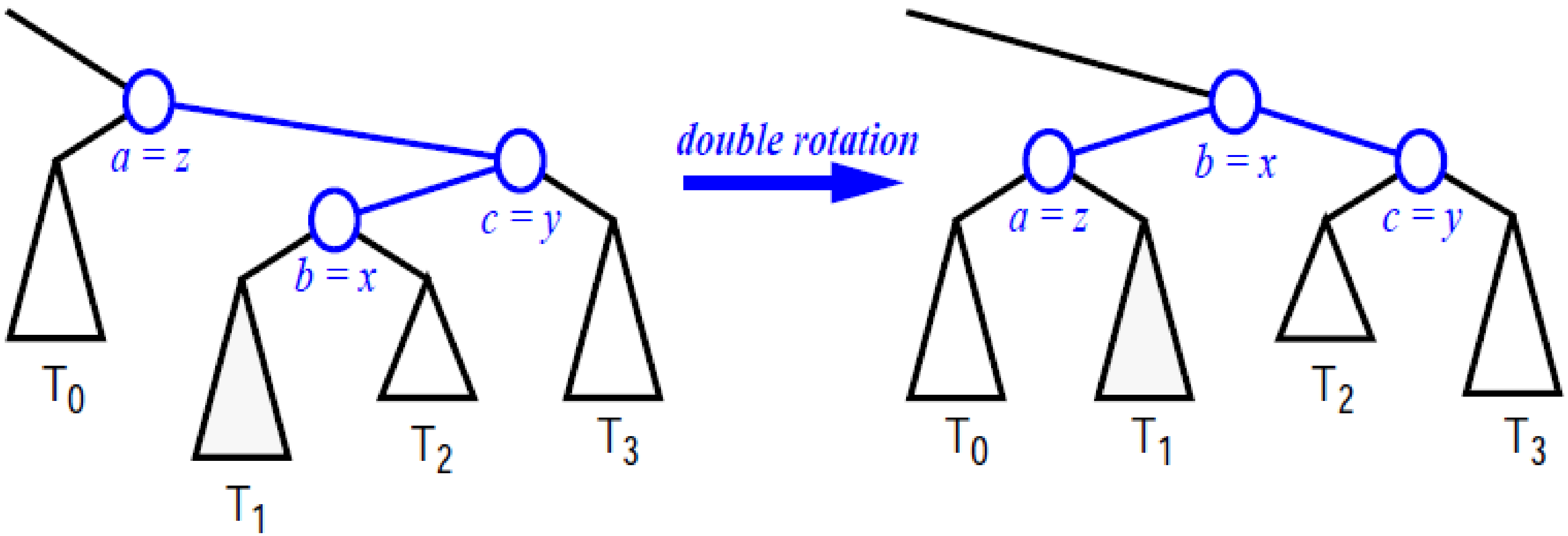
https://www.cs.purdue.edu/homes/ayg/CS251/
SE-286: Data Structures, Virendra Singh, IISc, 2010

AVL Rotations: Outside Case, Single Rotation



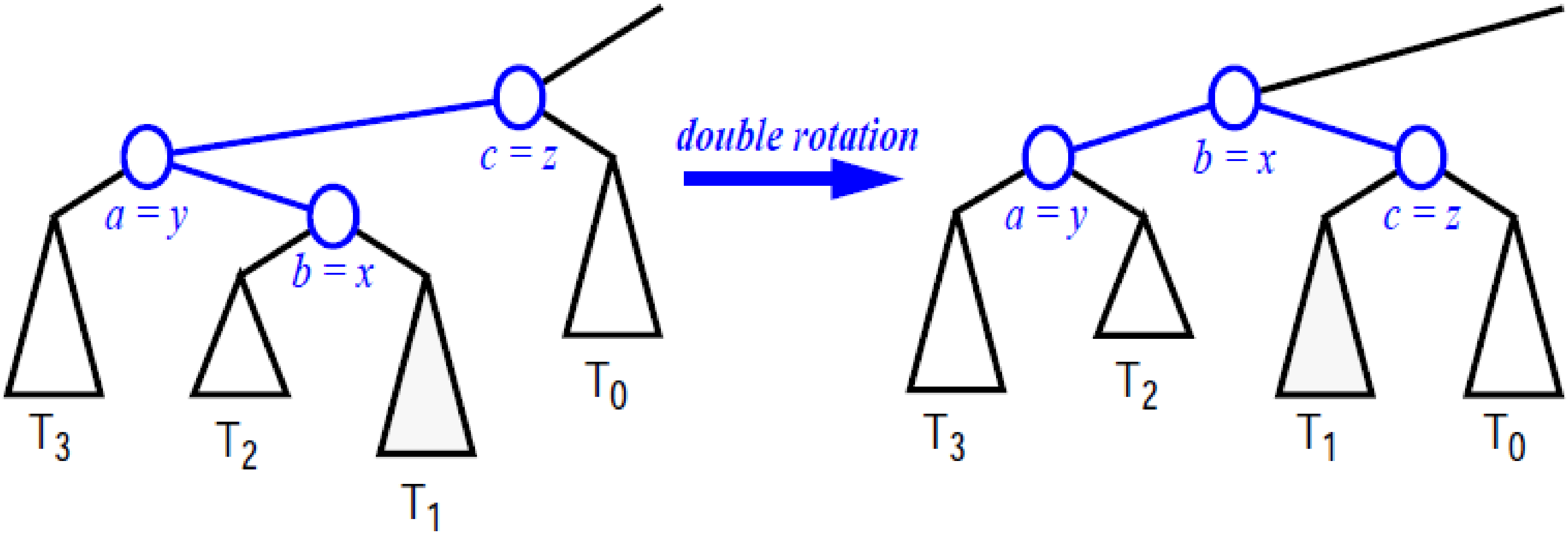
https://www.cs.purdue.edu/homes/ayg/CS251/

