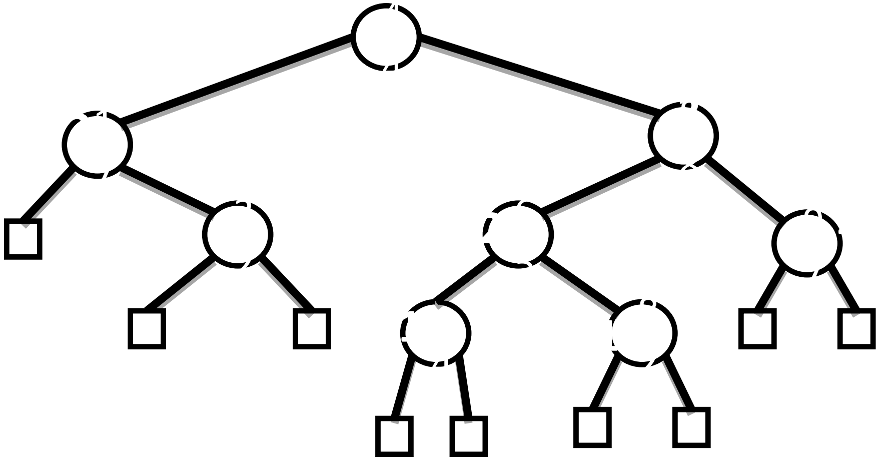
AVL Tree



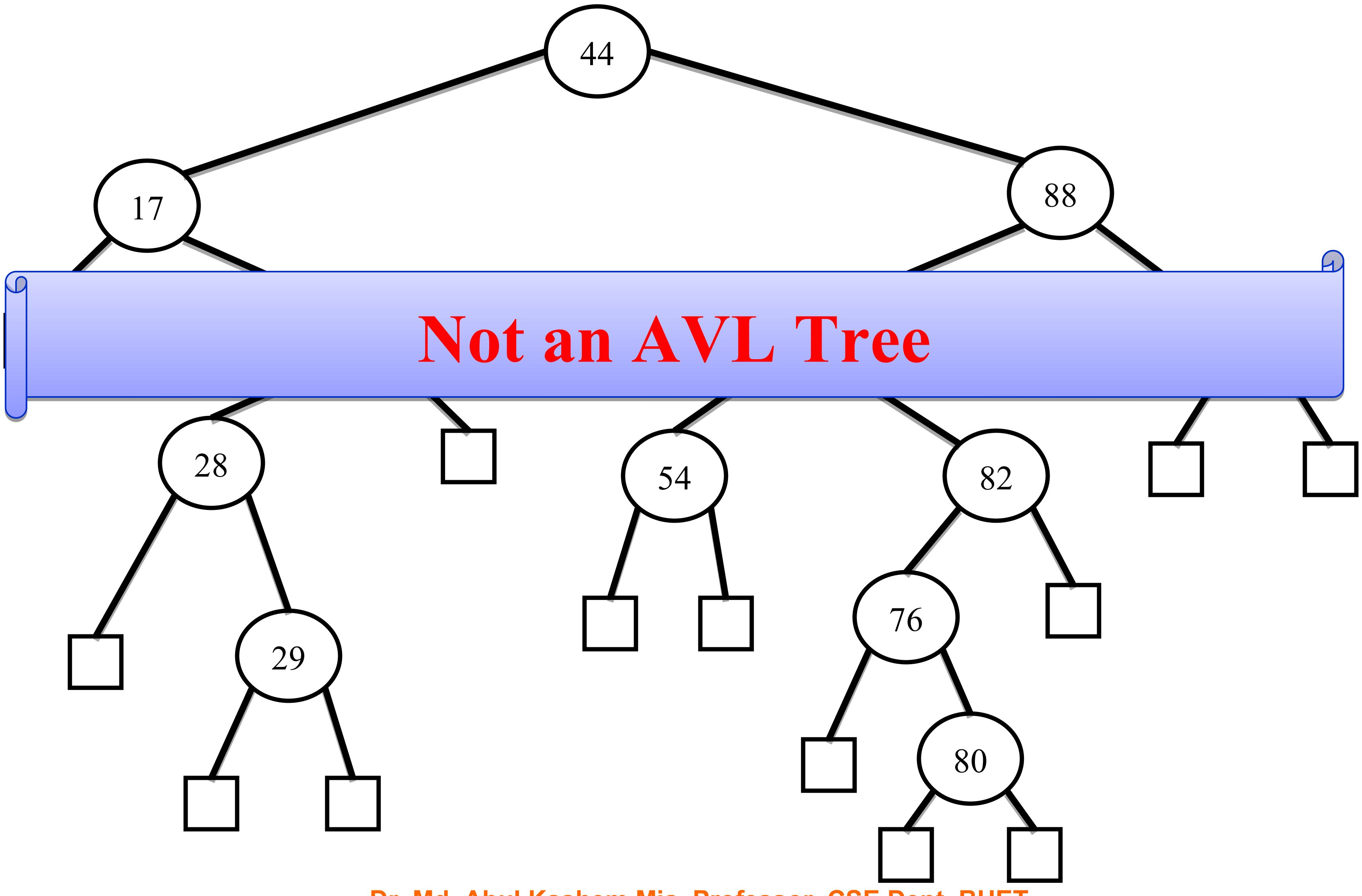
AVL Tree: Definition

- An **AVL tree** is a binary search tree that is *height balanced*: for each node x, the heights of the left and right subtrees of x differ by at most 1.
  - A subtree of an AVL tree is itself an AVL tree.

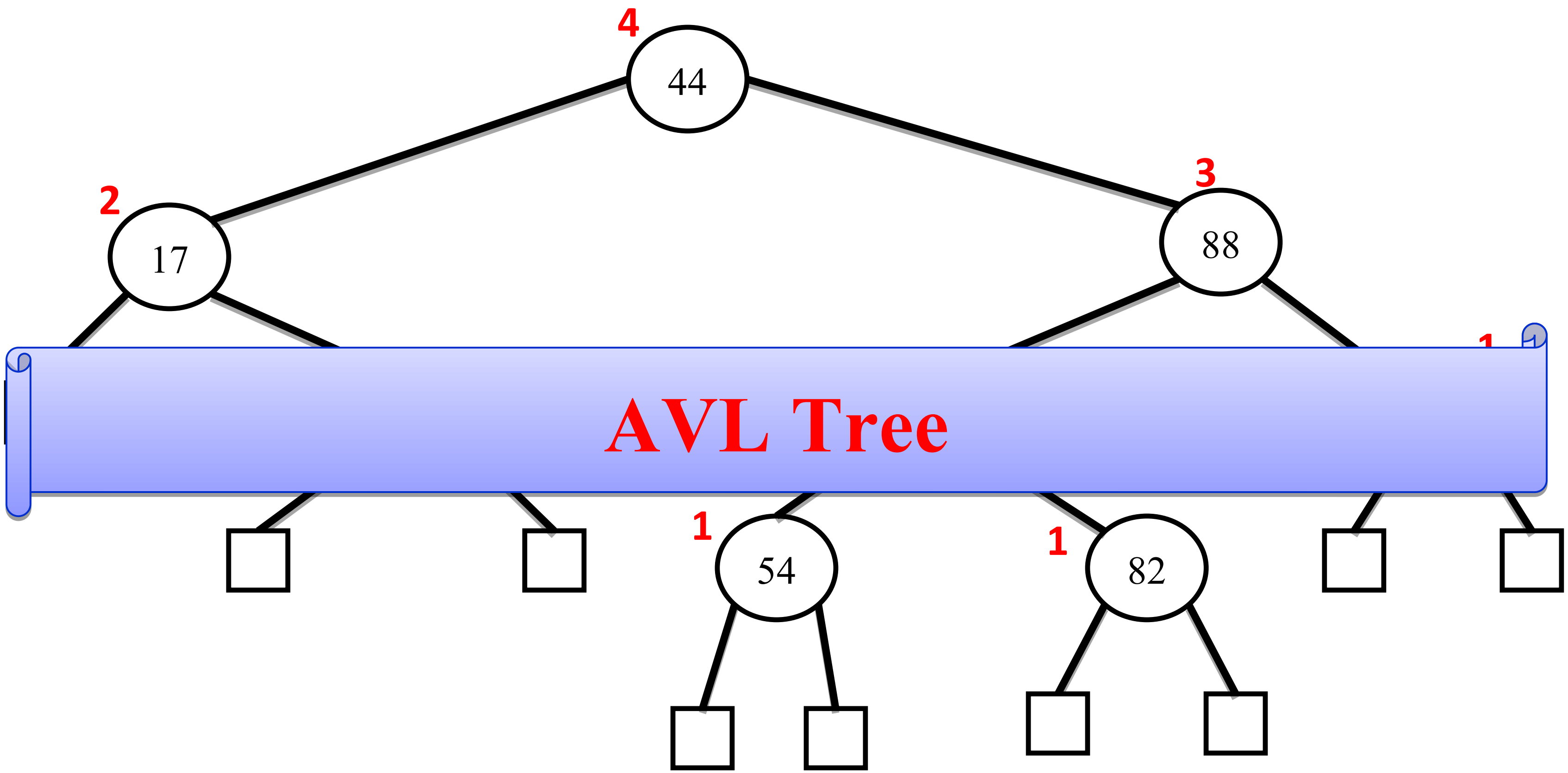
- **Height-Balance Property**: For every internal node v of T, the heights of the children of v can differ by at most 1.
  - Any Binary Search Tree (BST) that satisfies the height-balance property is said to be an *AVL tree*.

