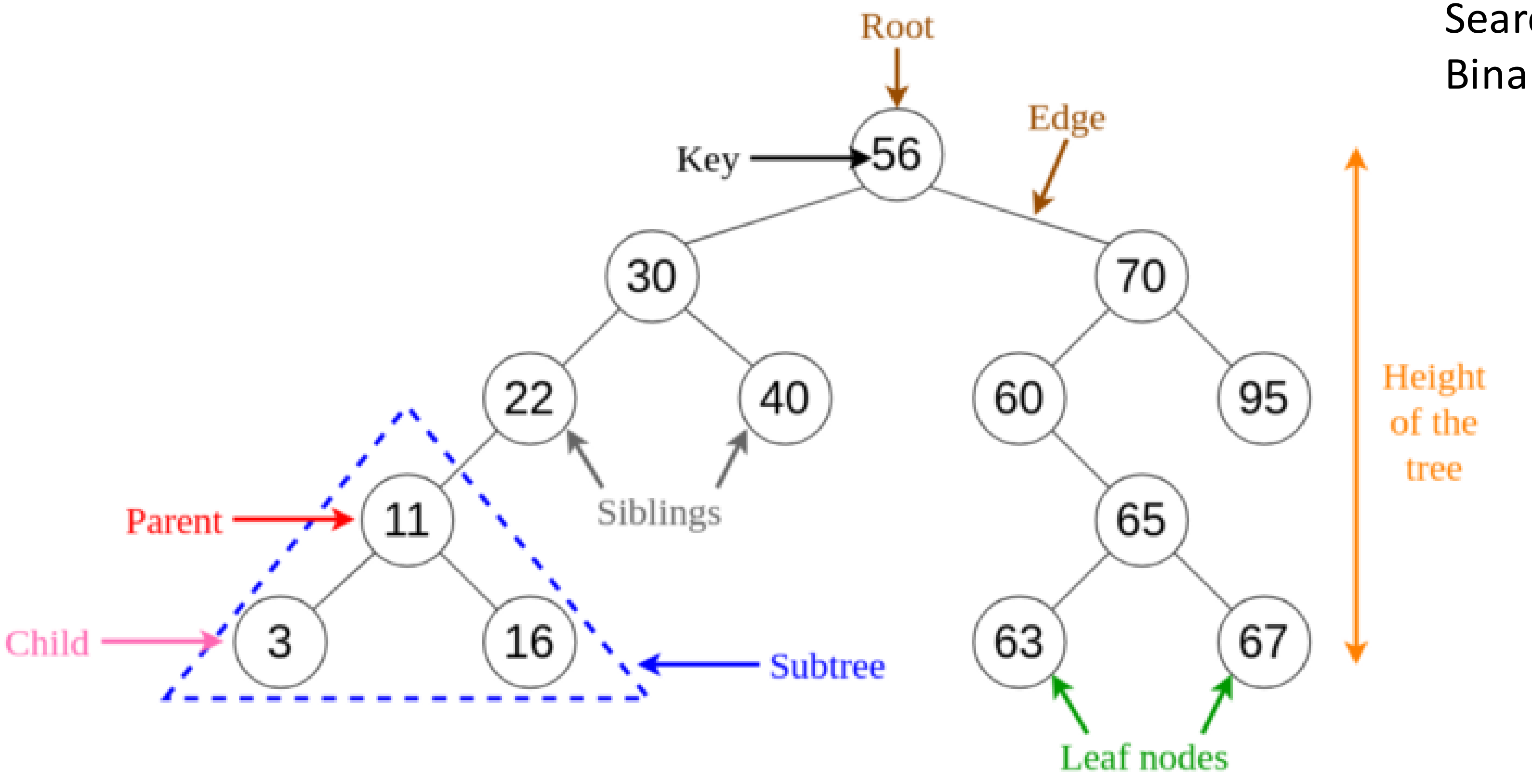
AVL (Adelson-Velsky and Landis) Tree

Bill

- ## Binary Search Trees

If we have a sequence {56, 30, 22, 70, 40, 60, 95, 65, 11, 3, 16, 63, 67}, we can form a BST

