COL106
Data Structures and Algorithms

Subodh Sharma and Rahul Garg

Height Balanced Binary Search Trees

AVL Trees

Height Balanced Binary Trees

- Ensure that the left and right subtrees are always "balanced"
- Height balanced trees
- For all u, | height(u->left) – height(u->right) | ≤ 1

AVL Trees: Examples

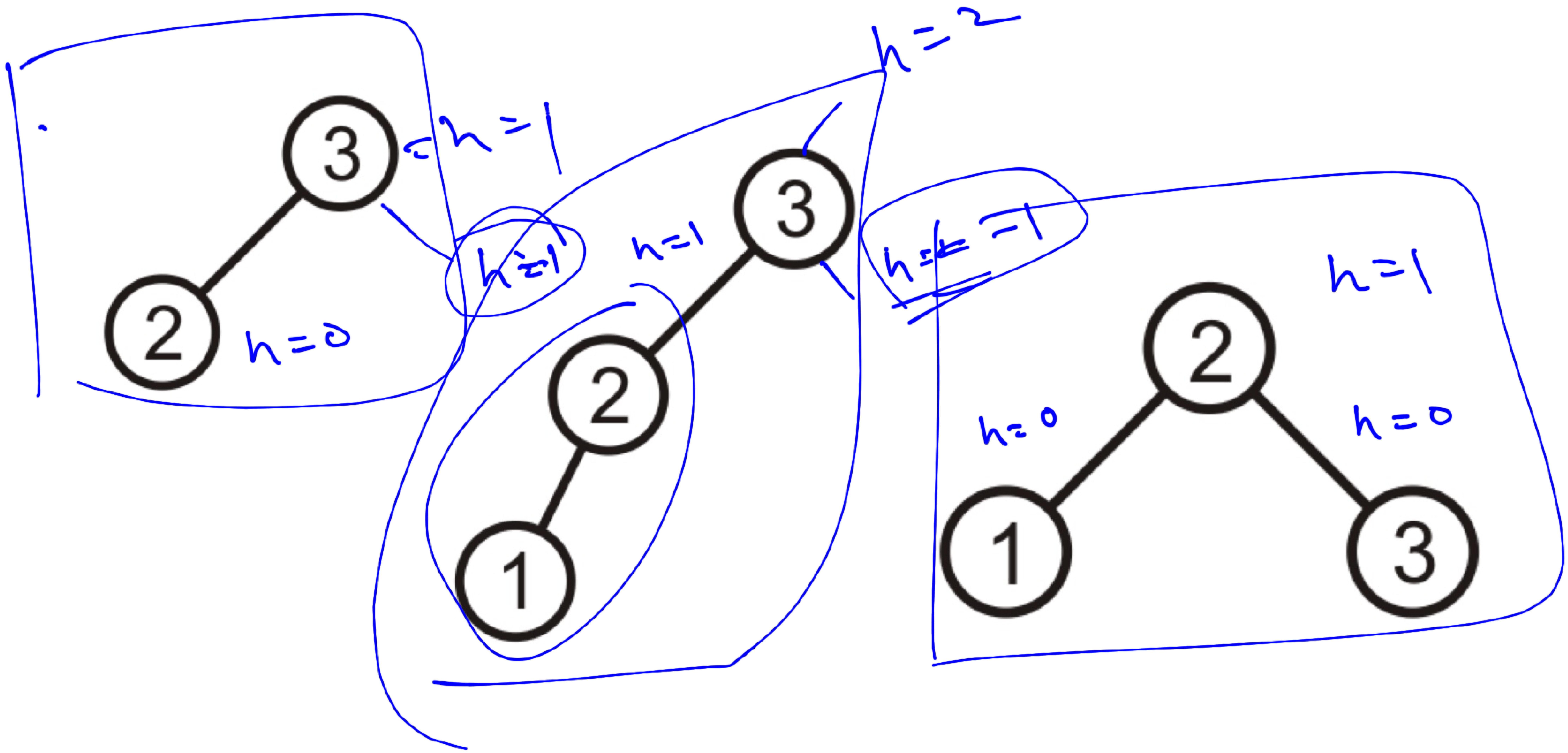


Figure courtesy: Douglas Wilhelm Harder, University of Waterloo, Canada

AVL Trees: Examples

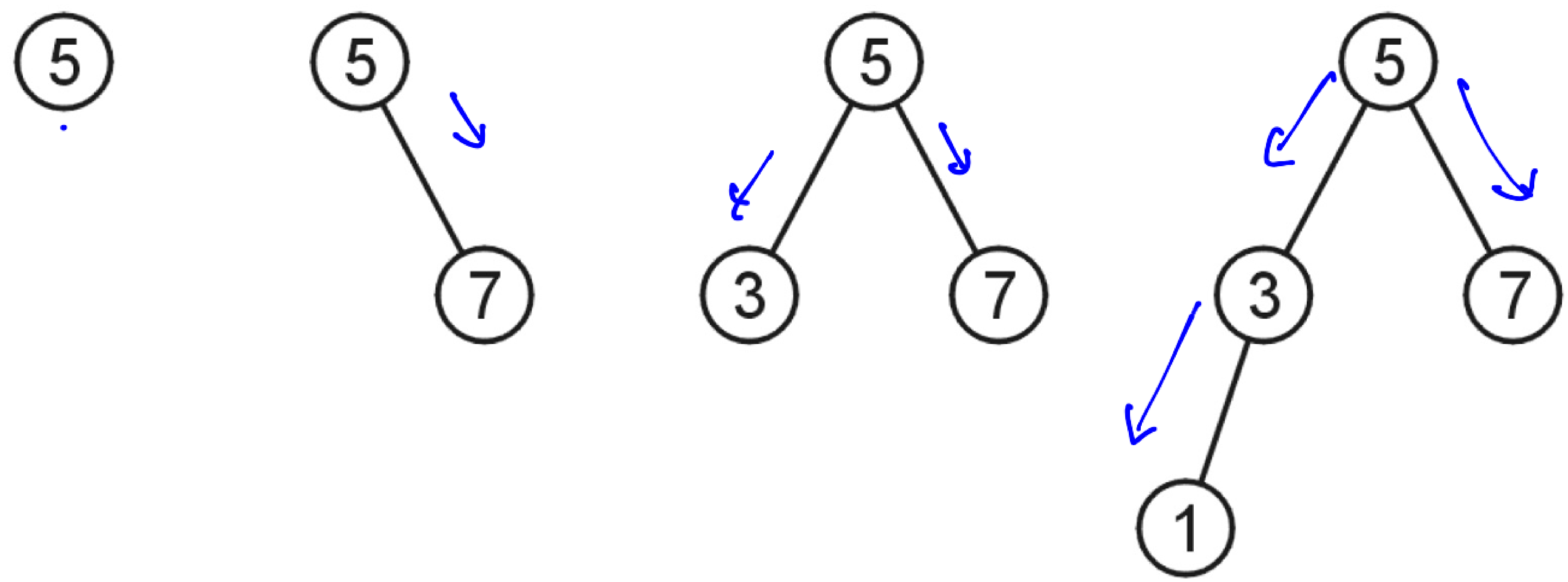


Figure courtesy: Douglas Wilhelm Harder, University of Waterloo, Canada

AVL Trees: Examples

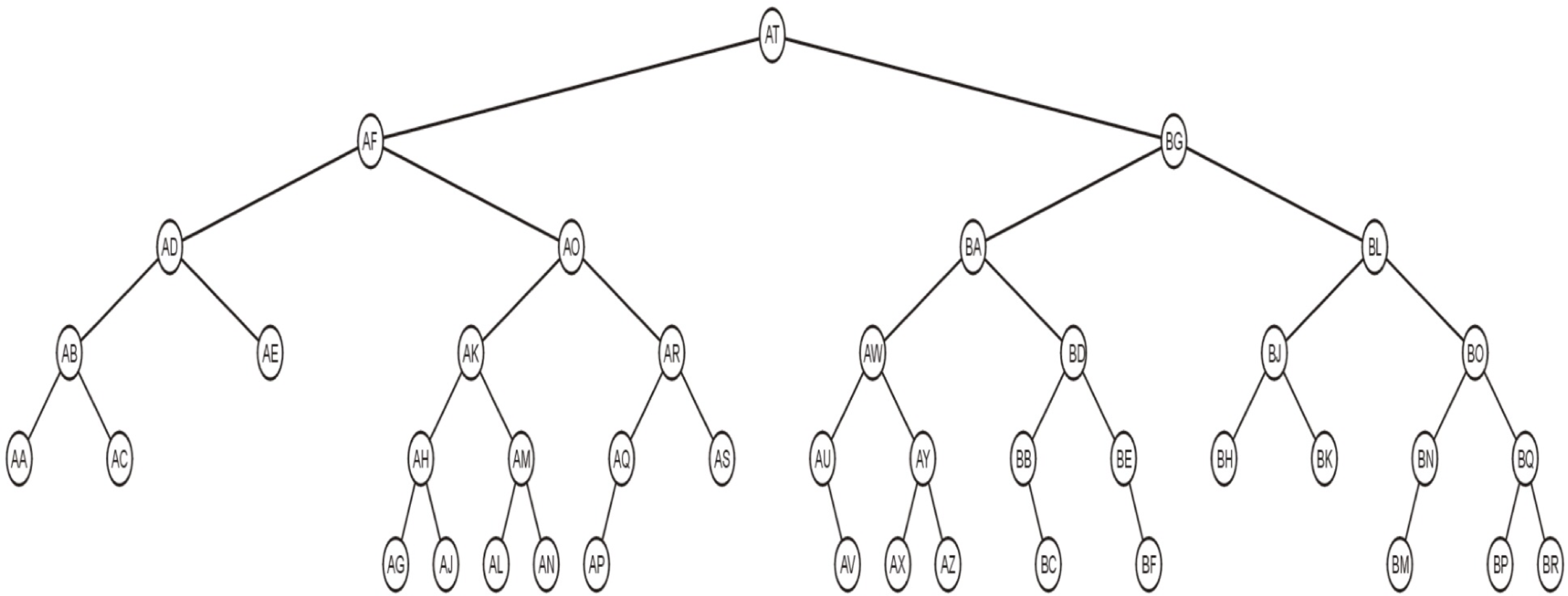


Figure courtesy: Douglas Wilhelm Harder, University of Waterloo, Canada

AVL Trees: Examples

The root node is height balanced

- Left subtree height is 4