- Named after its two Soviet inventors –
  - G.M. **A**delson-**V**elskii and E.M. **L**andis.

Binary Search Tree



Binary Search Tree



AVL Tree

- **Proposition:** The height of an AVL tree T storing n elements is $O(\log n)$.

Justification:

Let, the minimum number of internal nodes be $n(h)$, where h is the height of the tree.

so, $n(1) = 1$; $n(2) = 2$; and
$n(h) = 1 + n(h\text{-}1) + n(h\text{-}2)$ for $h \geq 3$.

Since $n(h)$ is a strictly increasing function, we have $n(h\text{-}1) > n(h\text{-}2)$.

Then $n(h) > 2.n(h - 2)$
$> 4.n(h - 4)$
...
$> 2^i.n(h - 2i)$.

AVL Tree

We pick i so that h -$2i$ is equal to 1 or 2. That is, we pick

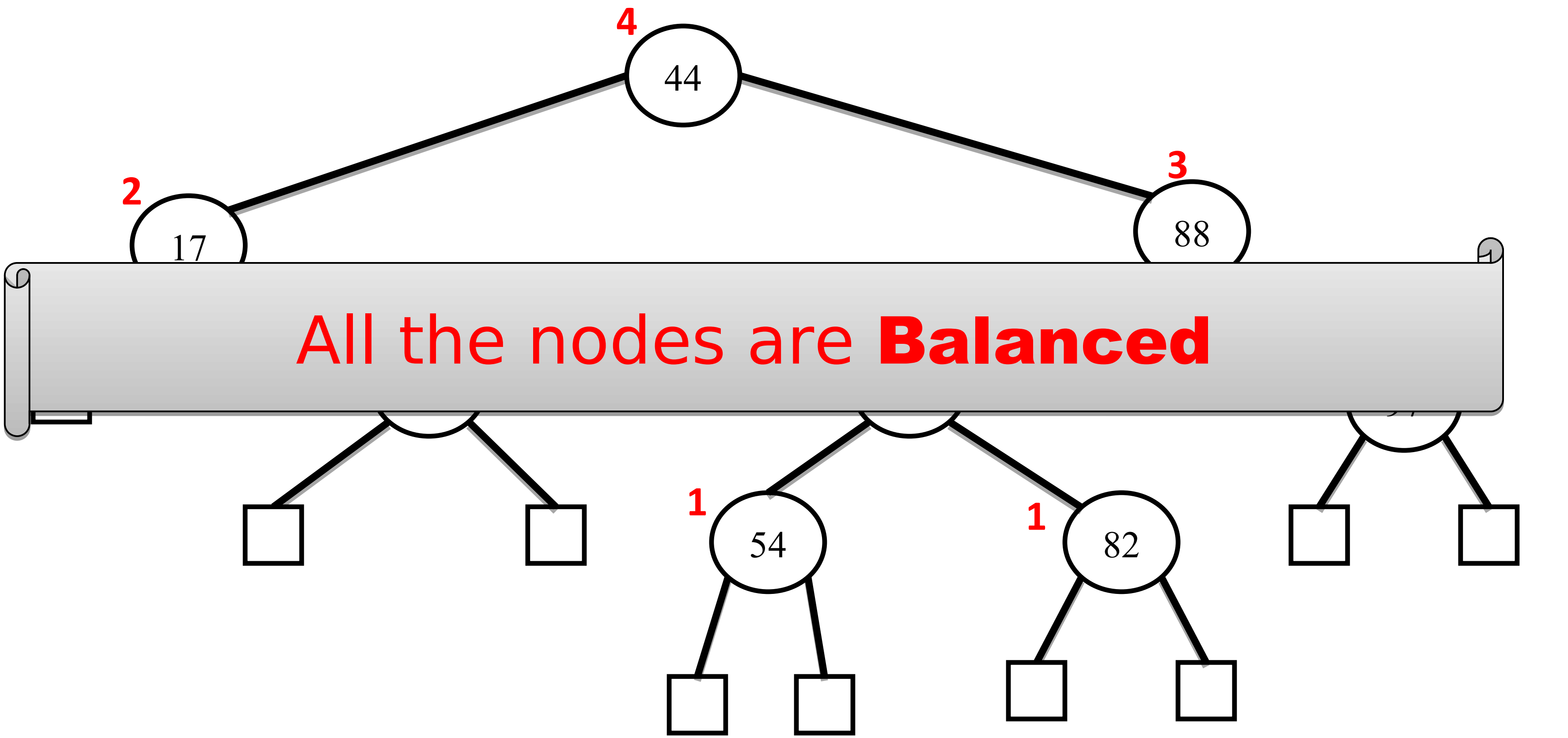
$$i = \left\lceil \frac{h}{2} \right\rceil - 1$$

$$\begin{aligned}
\text{so, } n(h) &> 2^{\left\lceil \frac{h}{2} \right\rceil - 1} \cdot n\left(h - 2\left\lceil \frac{h}{2} \right\rceil + 2\right) \\
&\geq 2^{\left\lceil \frac{h}{2} \right\rceil - 1} \cdot n(1) \\
&\geq 2^{\frac{h}{2} - 1}
\end{aligned}$$

$$\Rightarrow \log n(h) \geq \frac{h}{2} - 1$$

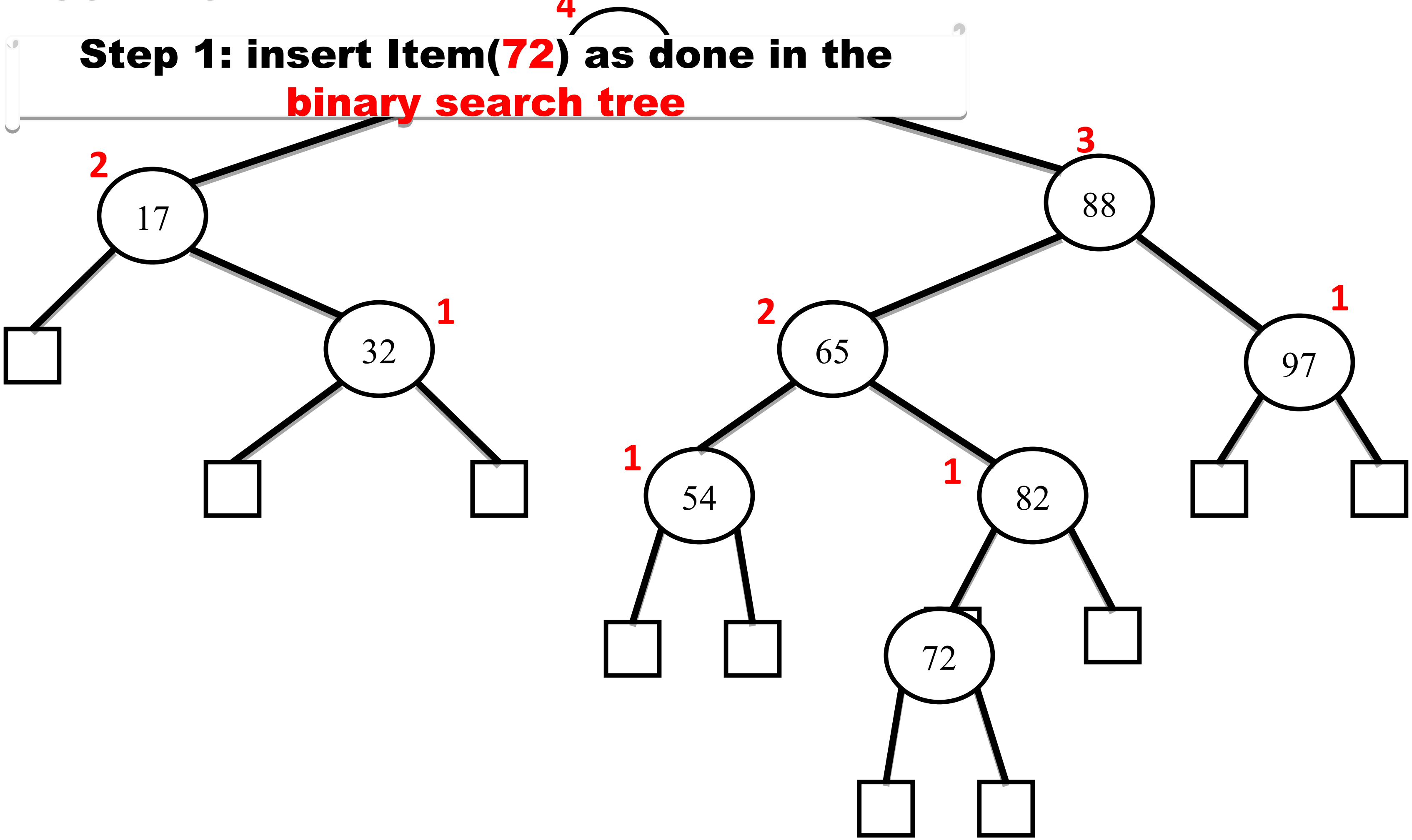
$$\Rightarrow h \leq 2 \log n(h) + 2 \Rightarrow h \leq O(\log n)$$