AVL Rotations: Inside Case, Double Rotation



https://www.cs.purdue.edu/homes/ayg/CS251/

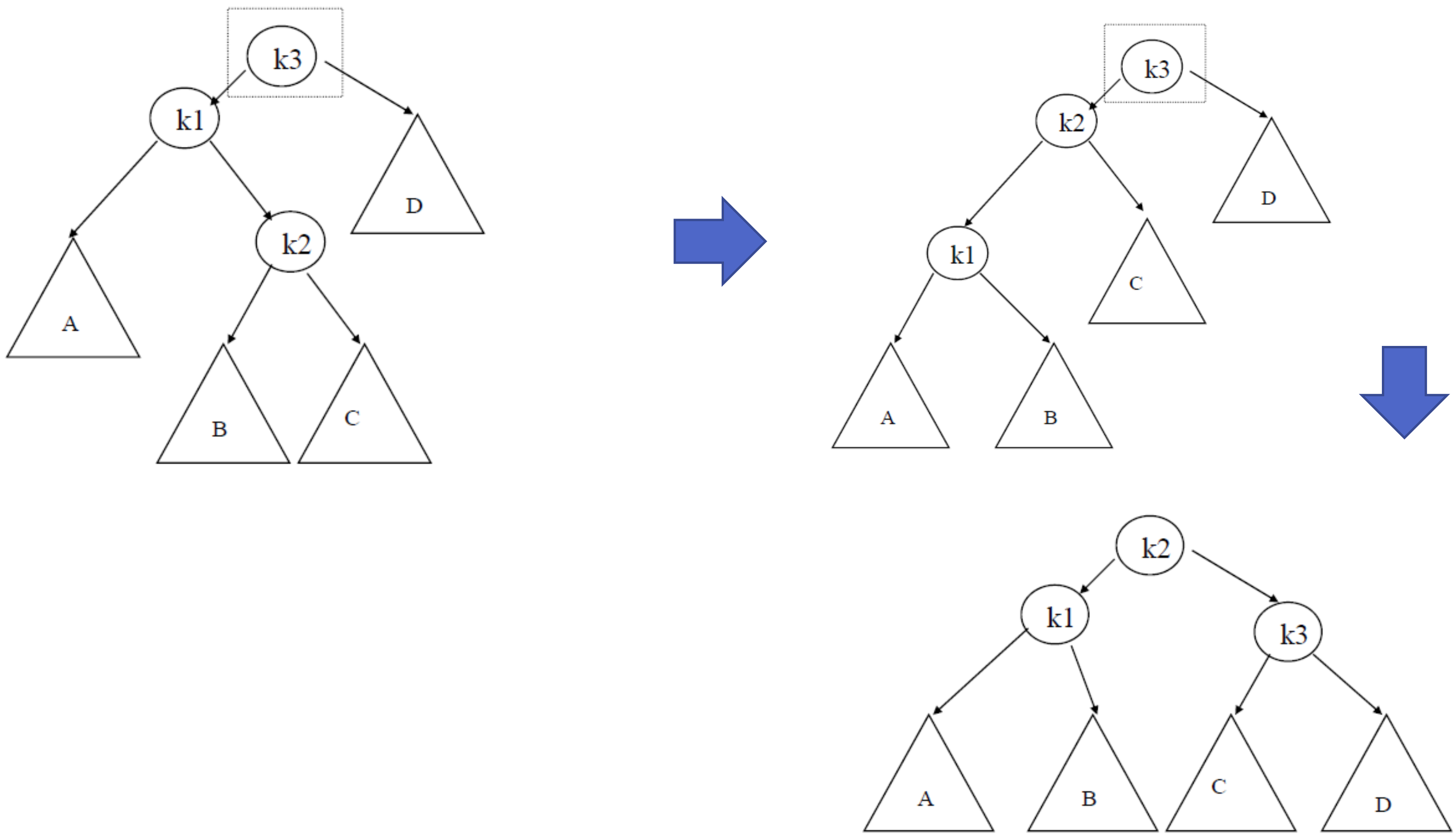
AVL Rotations: Inside Case, Double Rotation