- Right subtree height is also 4

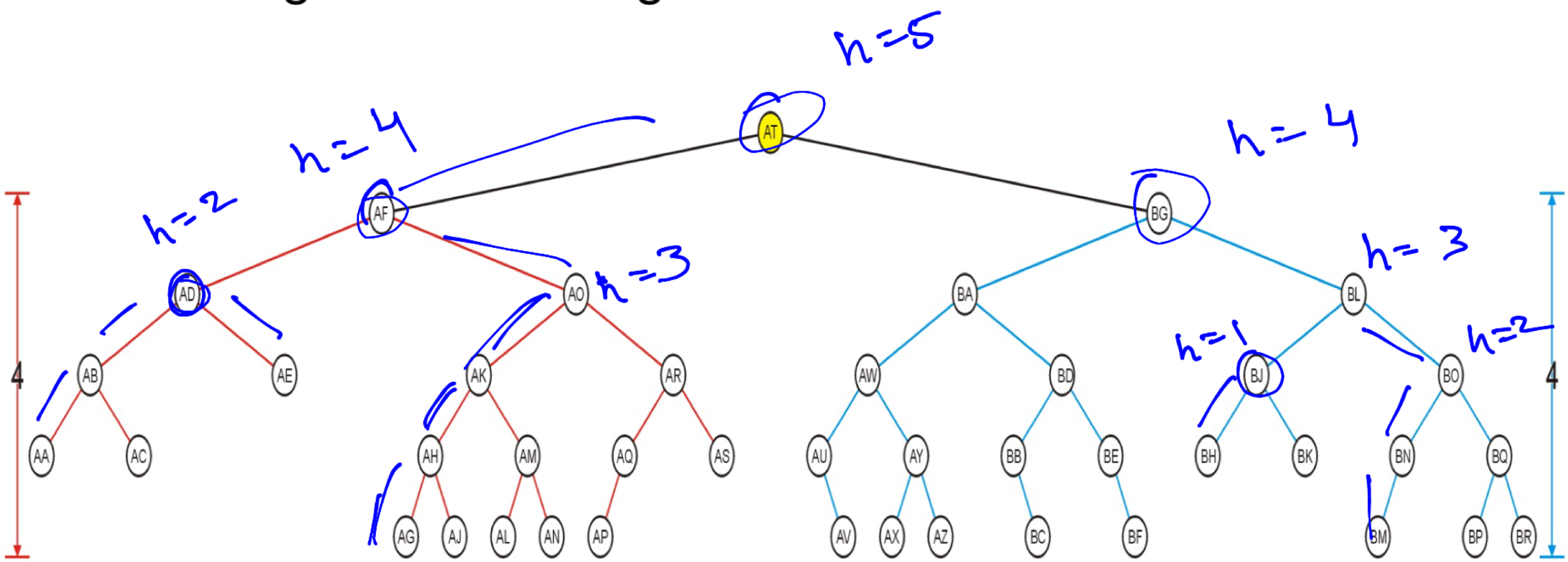


Figure courtesy: Douglas Wilhelm Harder, University of Waterloo, Canada

AVL Trees: Examples

- Verify for other nodes also

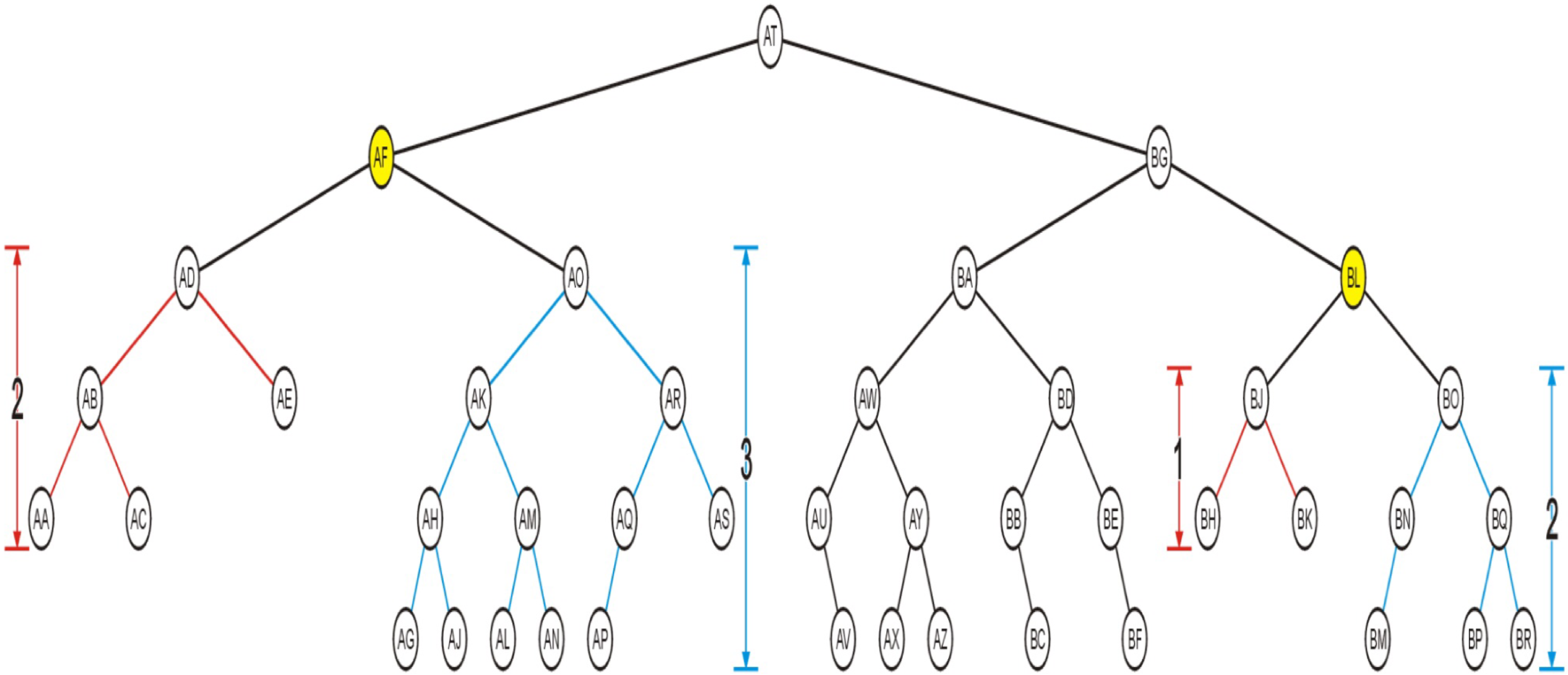


Figure courtesy: Douglas Wilhelm Harder, University of Waterloo, Canada

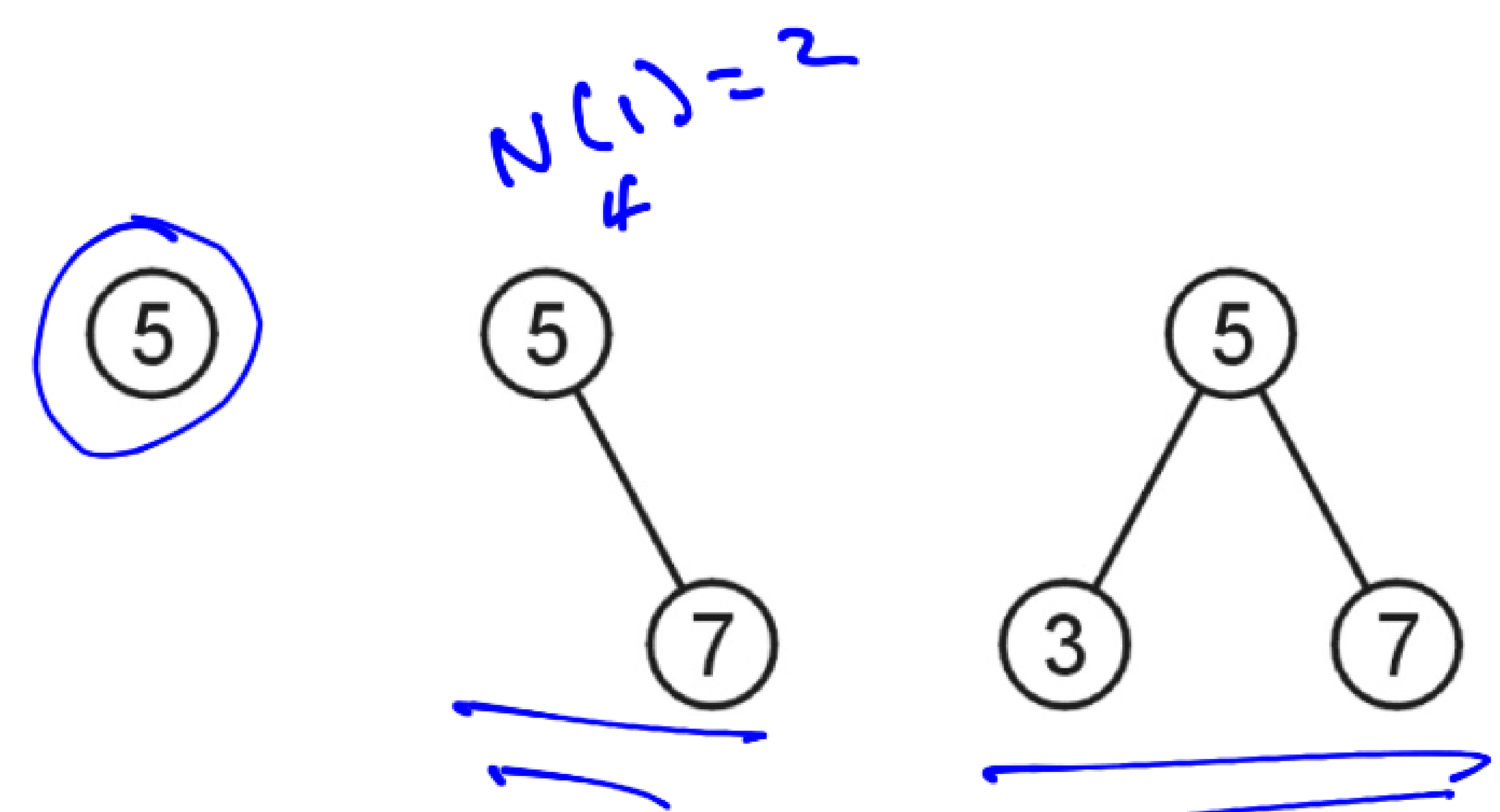
Bounding the Height of AVL Trees

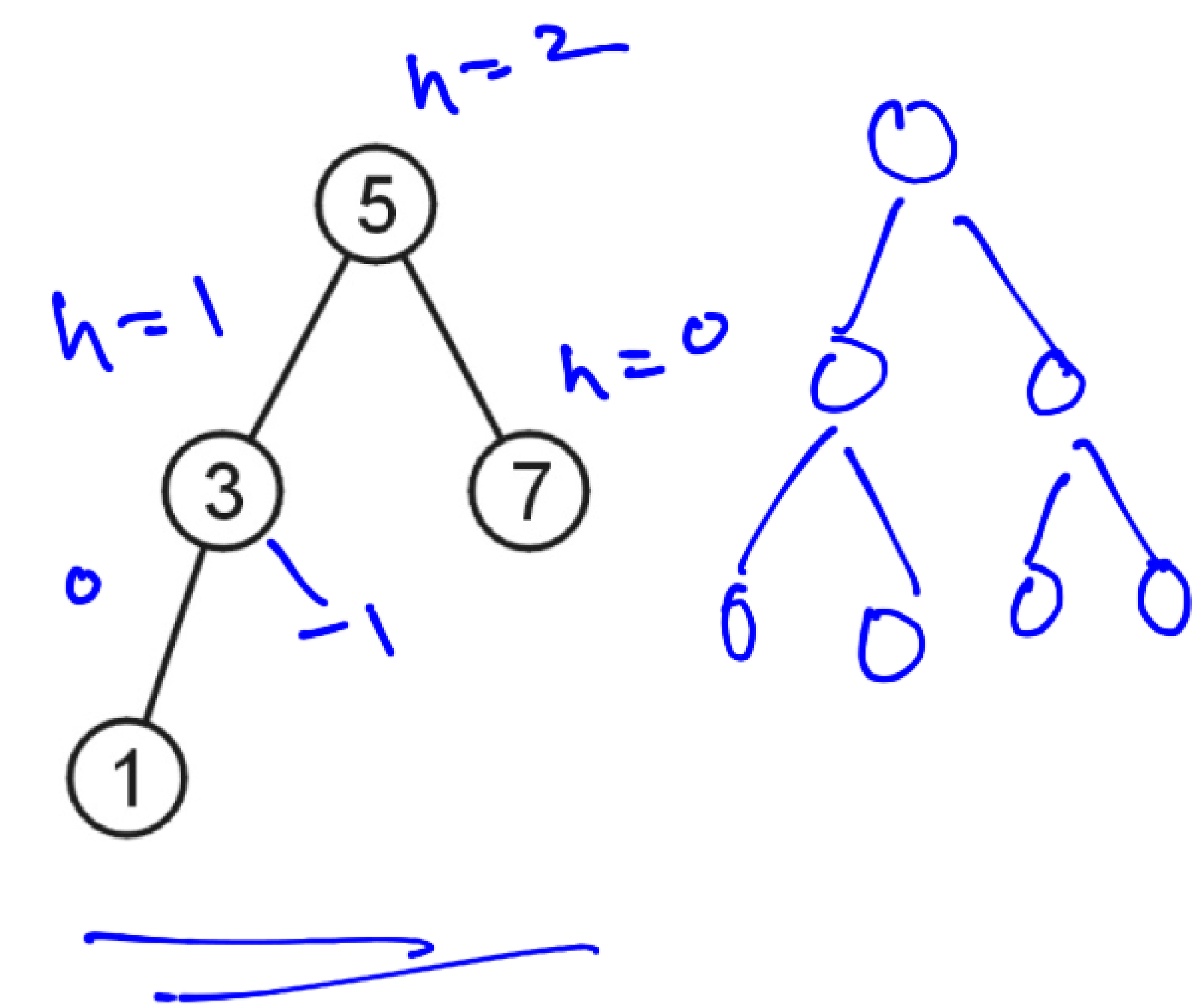
- Prove that the maximum height of a balanced (height or weight) tree with N nodes will be O(log(N))

Bounding the Height of AVL Trees

Let $N(h)$ be the smallest number of nodes in a tree of height h

$N(0) = 1$
$N(1) = 2$
$N(2) = 4$





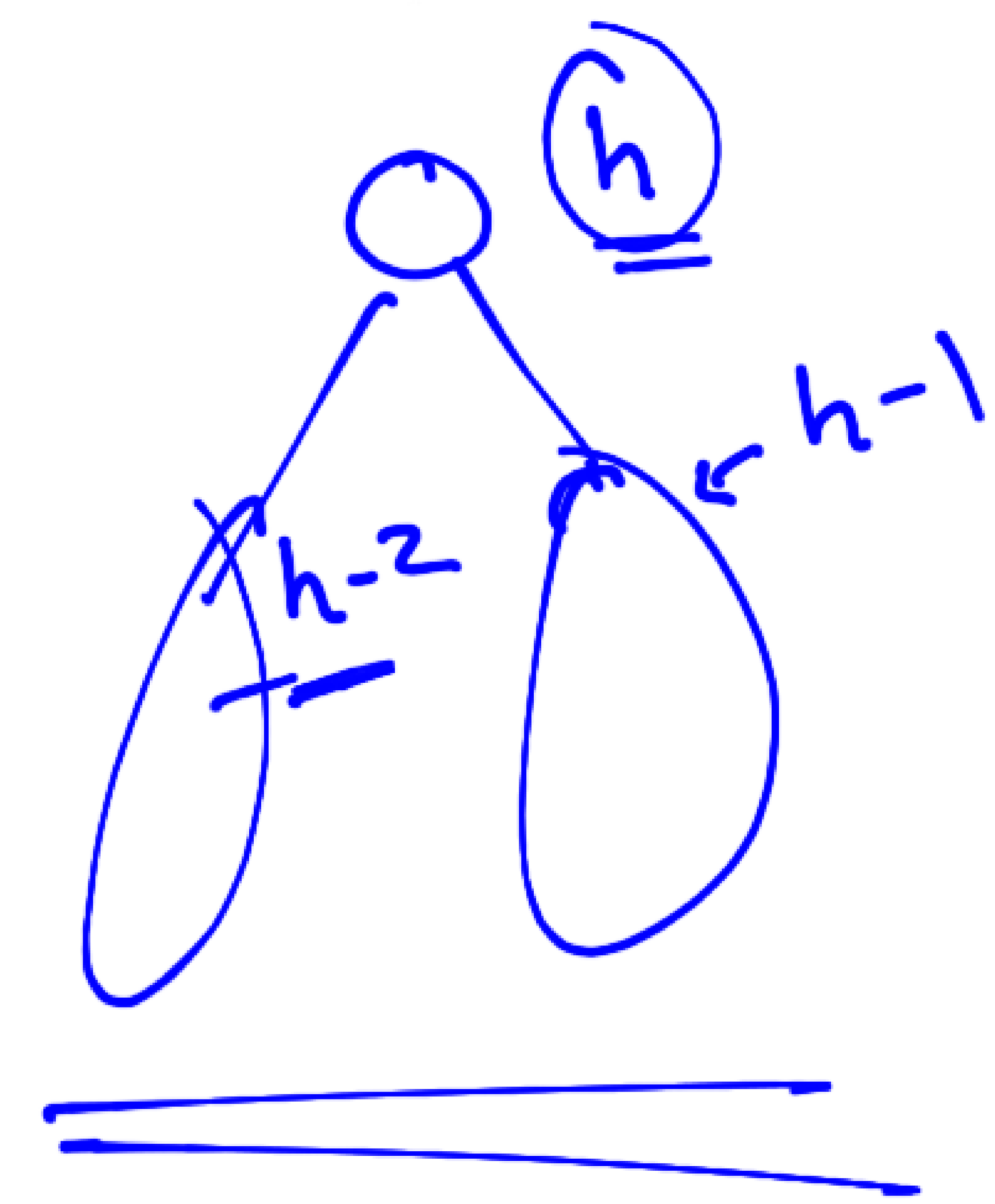
What will be a suitable recurrence for $N(h)$?