AVL Tree (Insertion)



AVL Tree (Insertion)

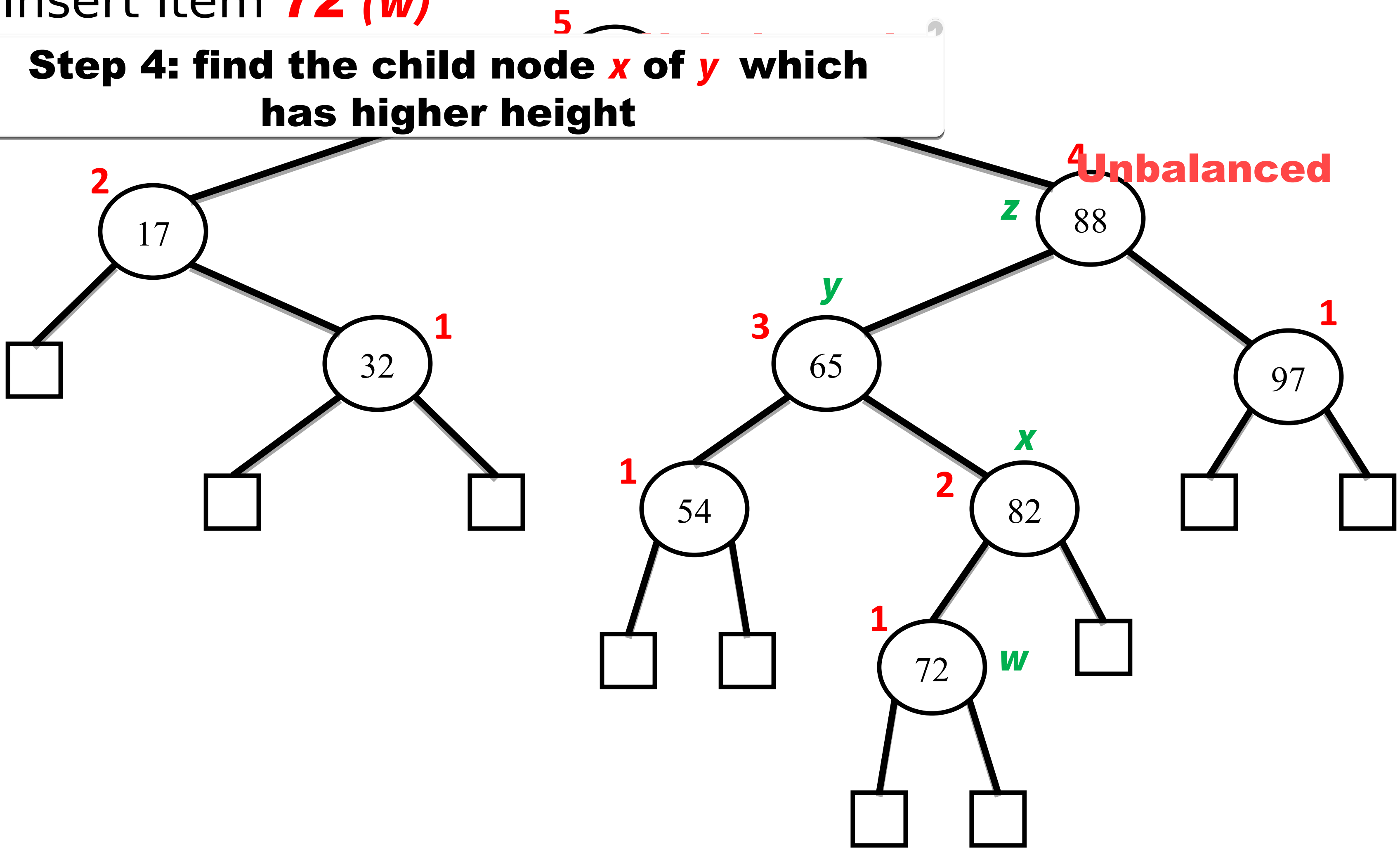
Insert Item **72**



AVL Tree (Insertion)

Insert Item **72** ***(w)***

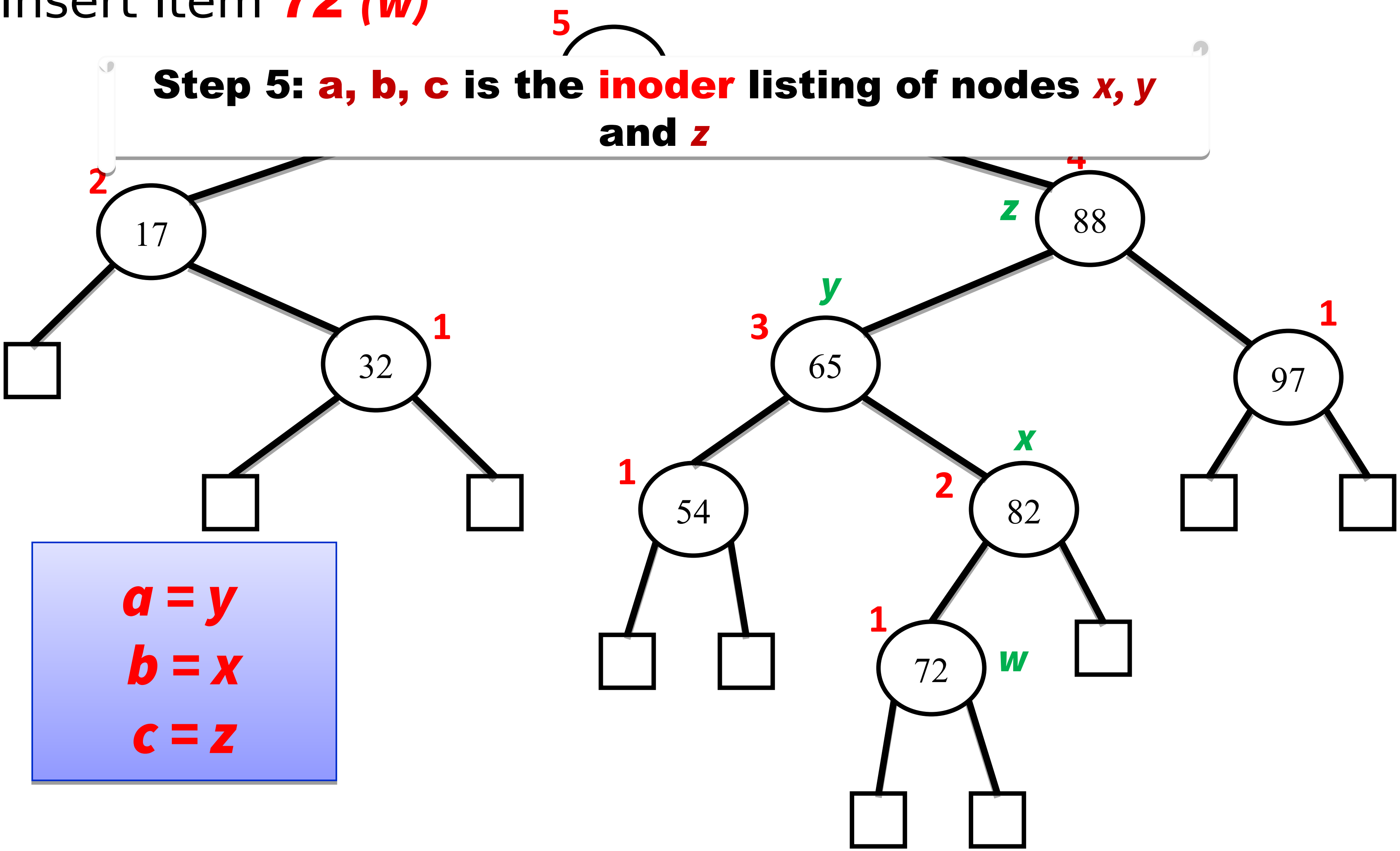
Step 4: find the child node *x* of *y* which has higher height



AVL Tree (Insertion)