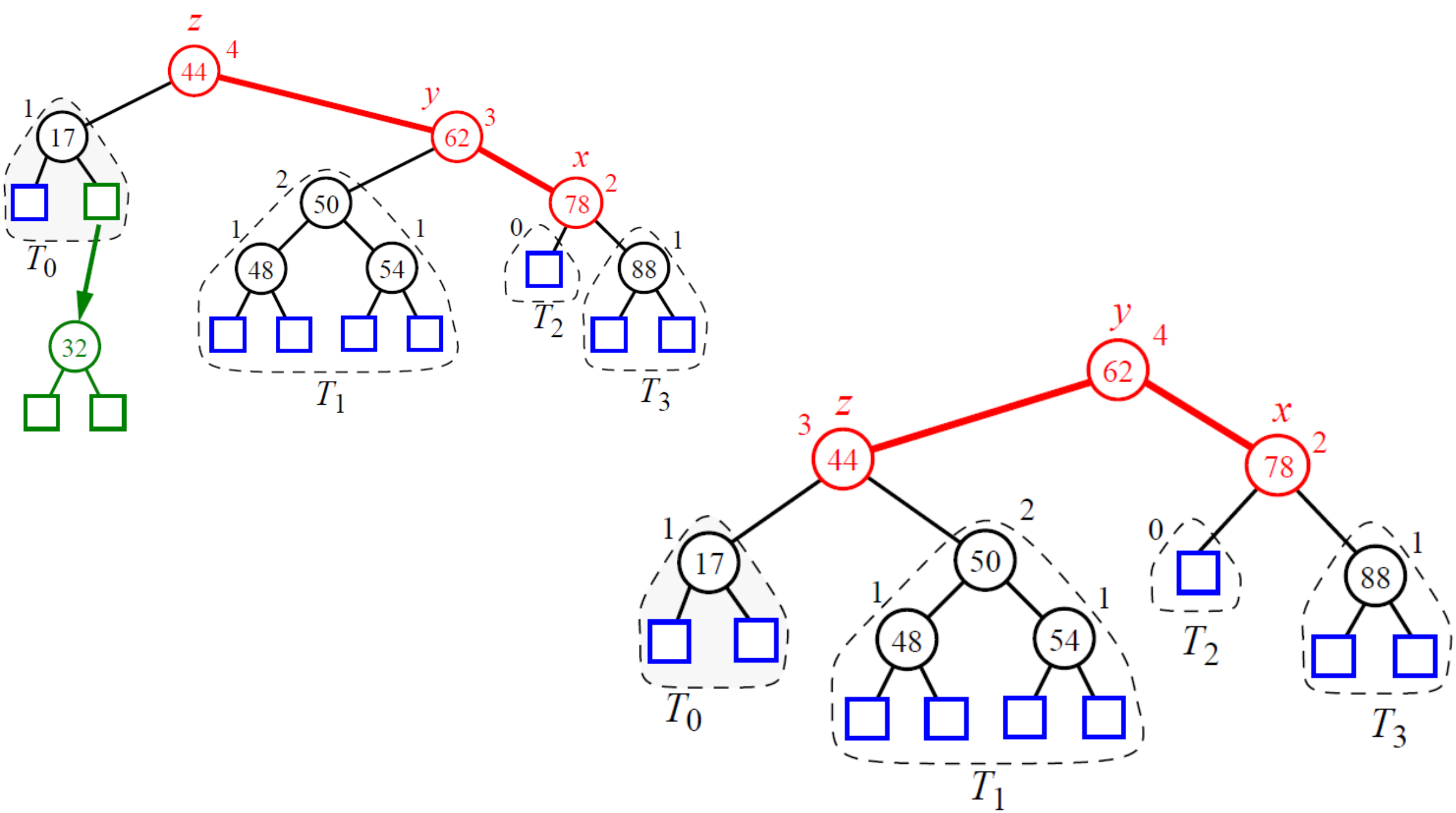
As with the single rotations, double rotations restore the height of the subtree to what it was before the insertion.

https://www.cs.purdue.edu/homes/ayg/CS251/
SE-286: Data Structures, Virendra Singh, IISc, 2010

AVL Rotations:
Double Rotation = 2 Single Rotations



AVL Rotations: Deletion



https://www.cs.purdue.edu/homes/ayg/CS251/

Tasks

- Self study
  - **Read**: AVL Rotations (online sources)
- Finish Assignment 4 by Wed Oct 26 *(75 points)*
- Make progress on CodeChef (100 points)

Questions?