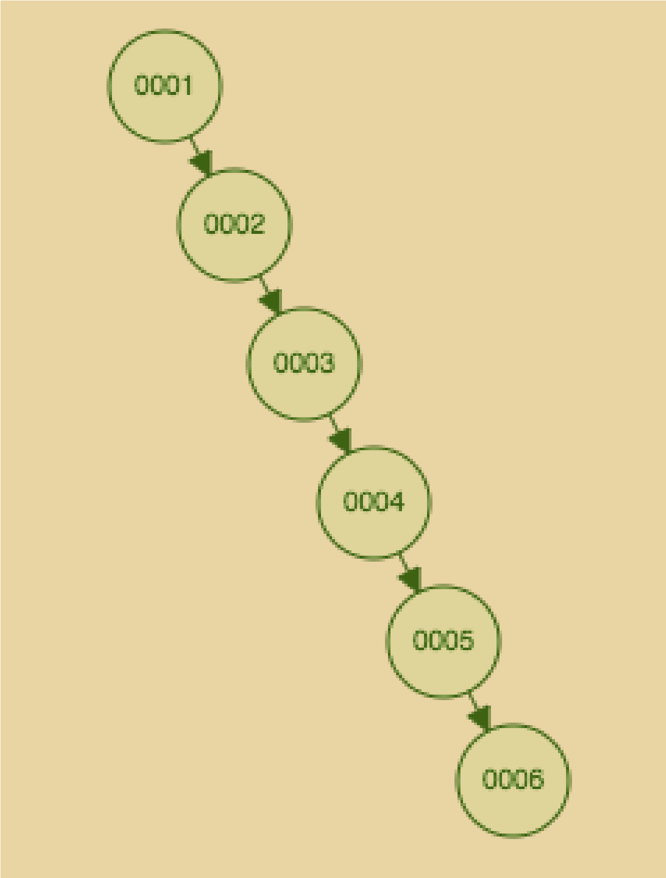


Search time complexity of Binary search is $O(\log_2 n)$

Image if from https://levelup.gitconnected.com/an-into-to-binary-search-trees-432f94d180da

- # Binary Search Trees

What if we have a sequence {1, 2, 3, 4, 5, 6, …}, the BST we built will be highly imbalanced