Bounding the Height of AVL Trees

The smallest AVL tree of height h would have:

- A smallest AVL tree of height $h-1$ on one side,
- A smallest AVL tree of height $h-2$ on the other, and
- The root node

We get: $N(h) = N(h-1) + 1 + N(h-2)$



Bounding the Height of AVL Trees

This is a recurrence relation:

$$N(h) = \begin{cases} 1 & h = 0 \\ 2 & h = 1 \\ N(h-1) + N(h-2) + 1 & h > 1 \end{cases}$$

Solution?

Height of an AVL Tree

- **Theorem**: The ***height*** of an AVL tree with n nodes is O(log n).
- **Proof**: Let **n(h)** represent the minimum number of ~~internal~~ nodes of an AVL tree of height h. We will obtain a lower bound for n(h).
- We have: n(0) = 1 and n(1) = 2
- For h > 1, an AVL tree of height h contains the root node, one AVL subtree of height h-1 and another of height h-2.
- That is, n(h) = 1 + n(h-1) + n(h-2)
- Also since, n(h-1) > n(h-2), we get n(h) > 2n(h-2). So
  - n(h) > 2n(h-2), n(h) > 4n(h-4), n(h) > 8n(n-6), … (by induction),
  - n(h) > 2^in(h-2i)
- Solving the base case we get: n(h) > $2^{h/2-1}$

$n(h) > \frac{1}{2}(\sqrt{2})^h$

- Taking logarithms: h < 2log n(h) +2
- Thus the height of an AVL tree is O(log n)

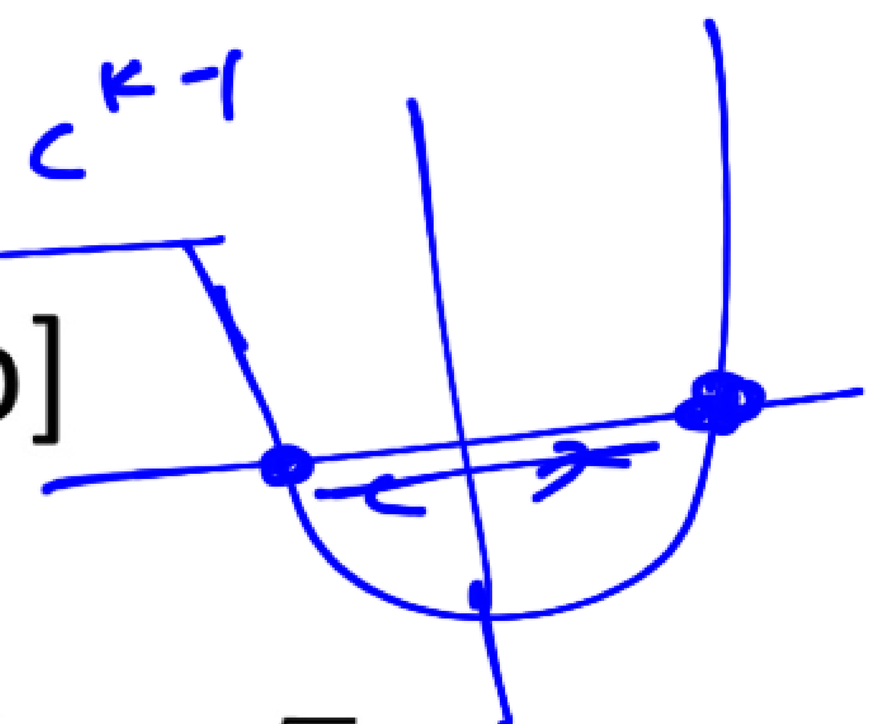
Slide adapted from Shweta and Amit, Department of CSE, IIT Delhi

A Tighter Bound

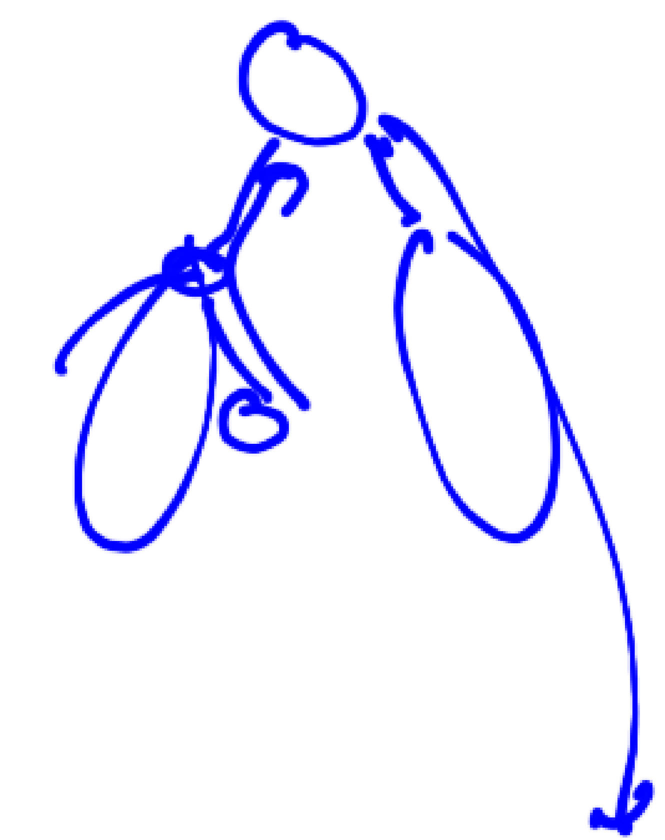
- Can we obtain a tighter bound?
- $n(h) = 1 + n(h-1) + n(h-2)$; $n(0) = 1$; $n(1) = 2$;
- Use induction to prove that $n(h) \geq c^{h-1}$ for some constant $c > 1$.
- Base case: $h = 1 \rightarrow n(h) = 2 > 1 = c^{h-1}$ $= c^{1-1} = c^0 = 1$
- Induction hypothesis: Assume $n(h) \geq c^{h-1}$ for all $h < k$
- Induction step: Show that $n(k) \geq c^{k-1}$
- $n(k) = 1 + n(k-1) + n(k-2)$
- $n(k) \geq 1 + c^{k-2} + c^{k-3}$

A Tighter Bound

- From induction hypothesis, $n(k) \geq 1 + c^{k-2} + c^{k-3}$
- If we can find a $c > 1$ such that, $n(k) \geq 1 + c^{k-1}$
- $c^{k-2} + c^{k-3} \geq c^{k-1}$ then we can say that $n(k) \geq c^{k-1}$
- $n(k) \geq c^{k-1}$ [thereby proving the induction step]
- Such a c must satisfy: $\boxed{c^2 - c - 1 \leq 0}$
- The above quadratic equation has roots $\frac{1-\sqrt{5}}{2}, \frac{1+\sqrt{5}}{2}$
- Any solution between these roots will satisfy the induction step
- We take $c = \frac{1+\sqrt{5}}{2} \cong 1.62$
- Thus $n(h) \geq 1.62^{h-1}$
- $h \leq 1 + \log_{1.62}(n)$

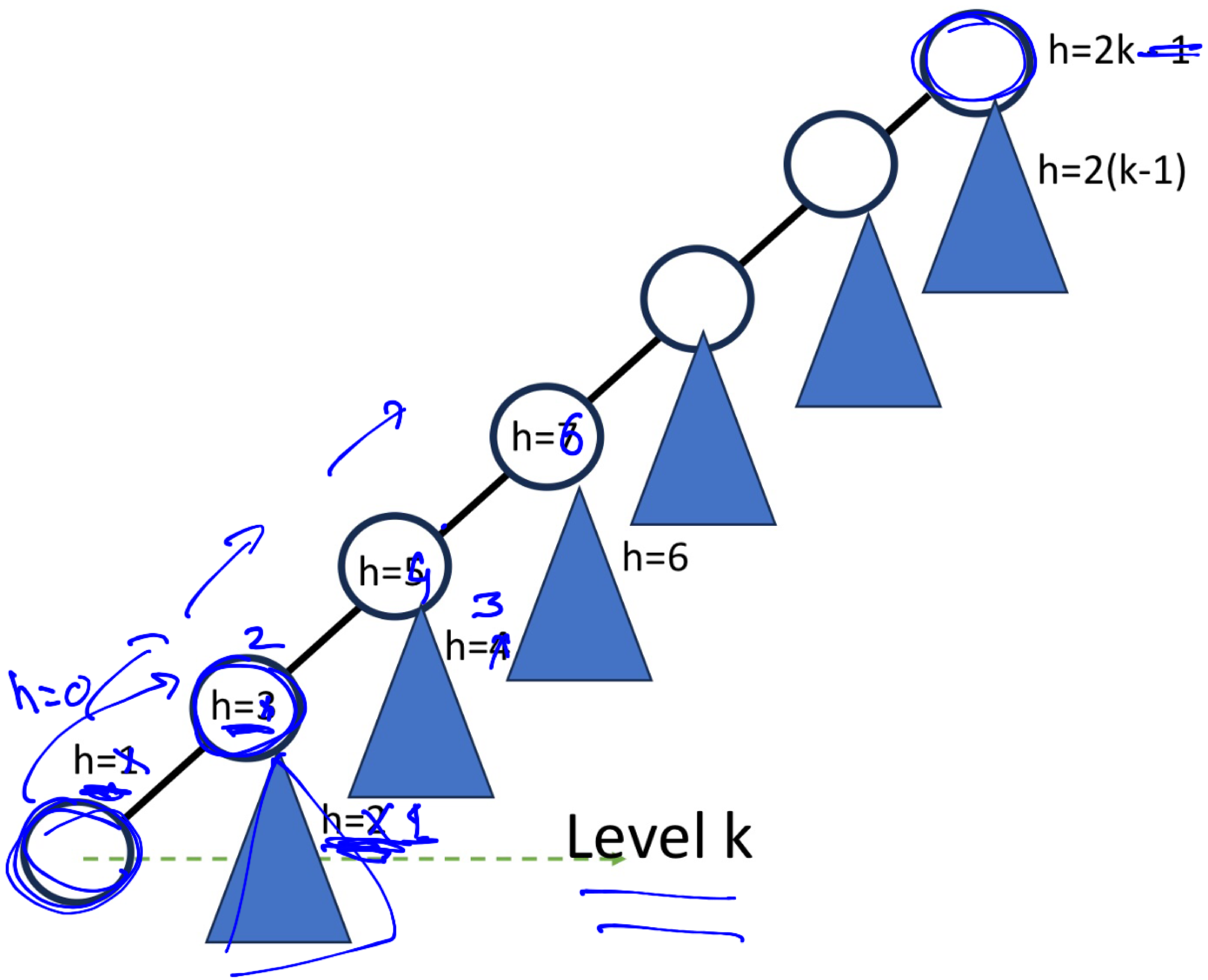


Structure of AVL Trees



- An AVL tree with n nodes
- Min height: ceiling [$\log_2(n+1) - 1$]
- Max height: ceiling [$1 + \log_{1.62}(n)$]
- Consider a leaf which is closest to the root
- Suppose this leaf is at level k
- We will show that the maximum height of the tree is at most 2k-1

Structure of AVL Trees



Structure of AVL Trees

- By an inductive argument along the path from the root to the closest leaf node in the AVL tree, it may be proved that the maximum height of the tree is at most 2k~~-1~~

Maintaining a Height Balance

- Insert may increase the height of a subtree by 1
- Remove may decrease the height of a subtree by 1
- The BST may no longer be height balanced after these operations