Insert Item **72 *(w)***

Step 5: a, b, c is the inoder listing of nodes *x, y* and *z*



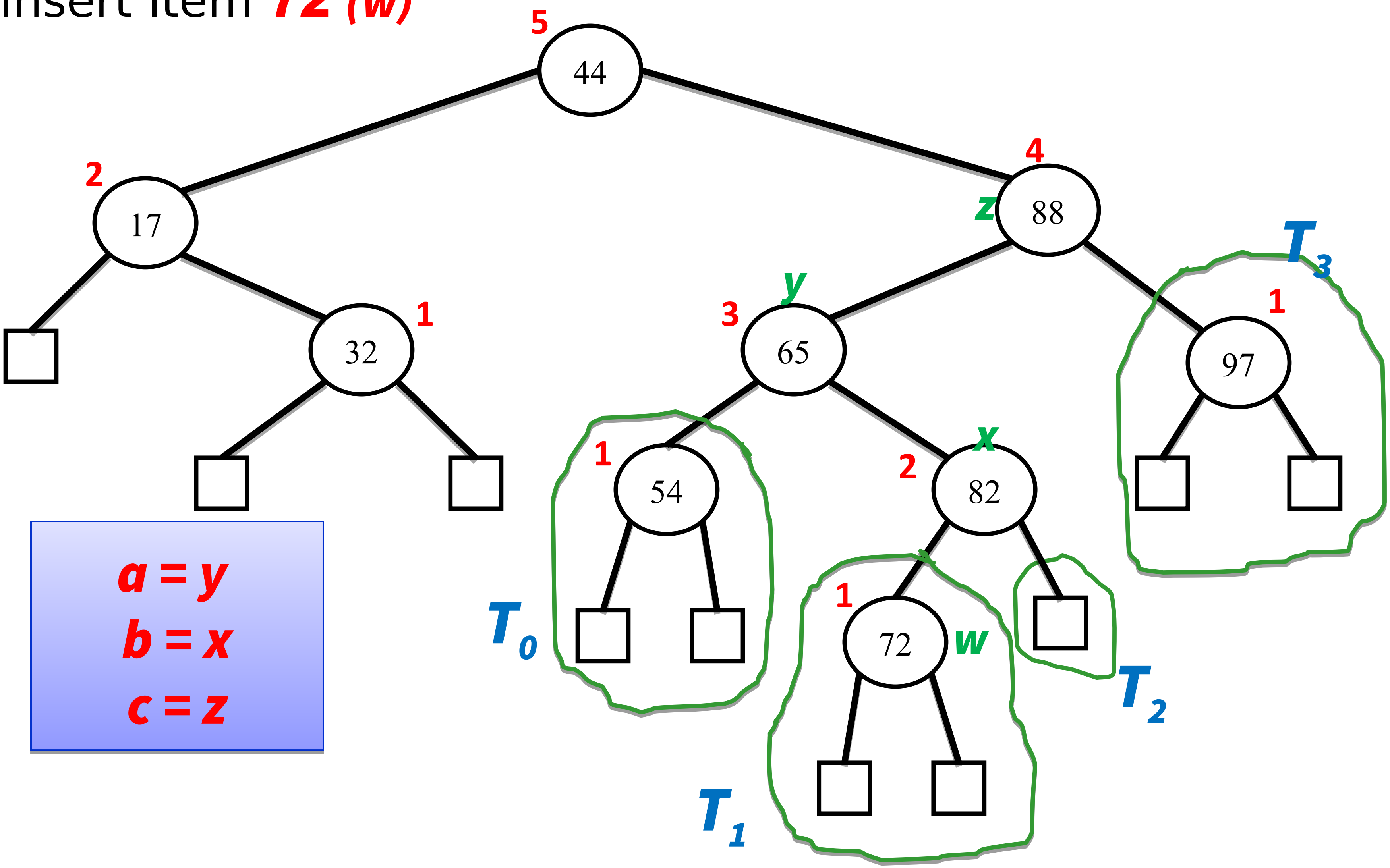
a = y

b = x

c = z

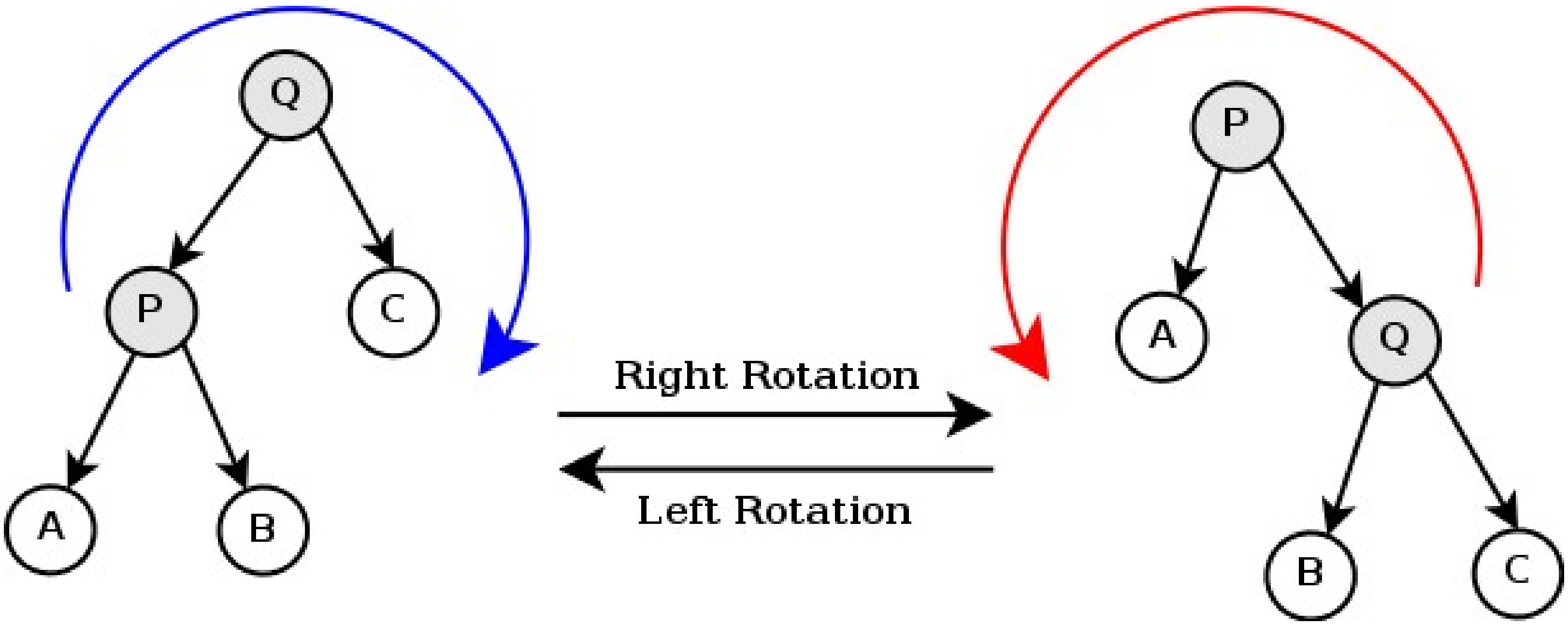
AVL Tree (Insertion)

Insert Item **72** ***(w)***

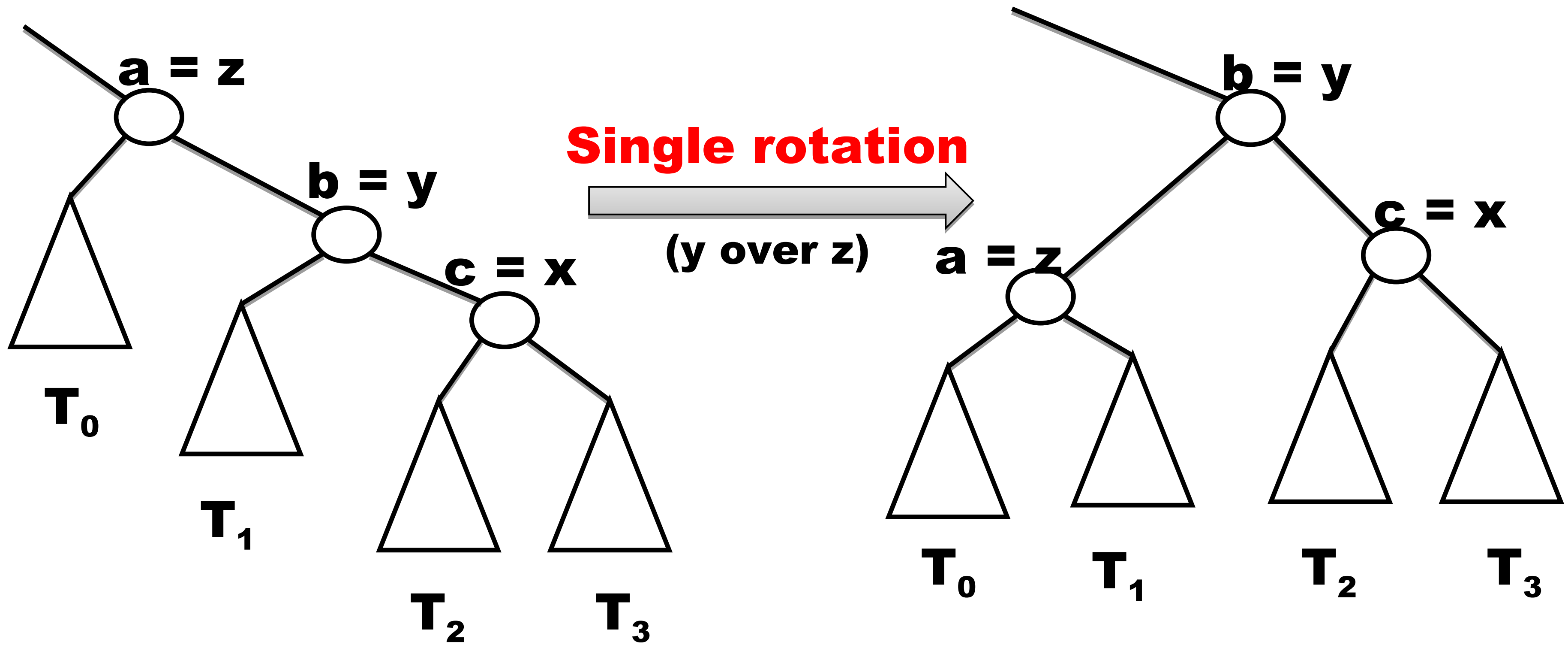


a = y
b = x
c = z

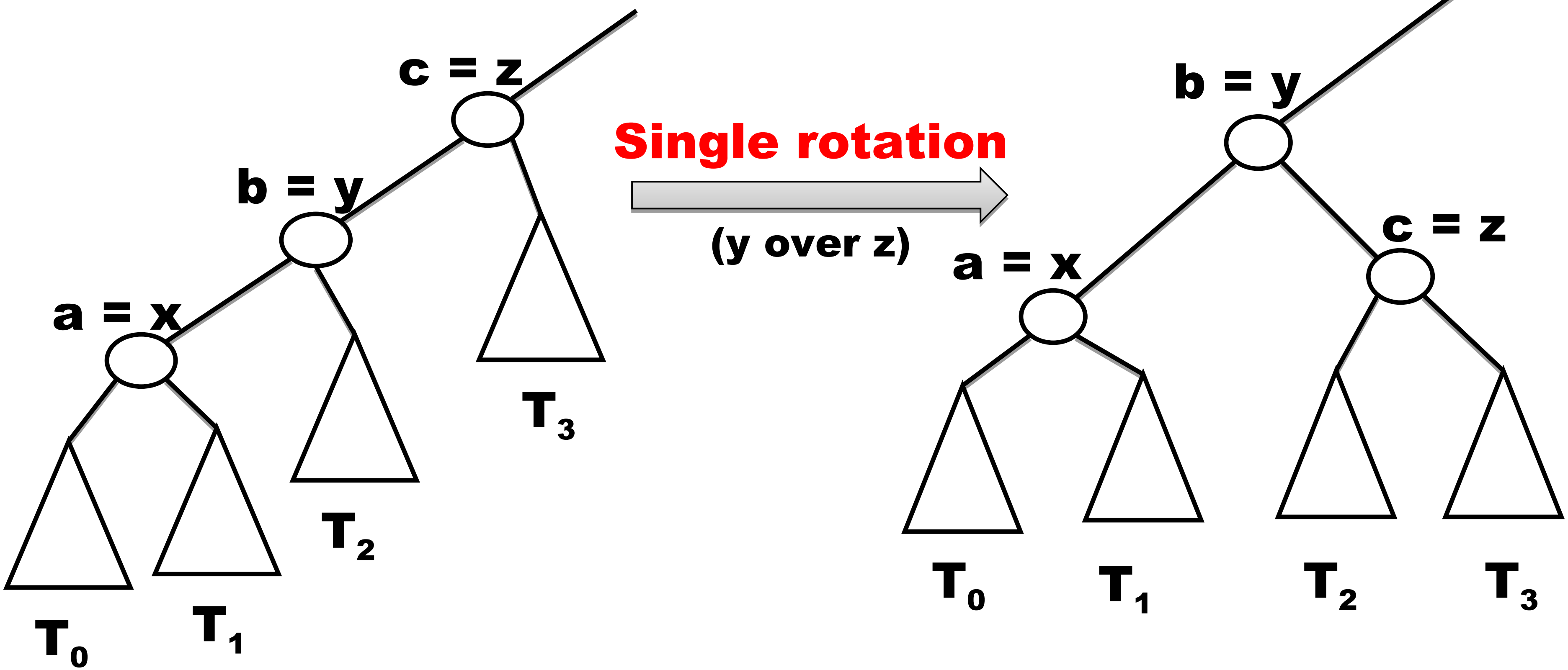
AVL Tree (Rotation)



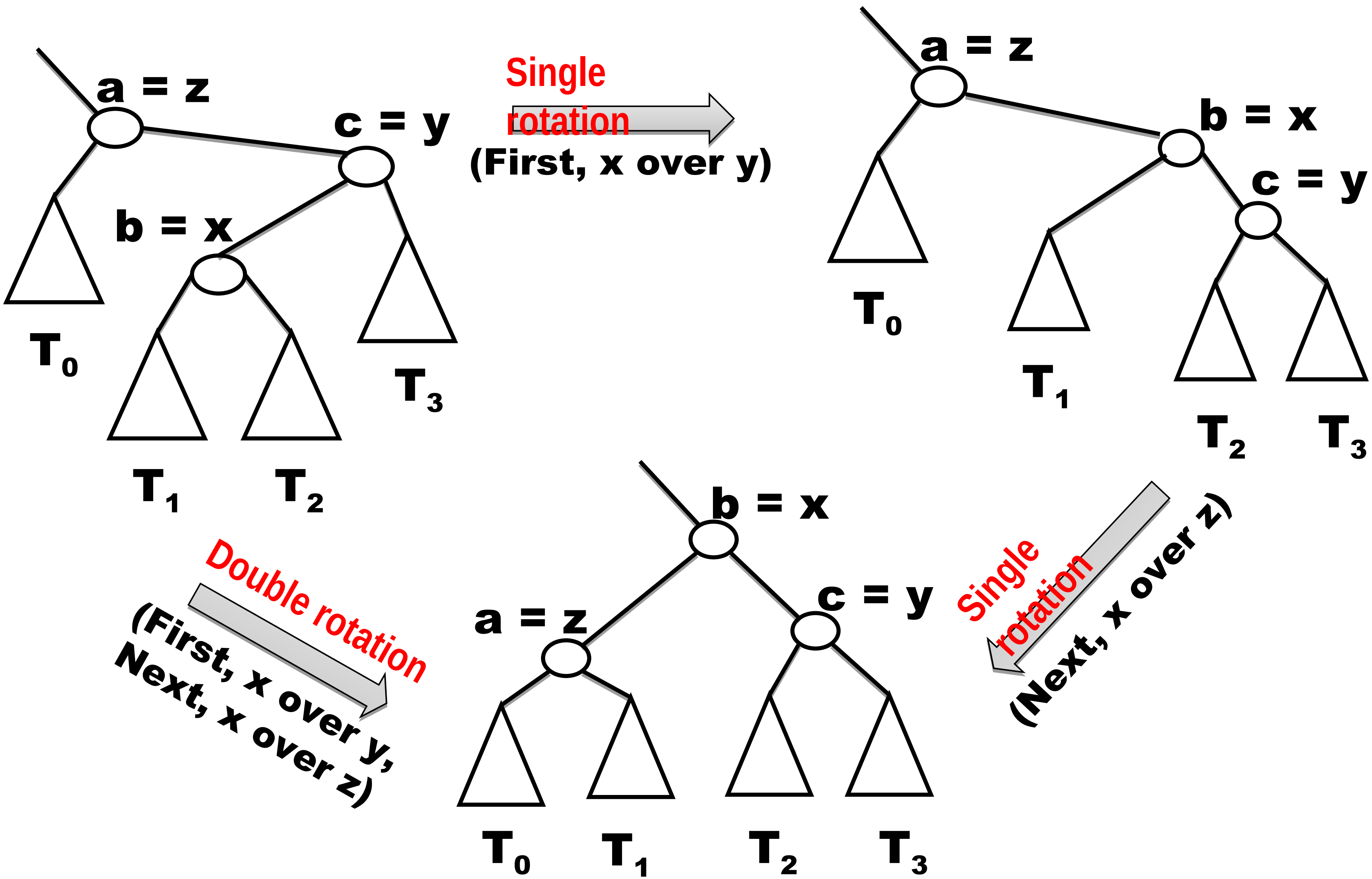
AVL Tree (Rotation)