Maintaining Balance

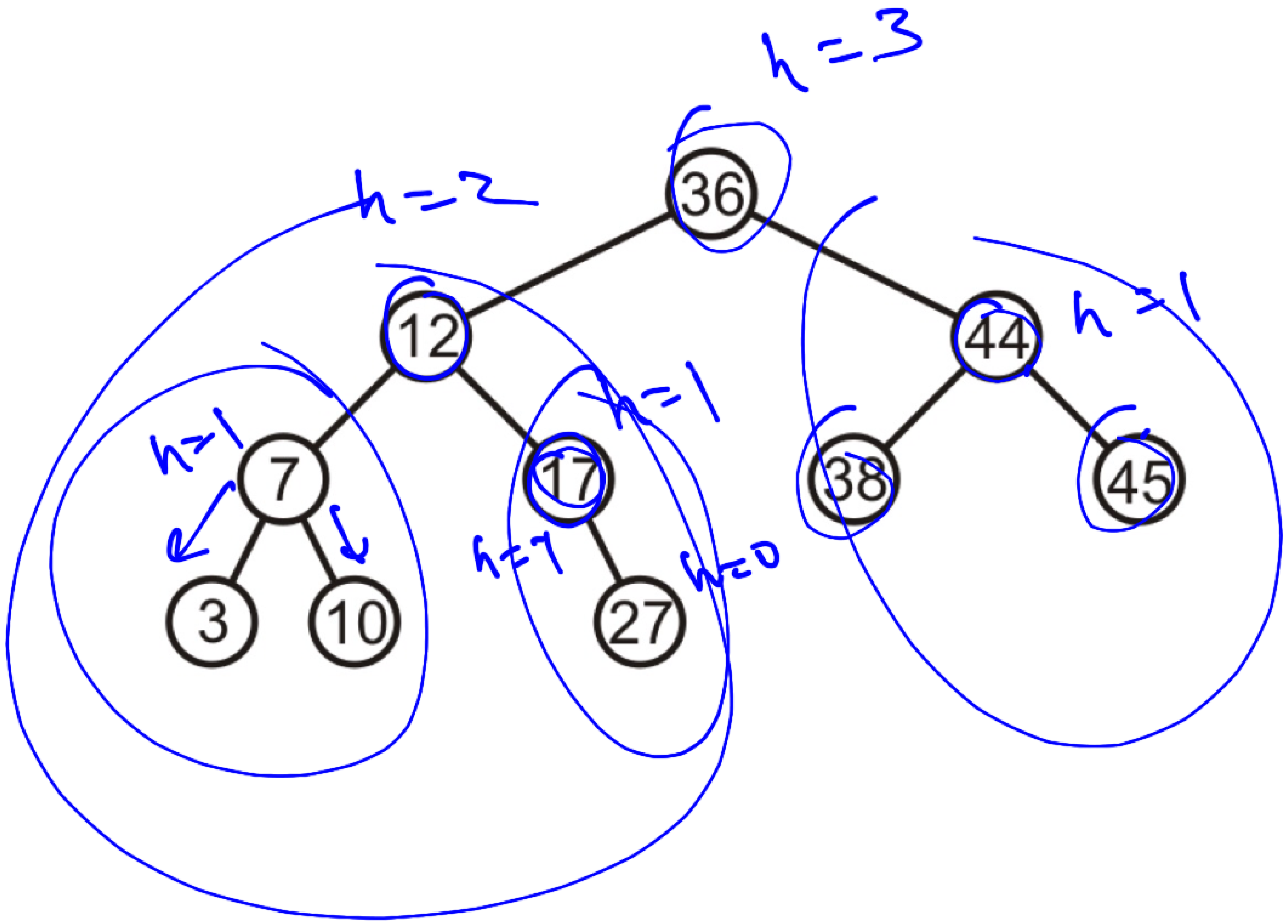


Image source: Douglas Wilhelm Harder, University of Waterloo, Canada

Maintaining Balance

Insert 15

How do the heights of subtrees change?

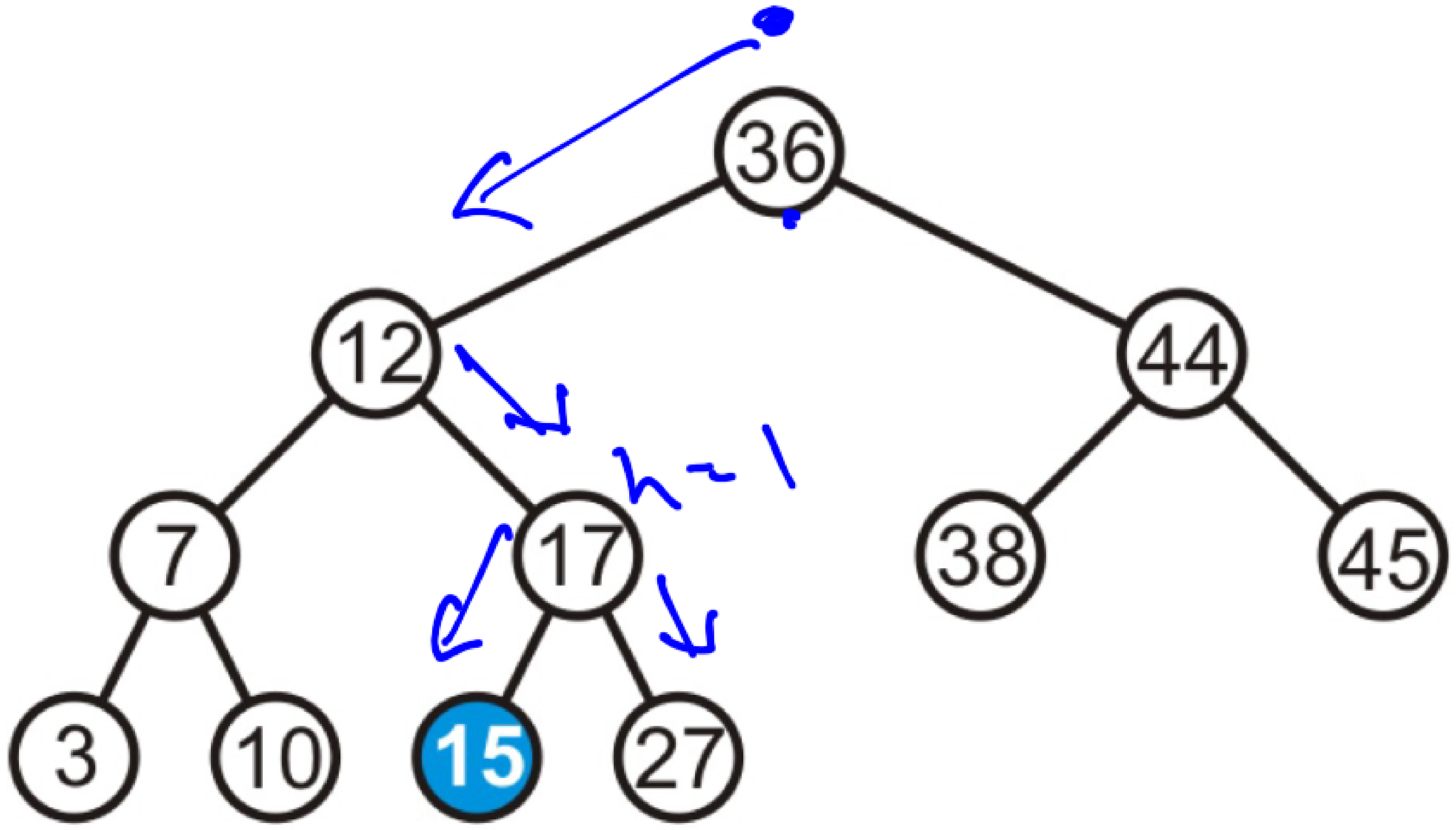


Image source: Douglas Wilhelm Harder, University of Waterloo, Canada

Maintaining Balance

Heights of none of the subtrees change

The tree is still balanced

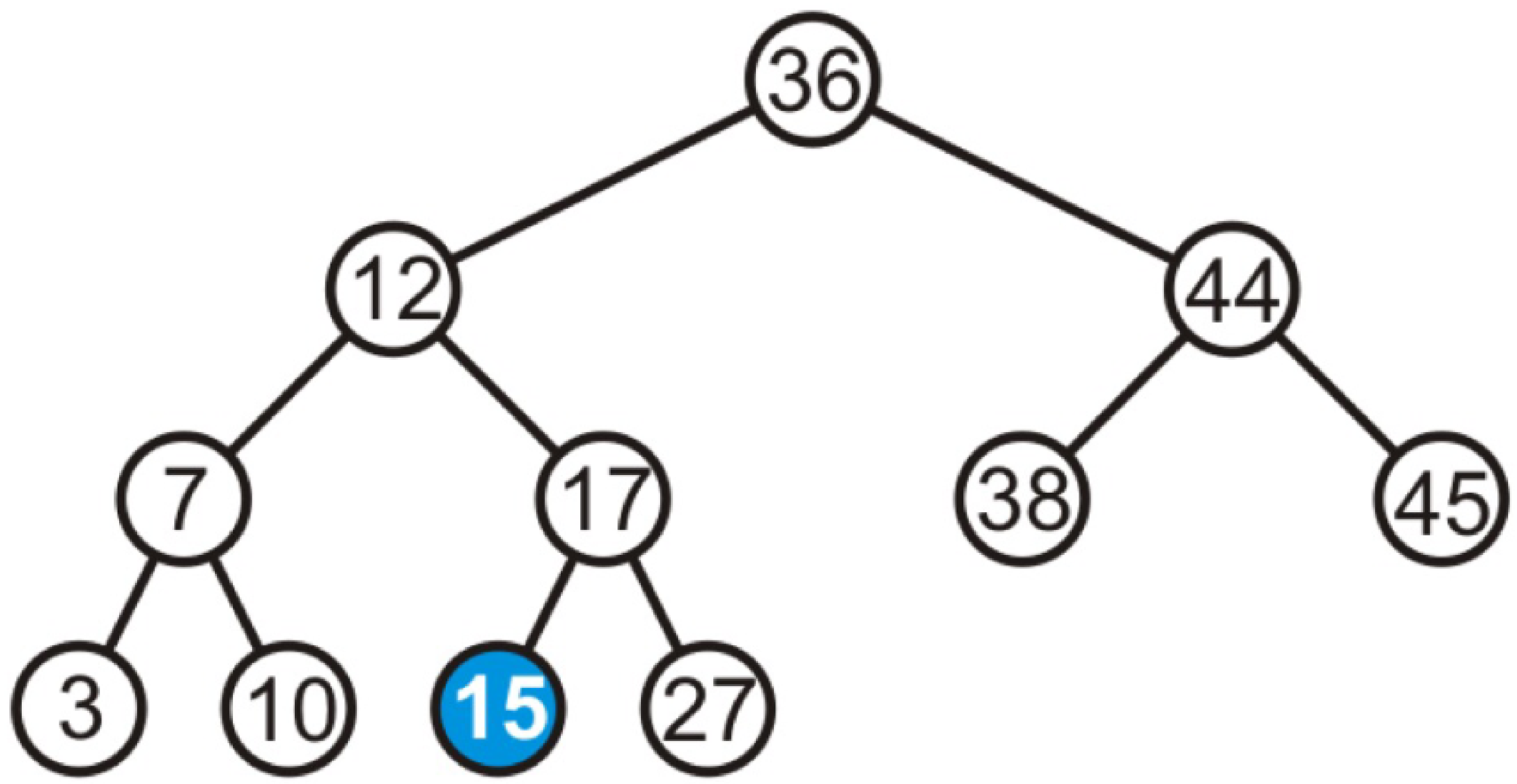


Image source: Douglas Wilhelm Harder, University of Waterloo, Canada

Maintaining Balance

- Insert 42 now

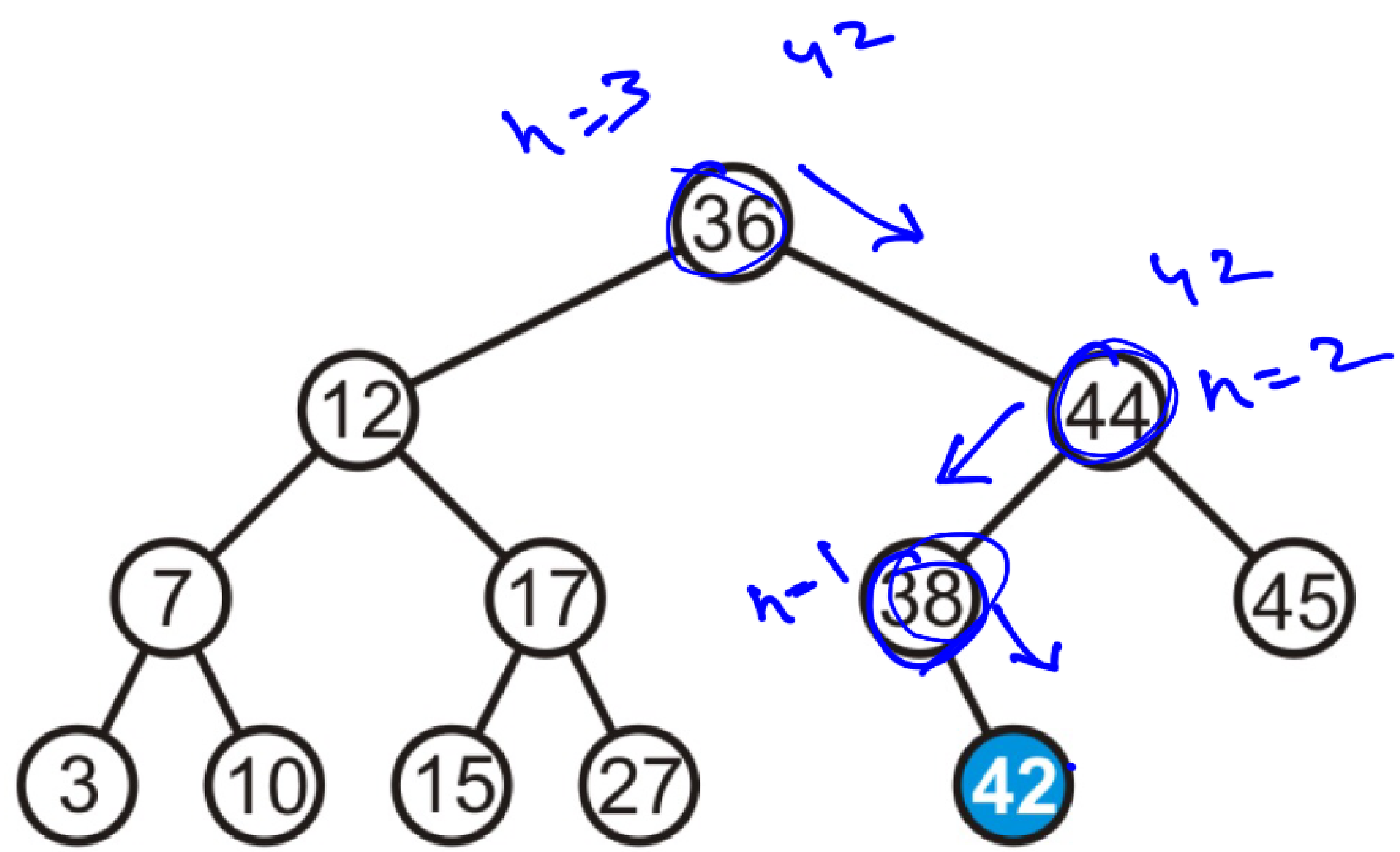


Image source: Douglas Wilhelm Harder, University of Waterloo, Canada

Maintaining Balance

Now the heights of two sub-trees have increased by one

The tree is still balanced

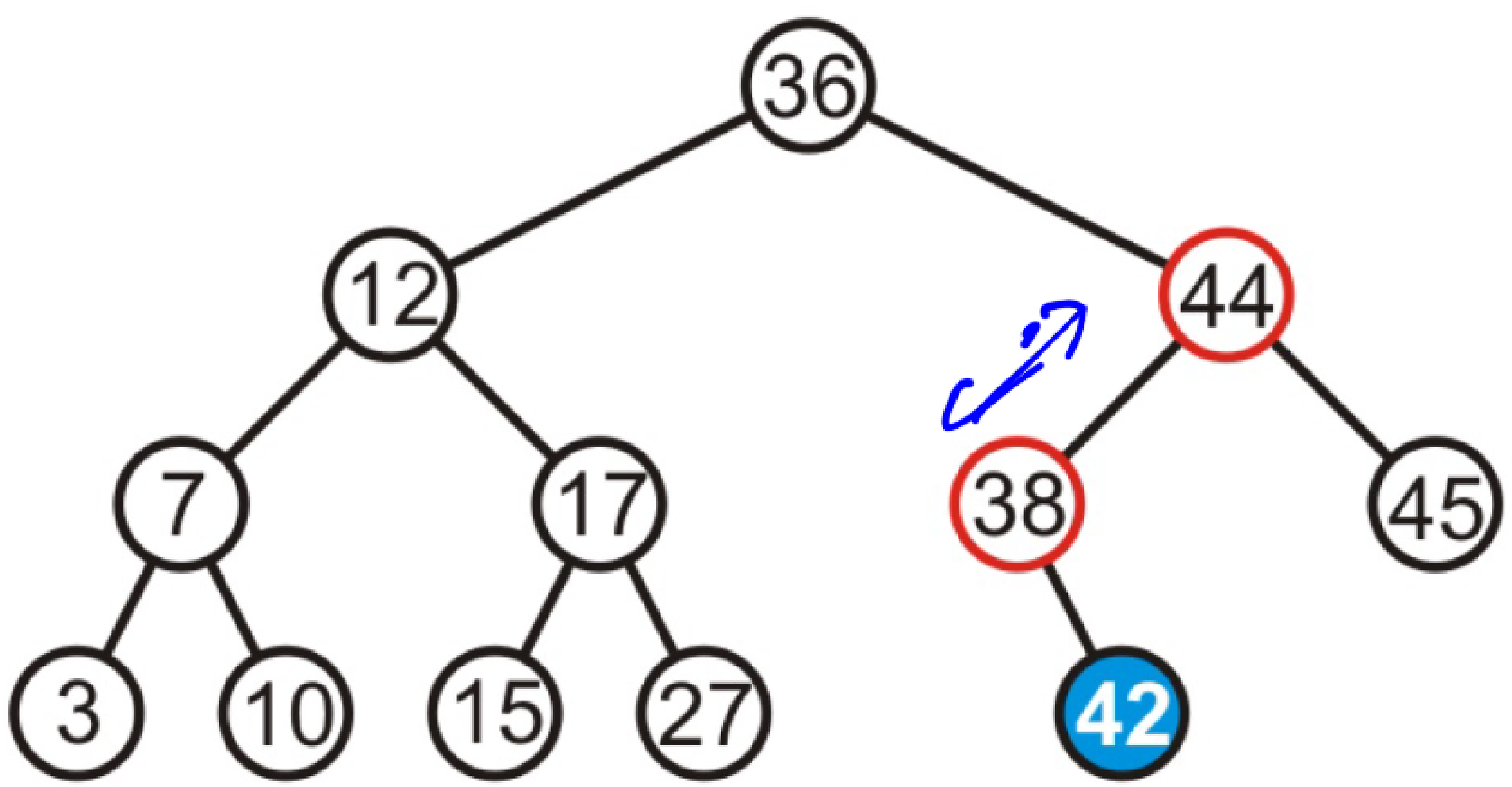
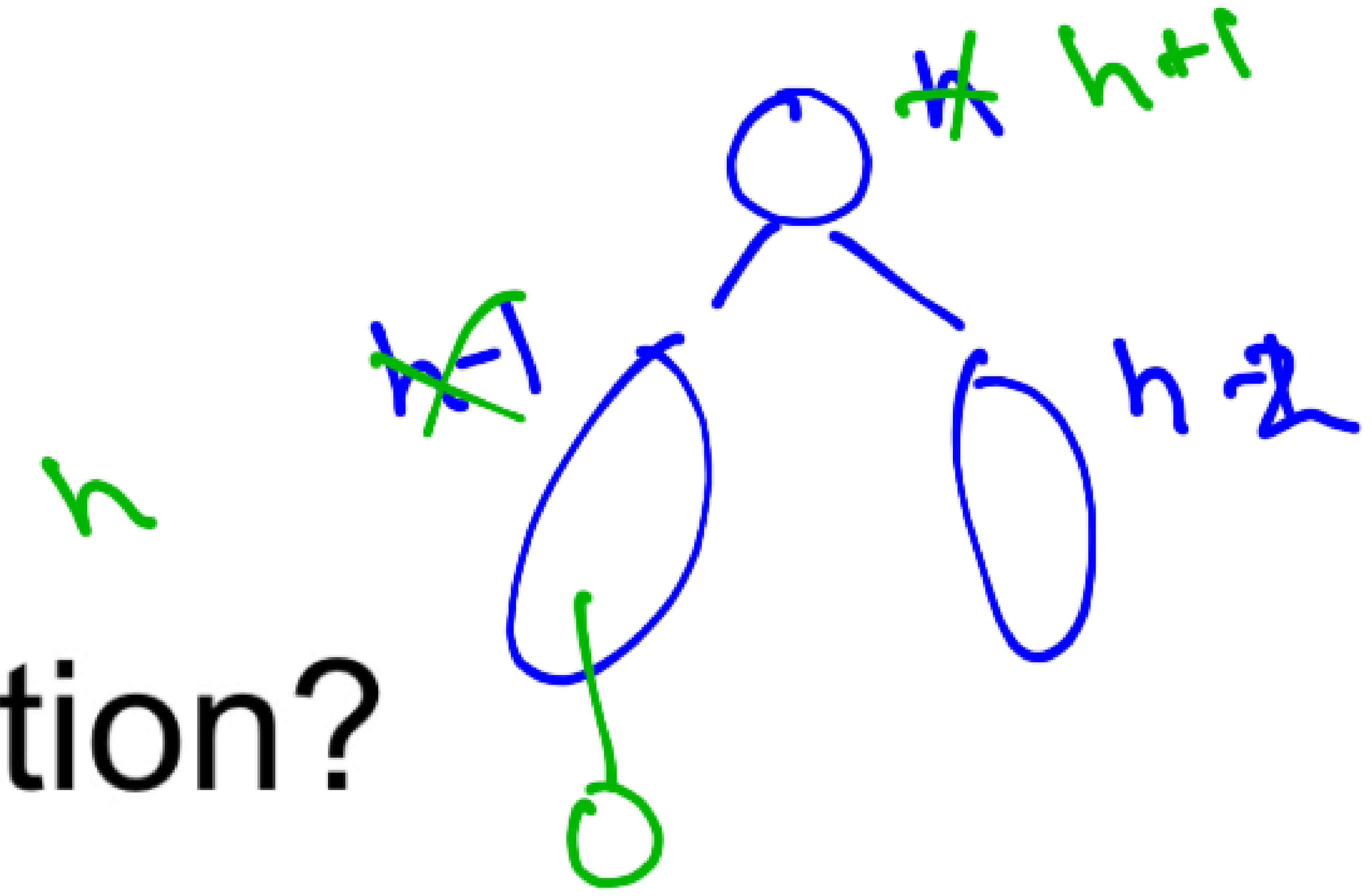


Image source: Douglas Wilhelm Harder, University of Waterloo, Canada

Maintaining Balance



How to create imbalance after insertion?

- Insert in a subtree where the heights of the two sub-trees differ by 1
- The insertion must increase the height of the deeper sub-tree

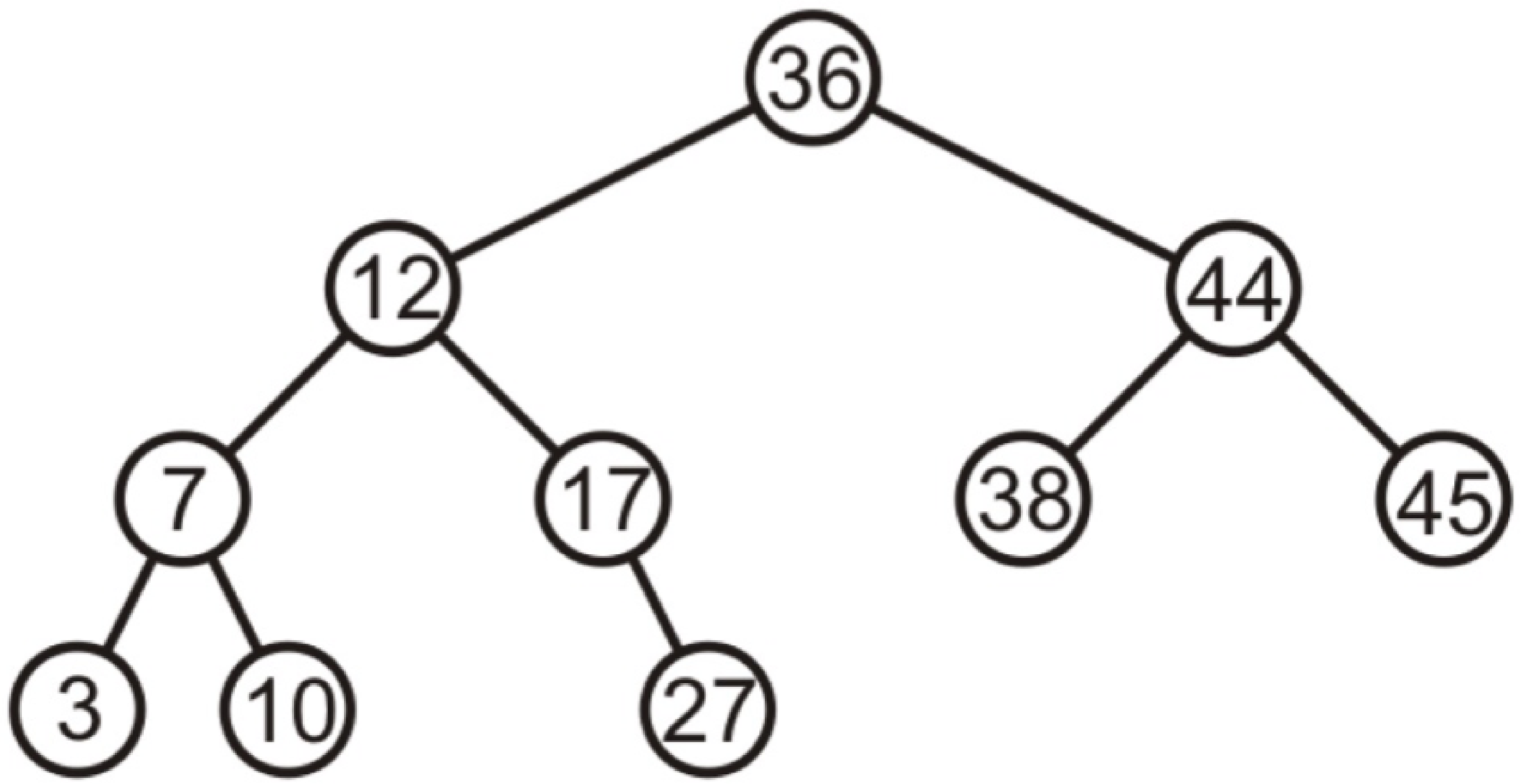
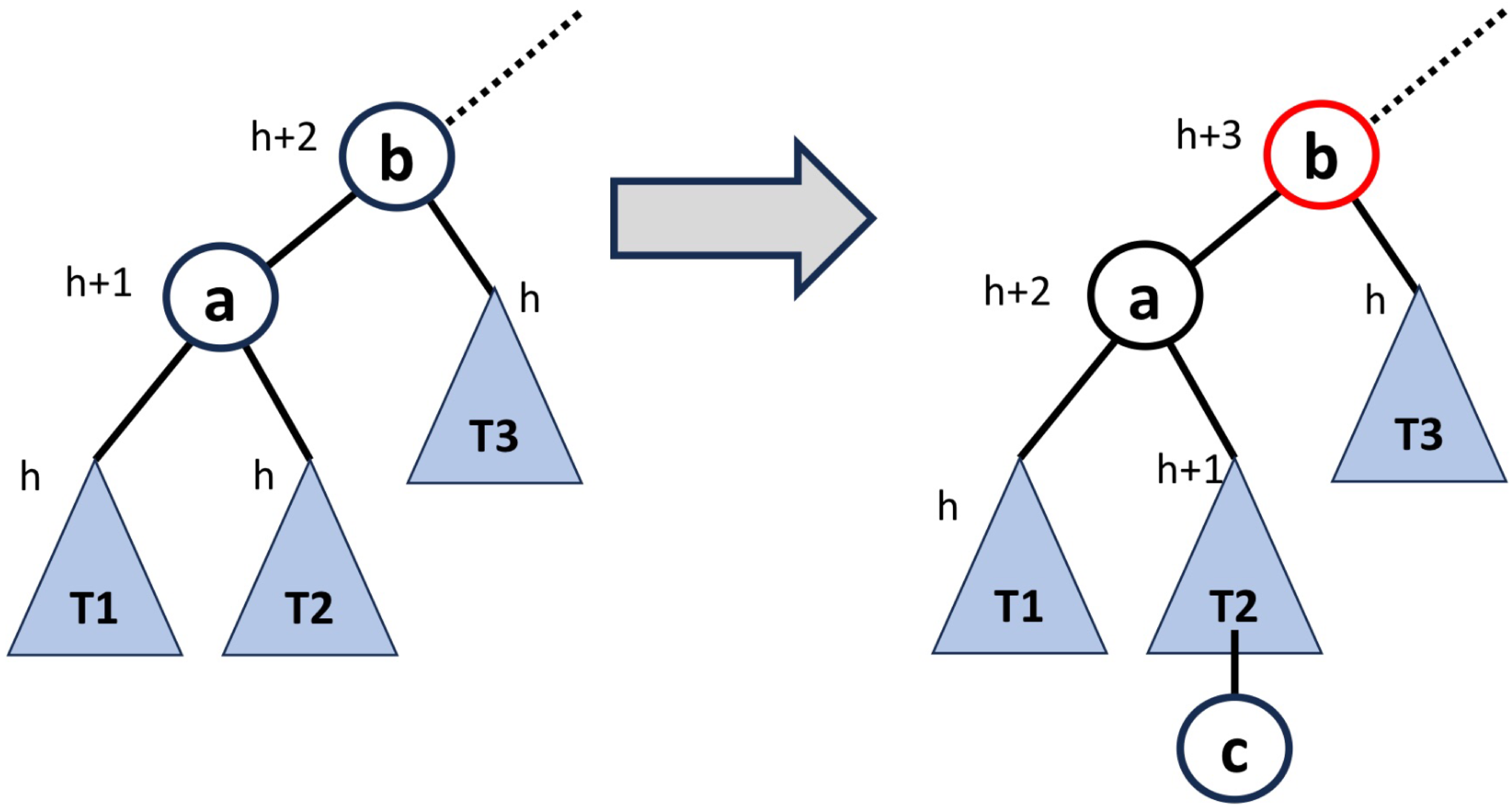