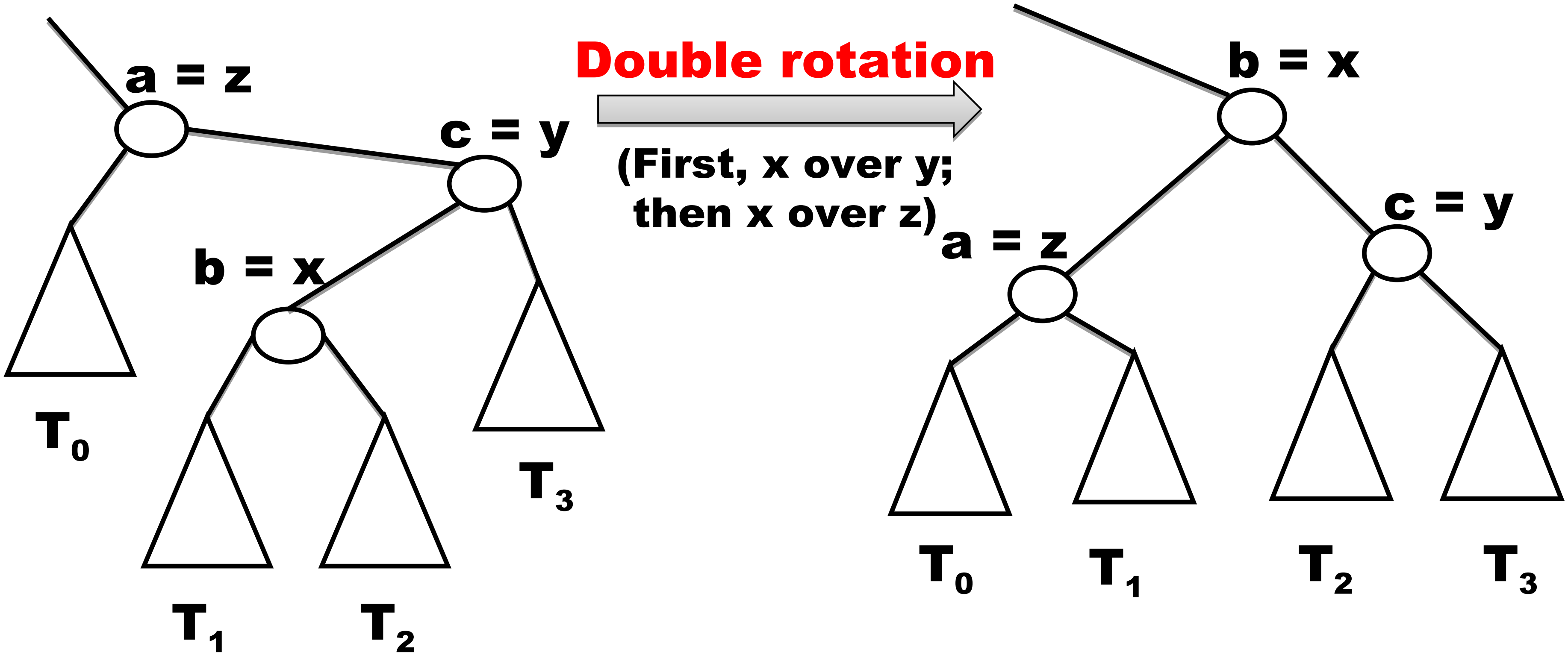
AVL Tree (Rotation)



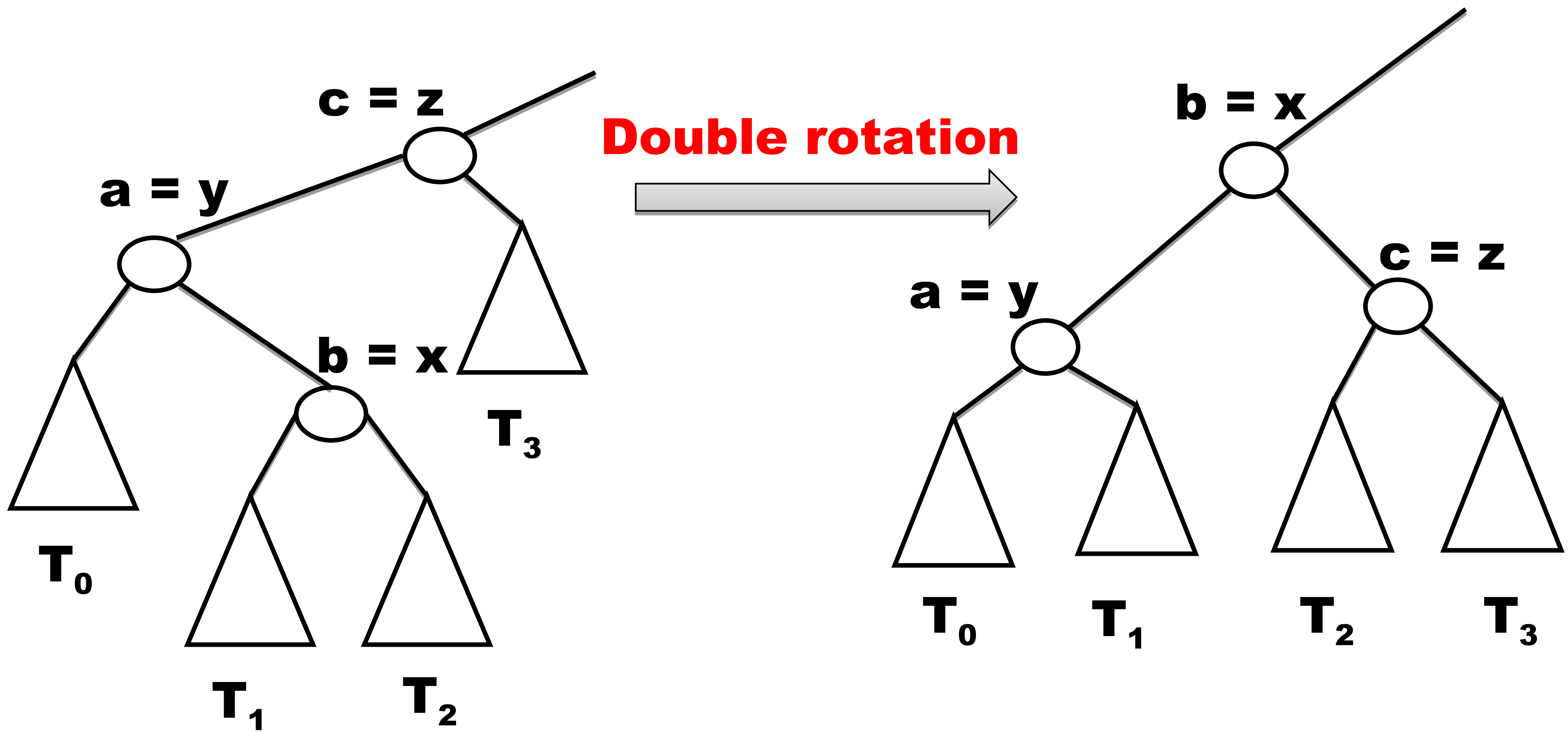
AVL Tree (Rotation)



AVL Tree (Rotation)