Image source: Douglas Wilhelm Harder, University of Waterloo, Canada

Insertion: Cases

- Assume that c is inserted in the AVL tree
- The first node (on path from c to the root) to become unbalanced is b
- Case 1: **c** is inserted in the left subtree of the left child of b
- Case 2 [Symmetric]: **c** is inserted in the right subtree of the right child of b
- Case 3: **c** is inserted in the right subtree of the left child of b
- Case 4 [Symmetric]: **c** is inserted in the left subtree of the right child of b

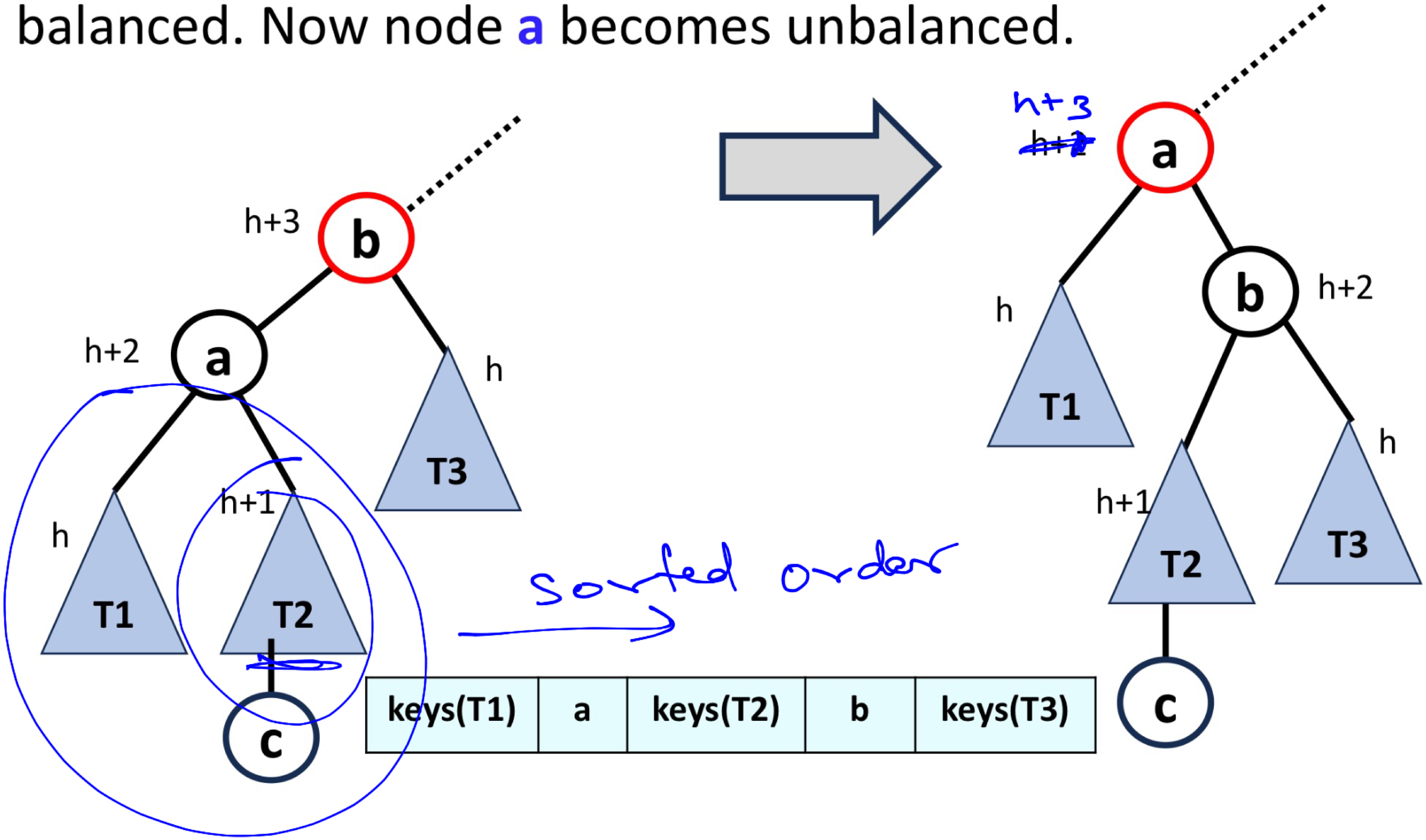
AVL Tree Insertion: Case 3



keys(T1)	a	keys(T2)	b	keys(T3)

AVL Tree Rotation: Case 3

Unfortunately, the same rotation doesn't make the tree balanced. Now node **a** becomes unbalanced.

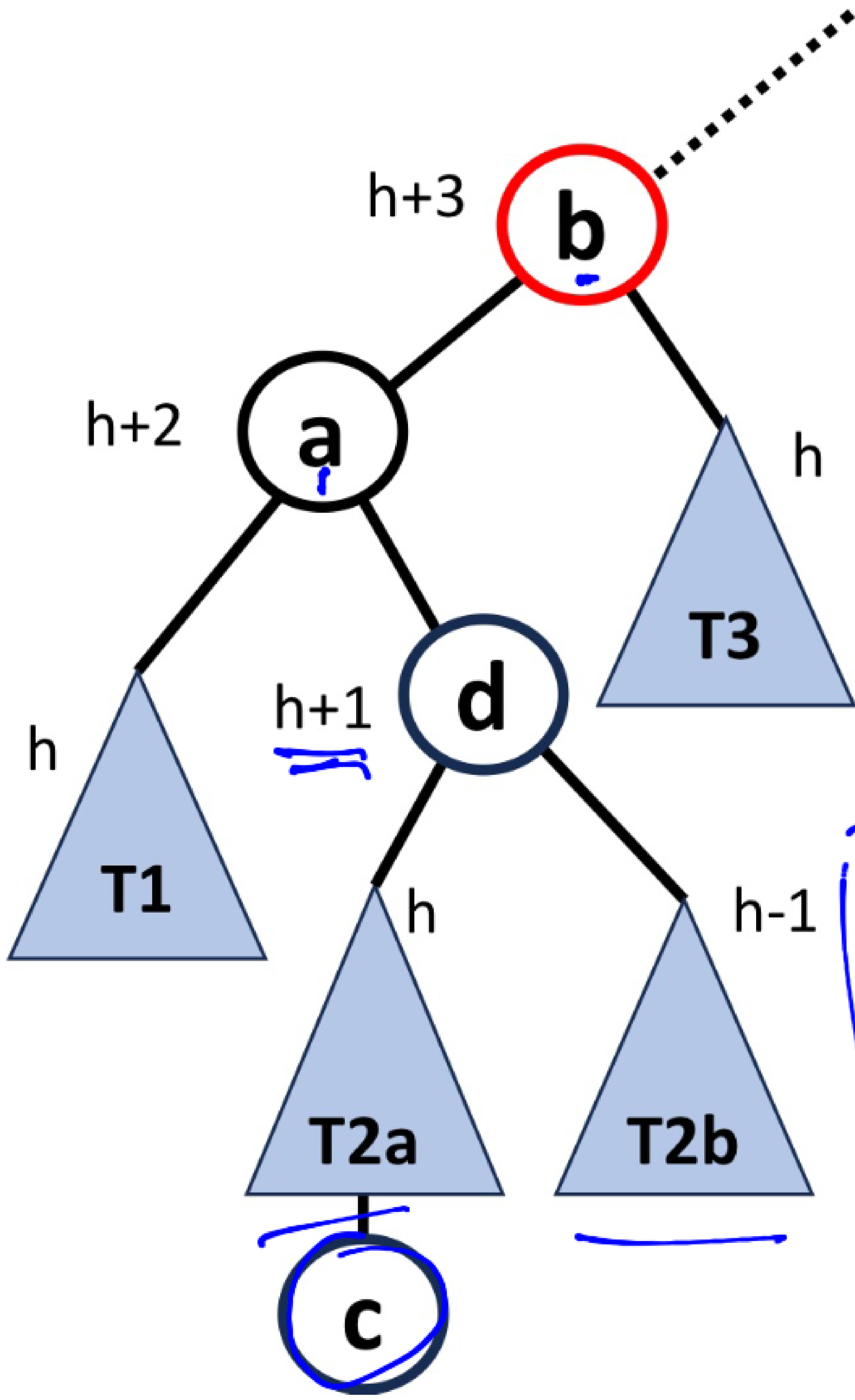


keys(T1)	a	keys(T2)	b	keys(T3)

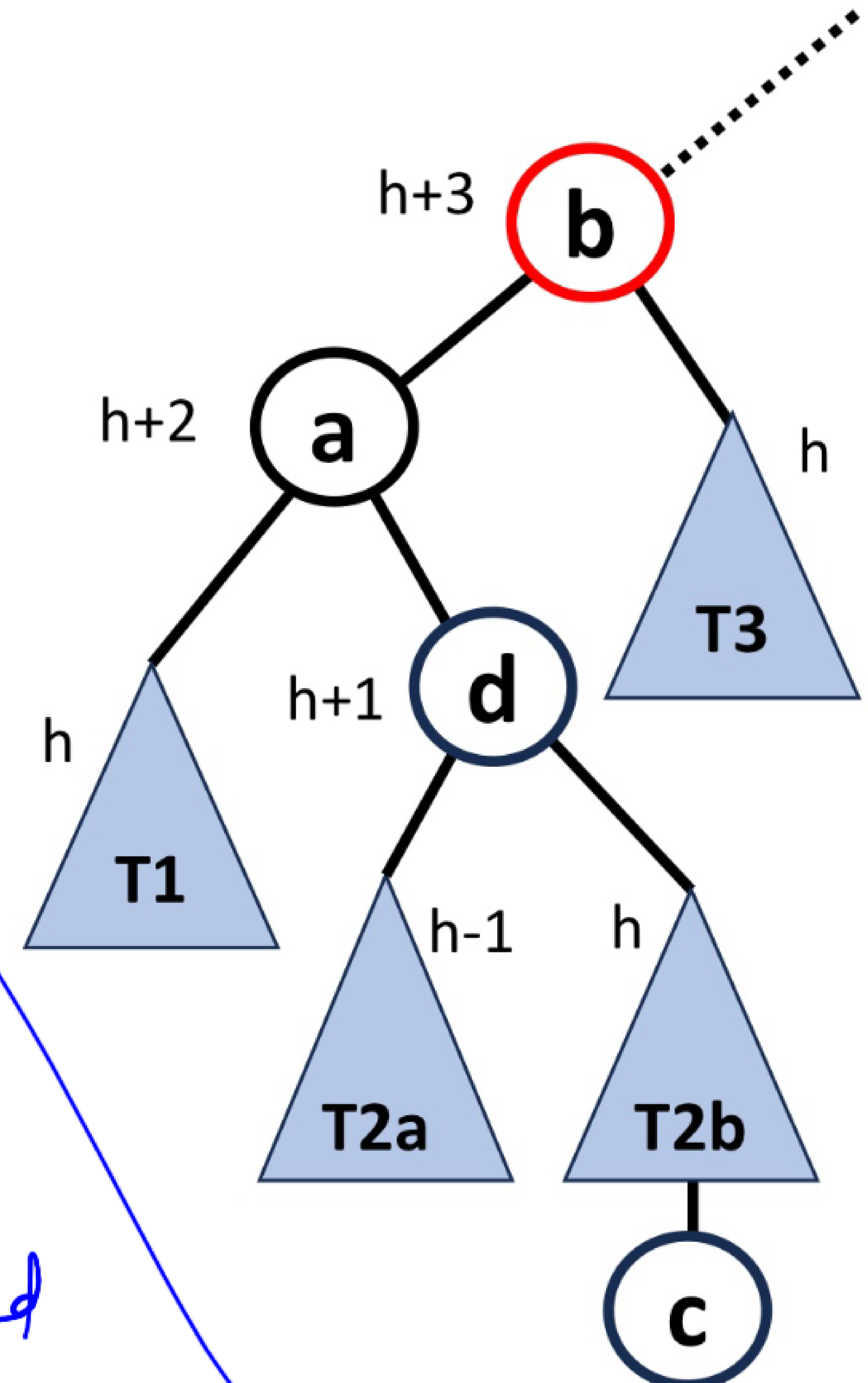
AVL Tree Rotation: Case 3

Solution: Split T2 into two sub-trees: T2a and T2b

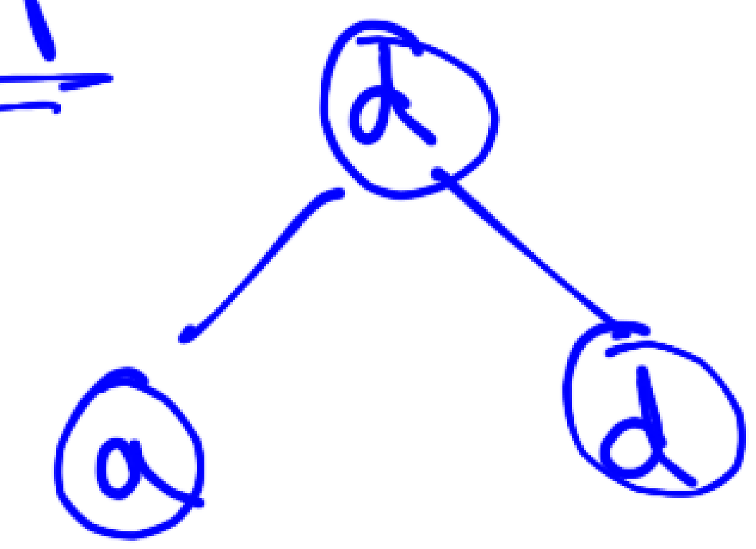
Two subcases can be handled identically



h => Not possible Because Tree was balanced before insertion of c



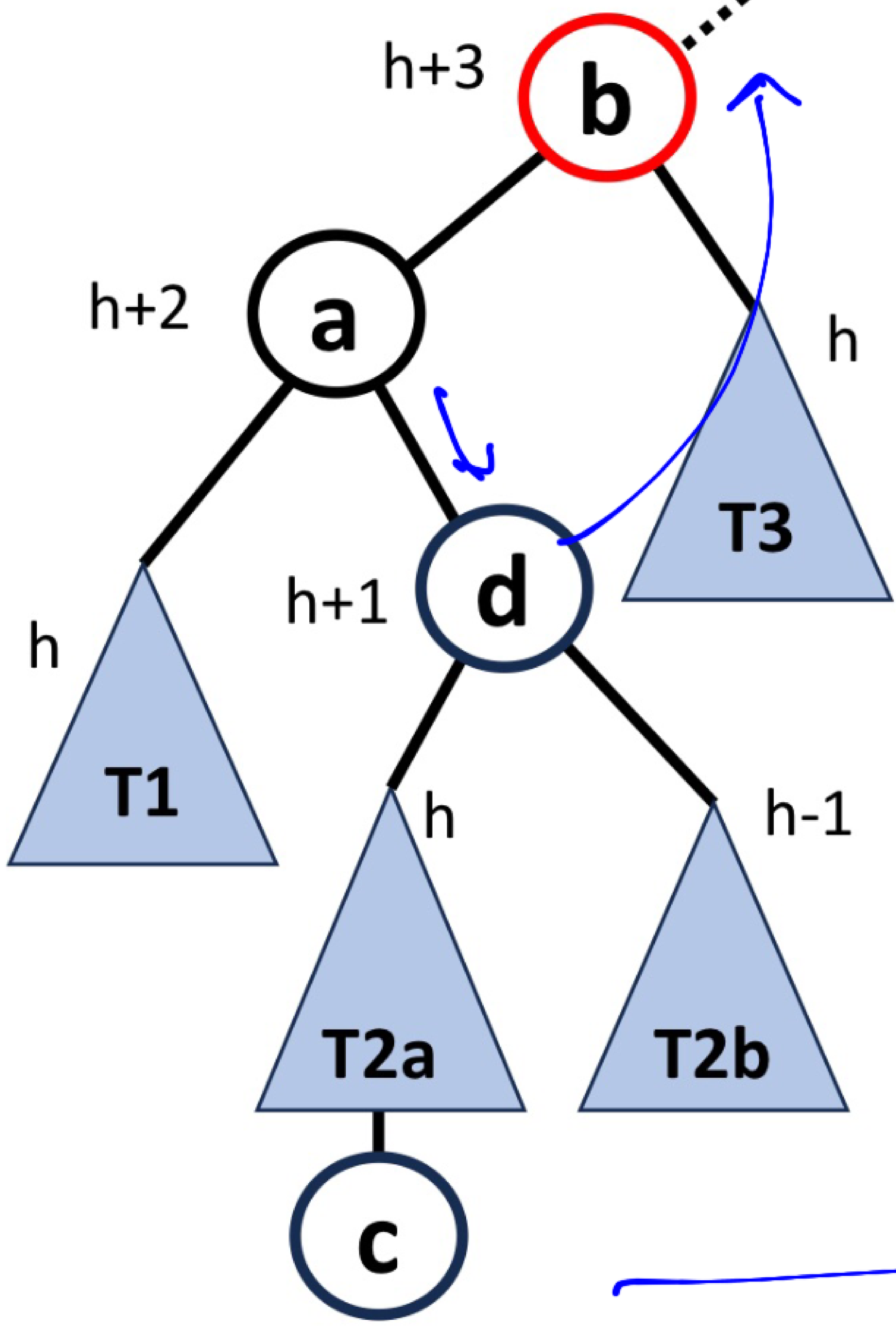
Proposal: Move d to the top

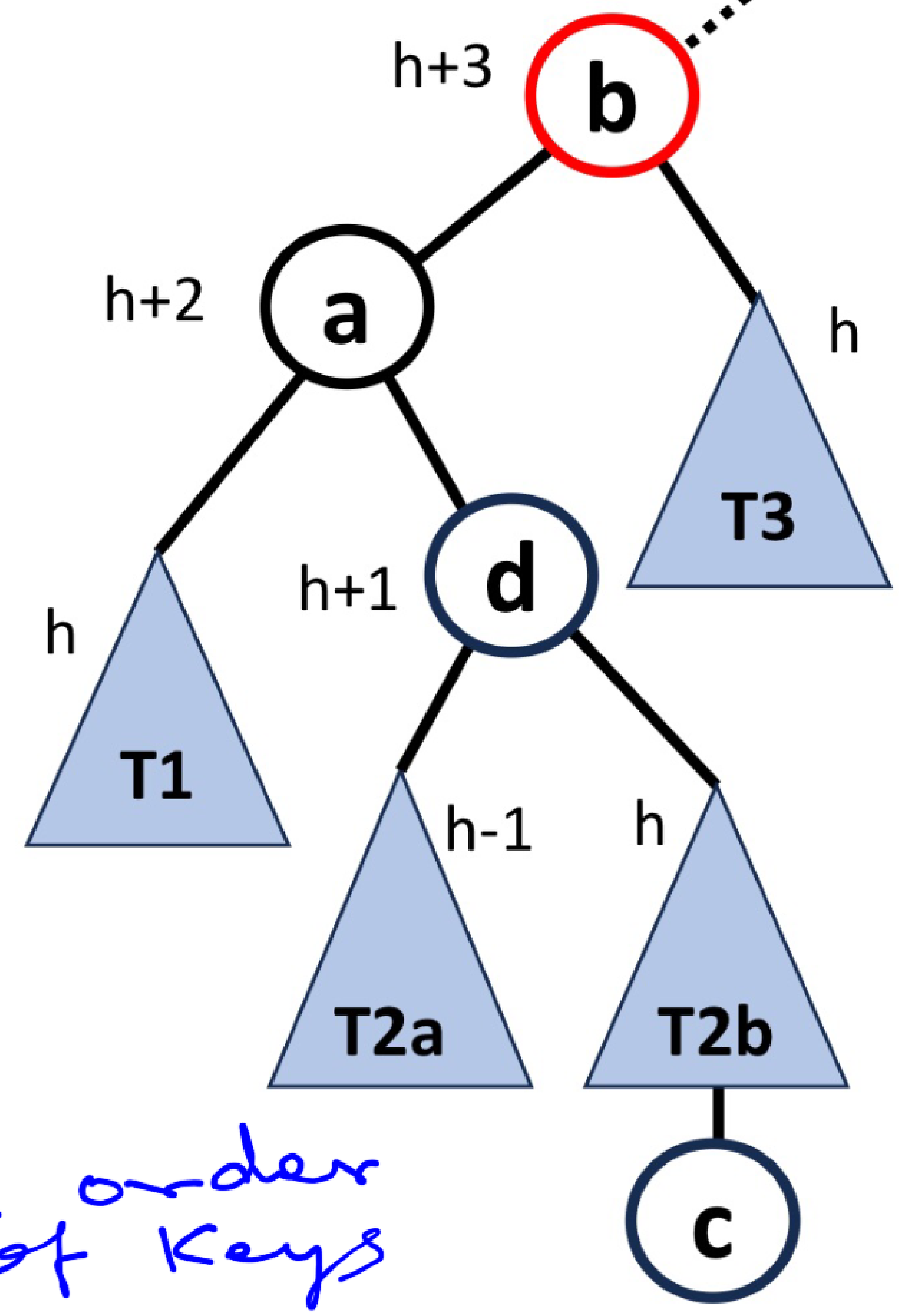


AVL Tree Rotation: Case 3

Solution: Split T2 into two sub-trees: T2a and T2b

Two subcases can be handled identically



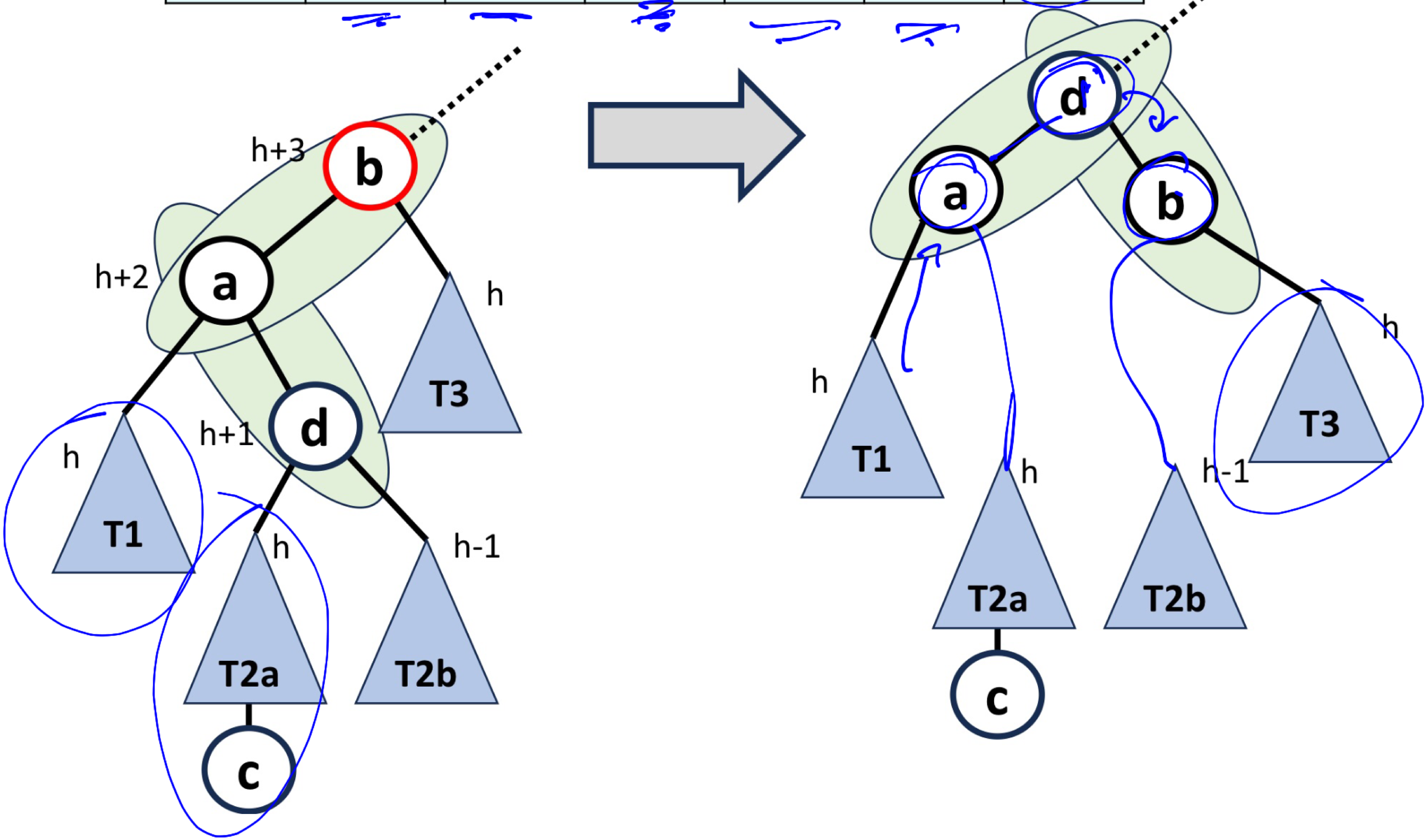


Ascending order of Keys

T1	a	T2a	d	T2b	b	T3

AVL Tree Rotation: Case 3