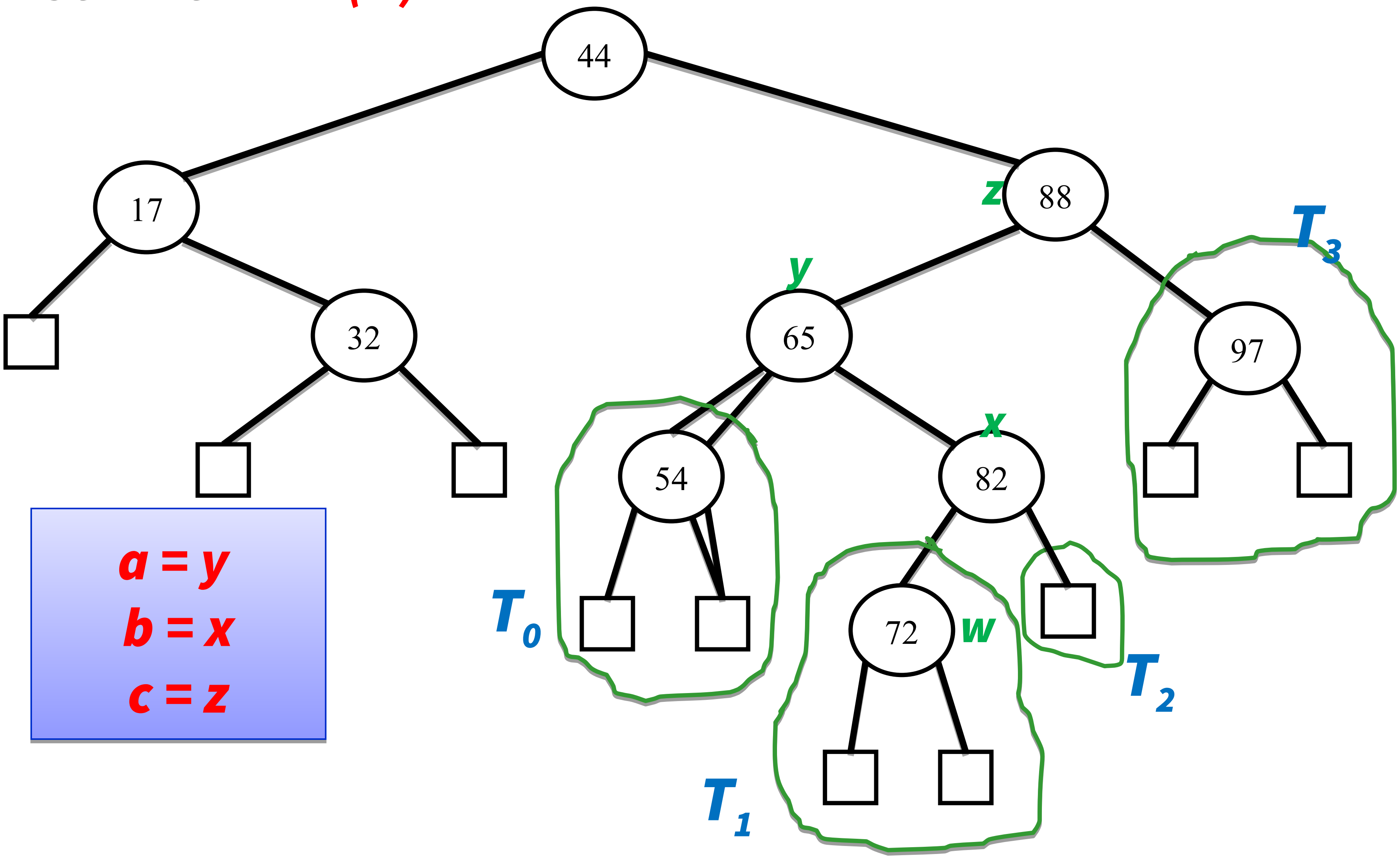


AVL Tree (Rotation)



AVL Tree (Insertion)

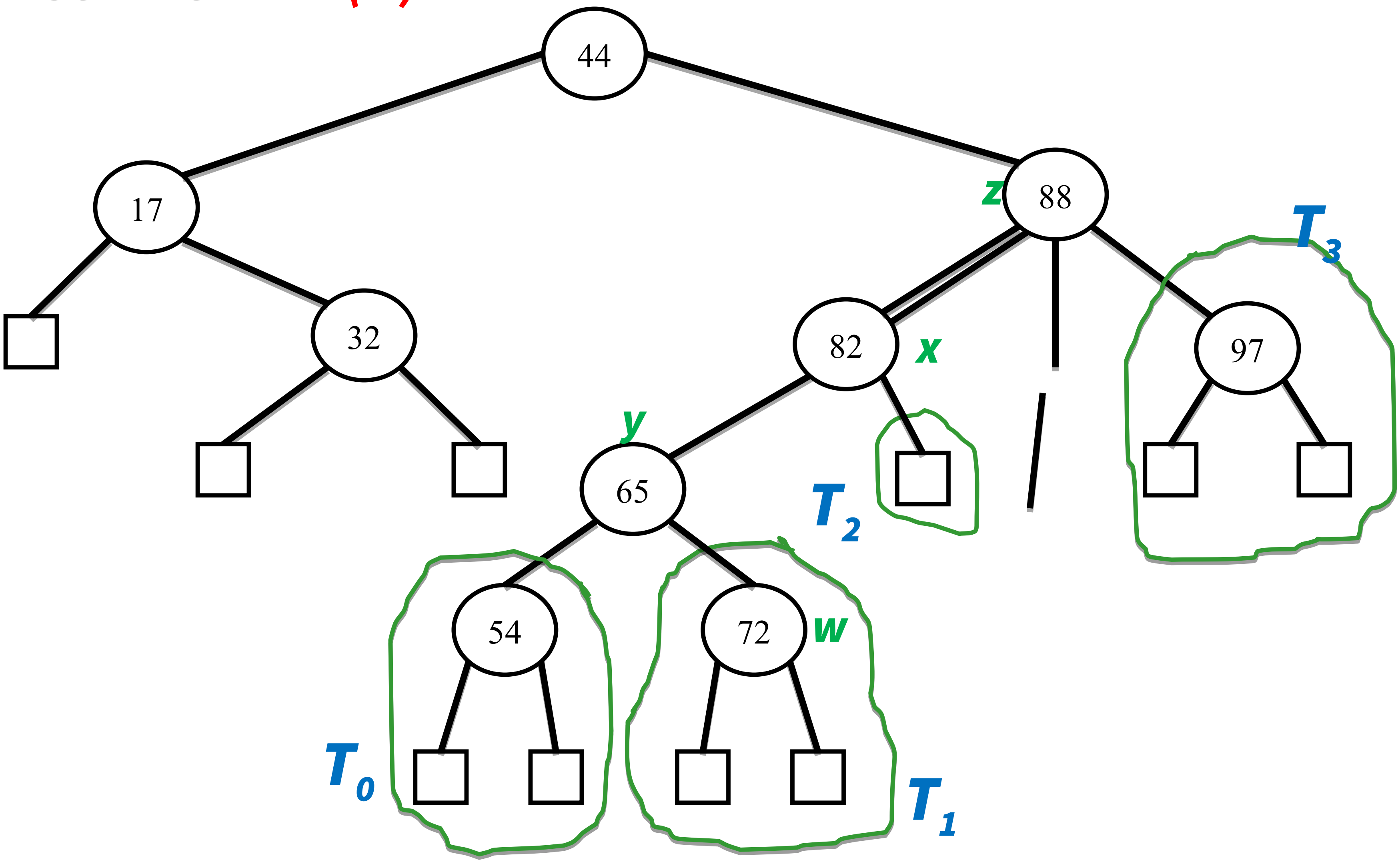
Insert Item **72** ***(w)***



a = y
b = x
c = z

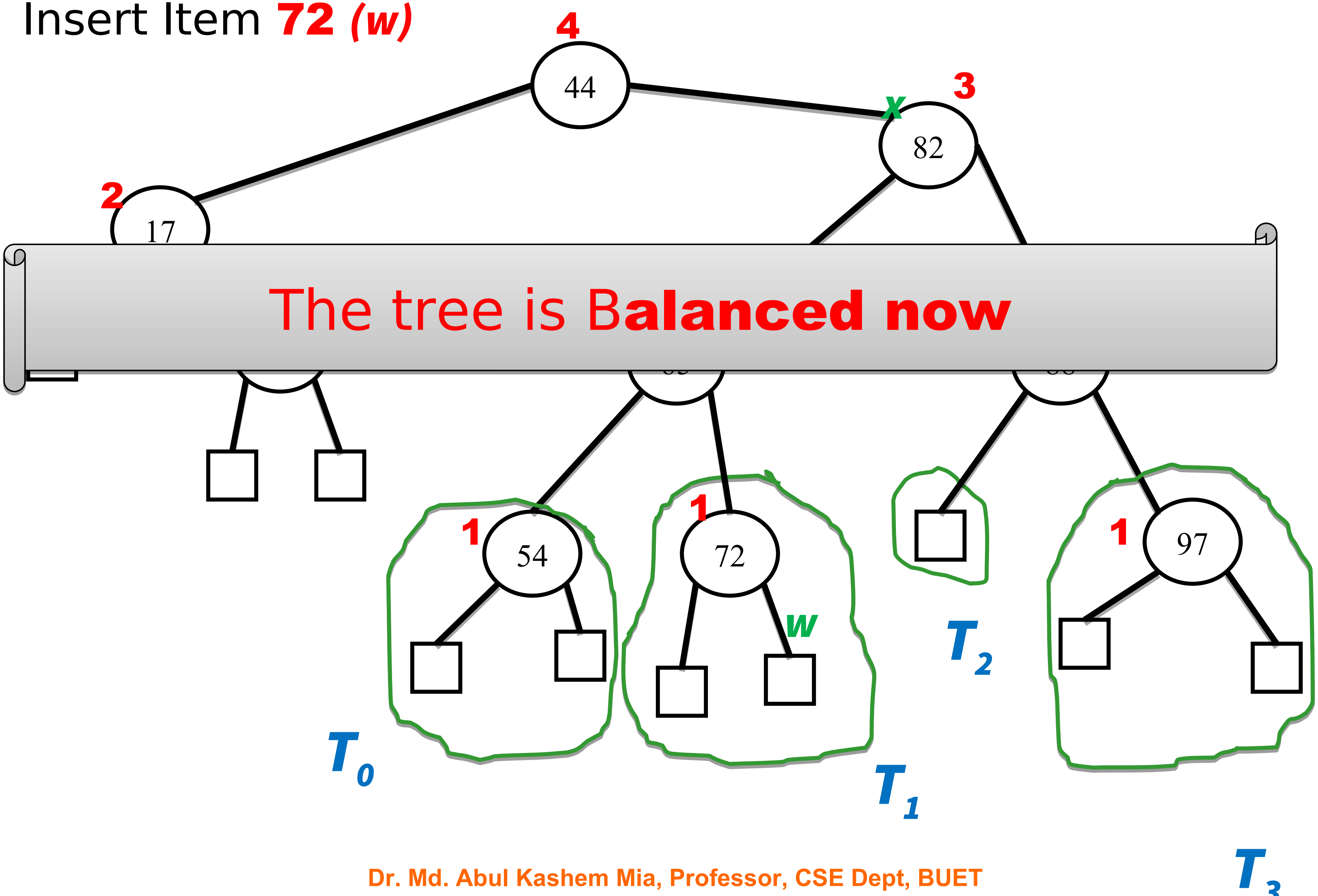
AVL Tree (Insertion)