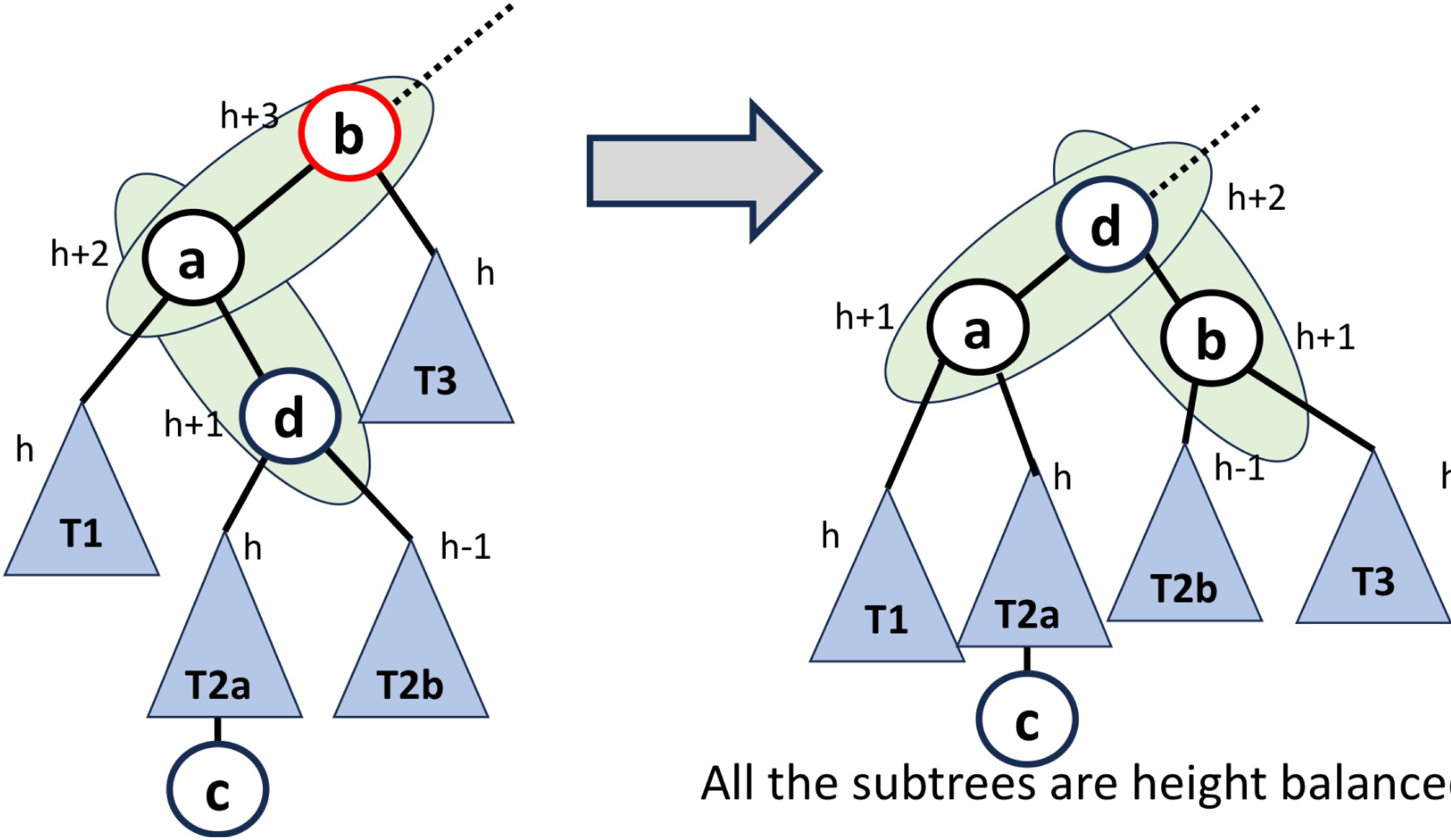
T1	a	T2a	d	T2b	b	T3



AVL Tree Rotation: Case 3

Keys →

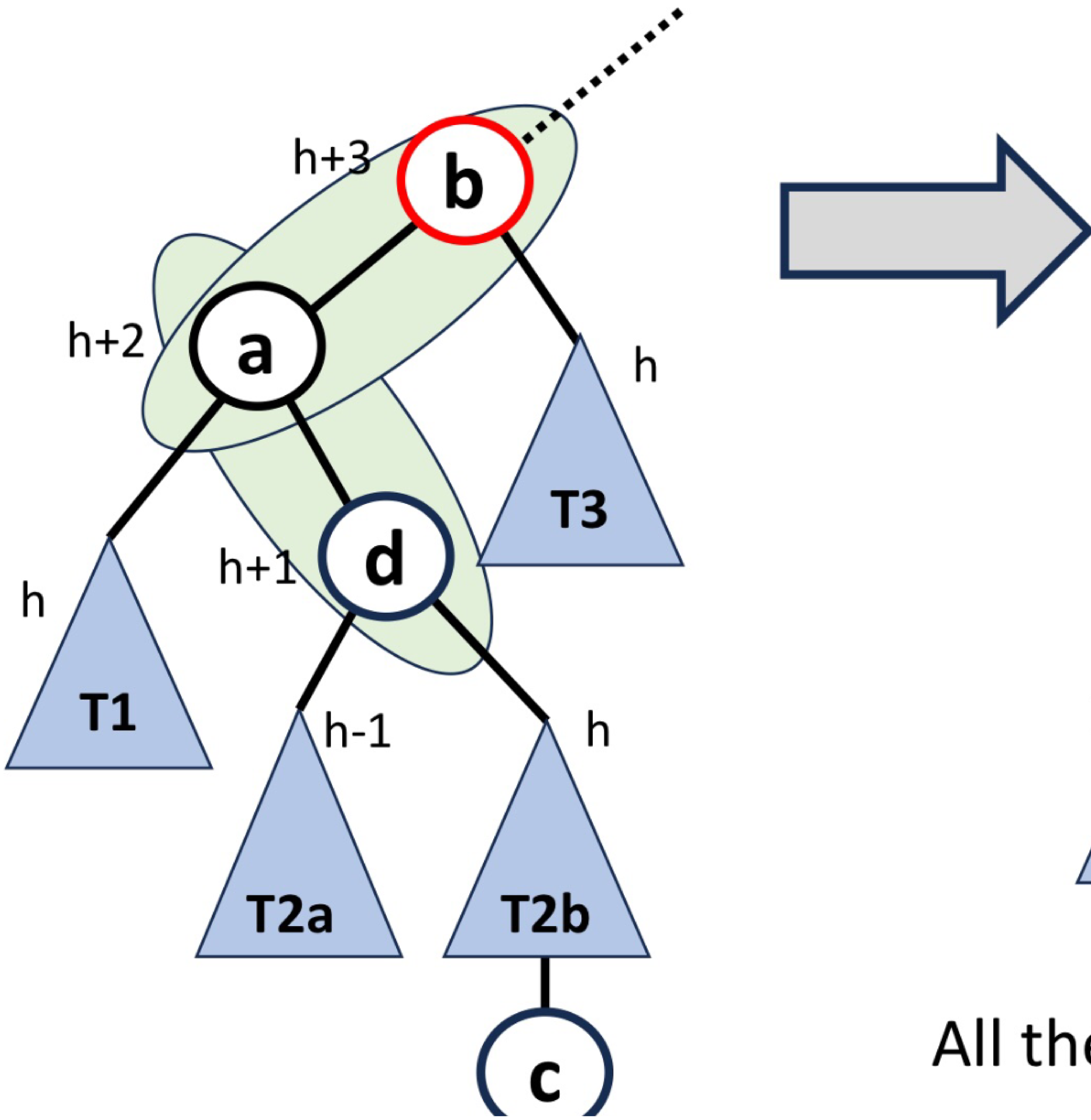
T1	a	T2a	d	T2b	b	T3

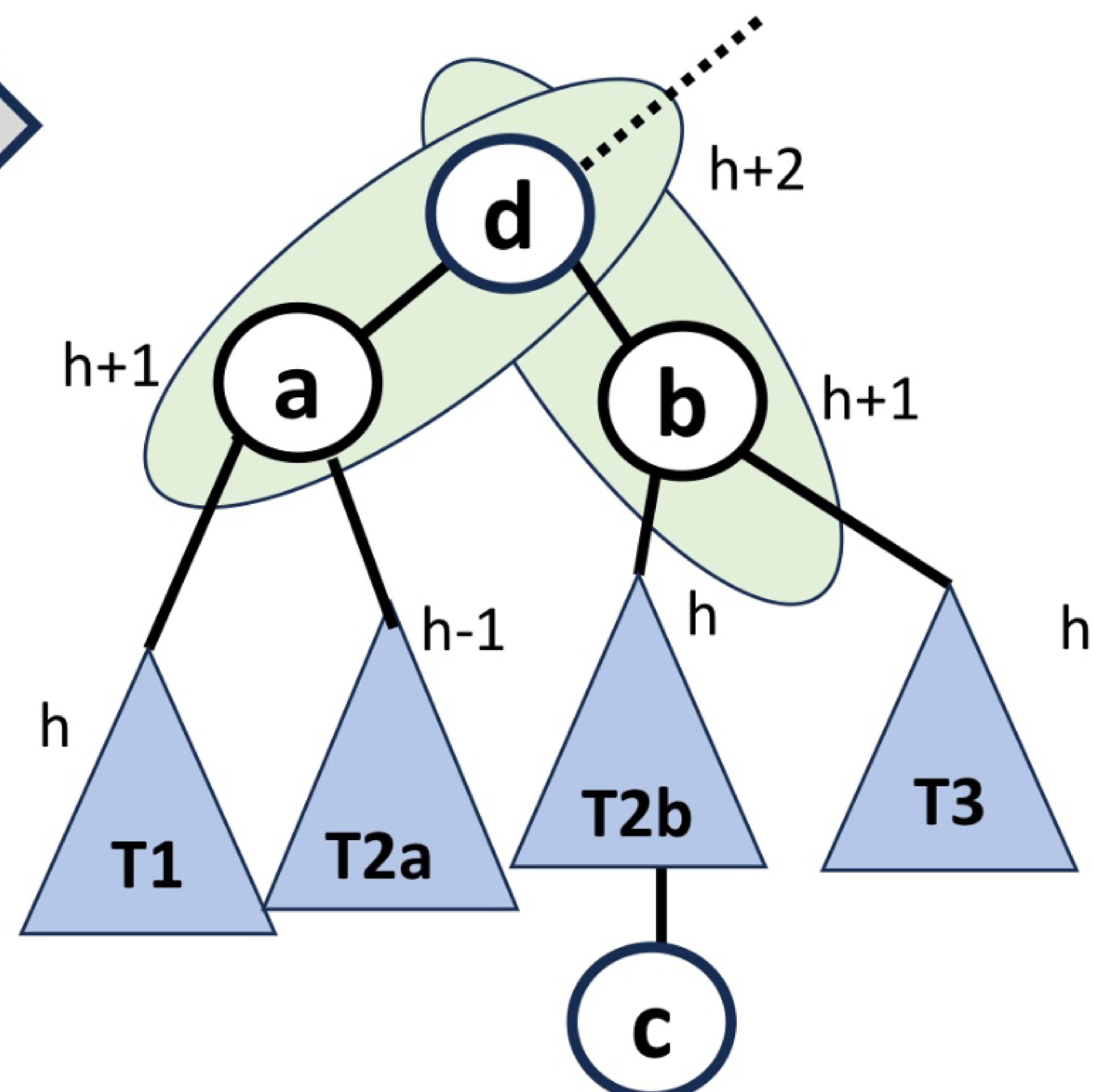


All the subtrees are height balanced

AVL Tree Rotation: Case 3b

T1	a	T2a	d	T2b	b	T3





All the subtrees are height balanced

AVL Tree Insert Rotations

- Case 4 [Symmetric to Case 3]: **c** is inserted in the left subtree of the right child of b
- b is the first node in the path from c to the root to have imbalance
- Homework: Work out the details of rotation for case 4 of AVL tree insertions

Thank You