Insert Item **72** ***(w)***



AVL Tree (Insertion)

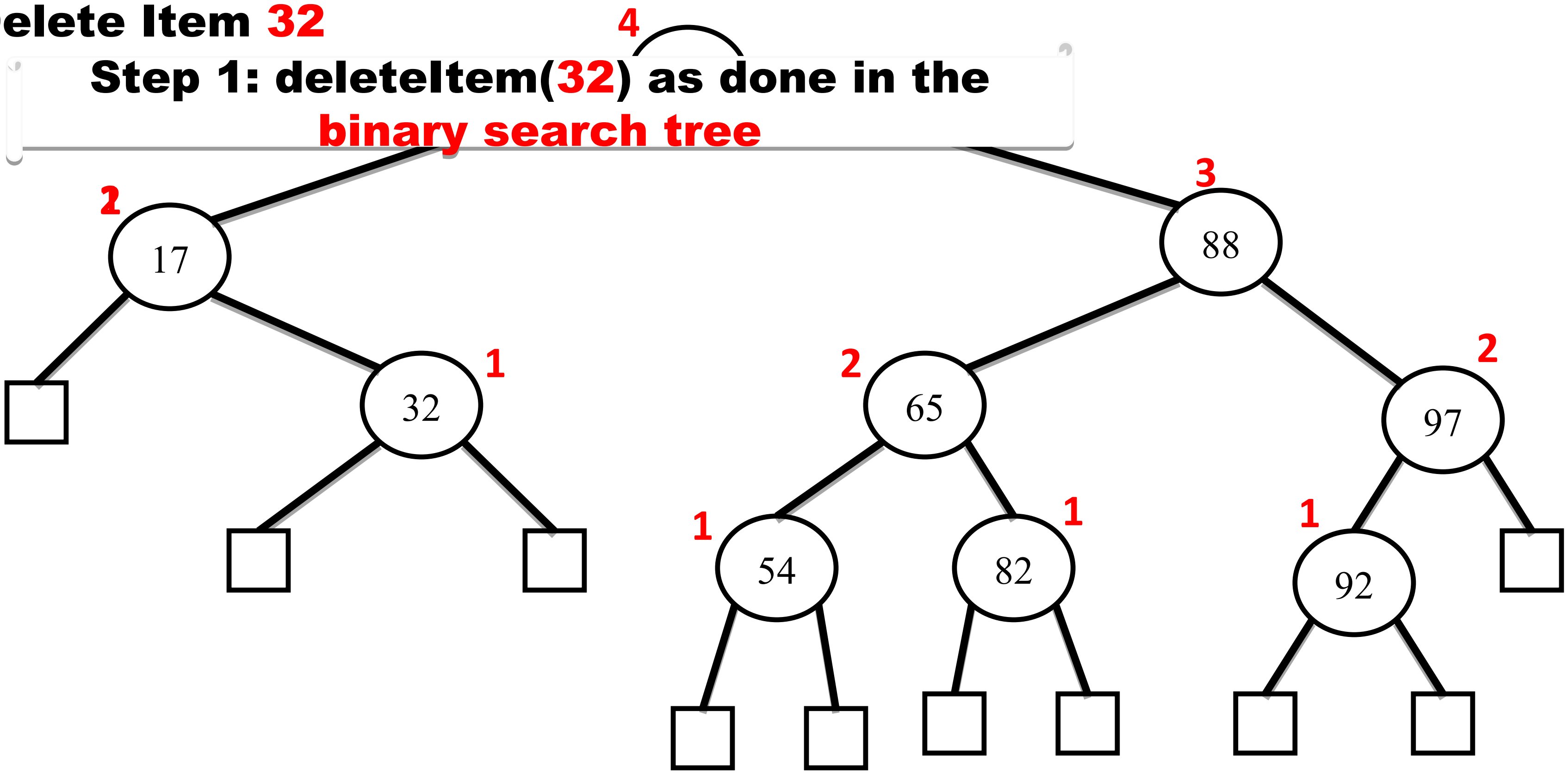
Insert Item **72** ***(w)***



AVL Tree (Deletion)

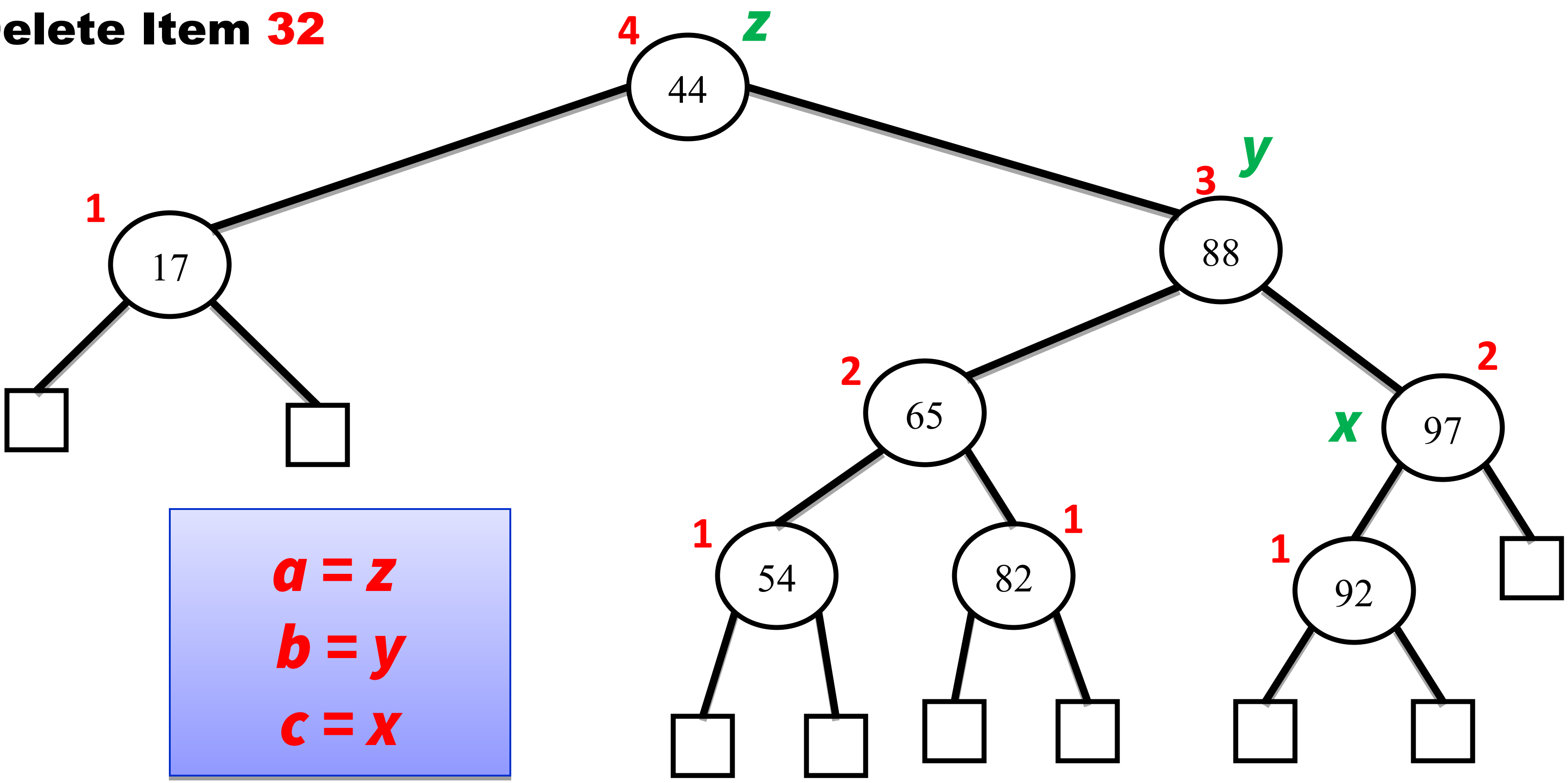
Delete Item 32

Step 1: deleteItem(32) as done in the binary search tree



AVL Tree (Deletion)

Delete Item 32



$a = z$
$b = y$
$c = x$

After a single rotation

AVL Tree (Deletion)