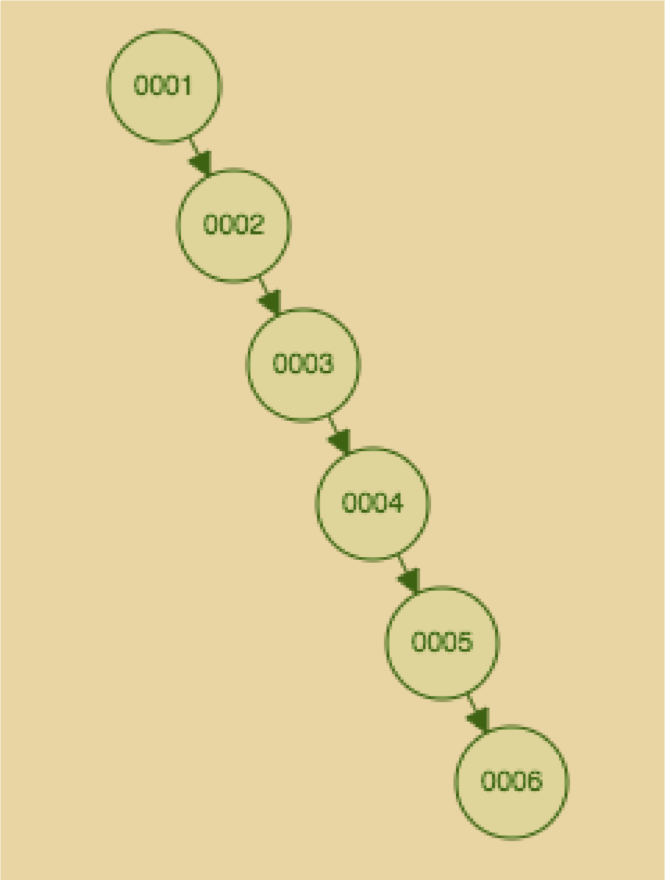


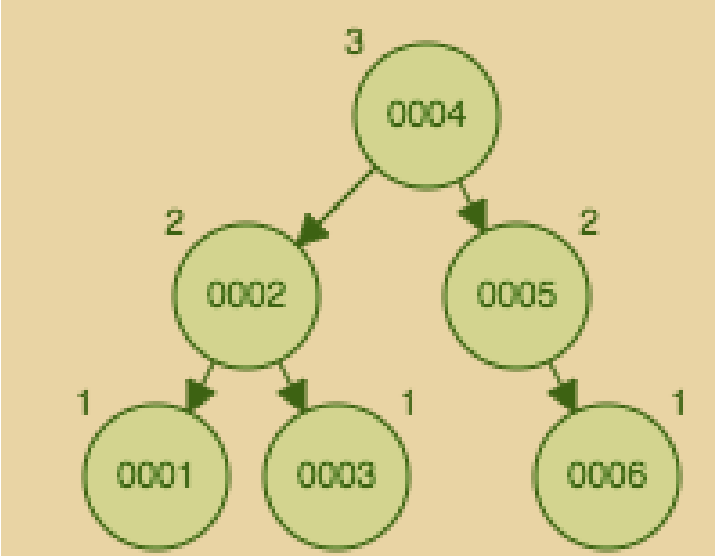
Time complexity becomes $O(n)$ rather than $O(\log_2 n)$

https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

• Balanced Binary Search Trees

In order to solve this problem, we have balanced BST



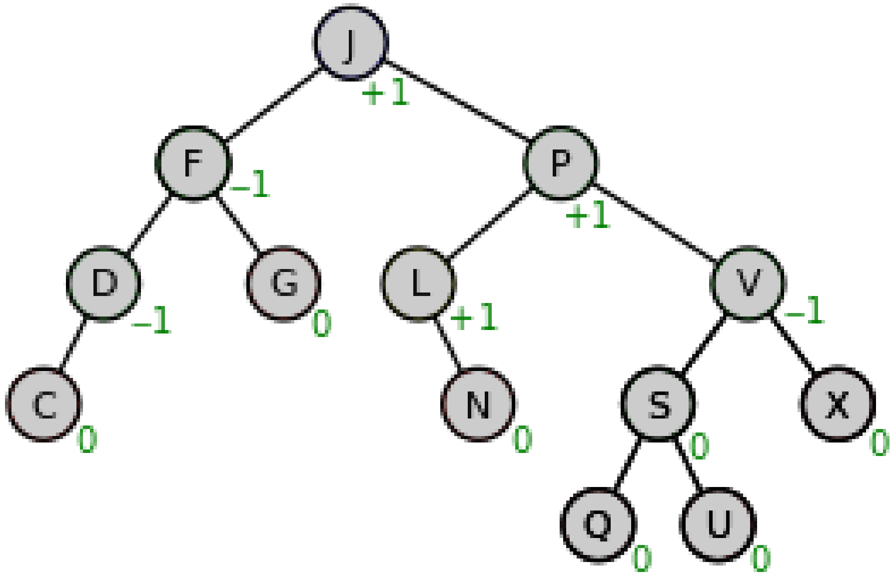


- **AVL Tree --- a self-balancing binary search tree**

$$\mathrm{BalanceFactor}(node) := \mathrm{Height}(\mathrm{RightSubtree}(node)) - \mathrm{Height}(\mathrm{LeftSubtree}(node))$$

(1) We hope |BalanceFactor| < 2

(2) Rotation operations



$$Rotations \begin{cases} RR, & Left\ rot \\ LL, & Right\ rot \\ RL, & Right + Left\ rot \\ LR, & Left + Right\ rot \end{cases}$$

- # AVL Tree --- An intuitive view

$$\mathrm{BalanceFactor}(node) := \mathrm{Height}(\mathrm{RightSubtree}(node)) - \mathrm{Height}(\mathrm{LeftSubtree}(node))$$

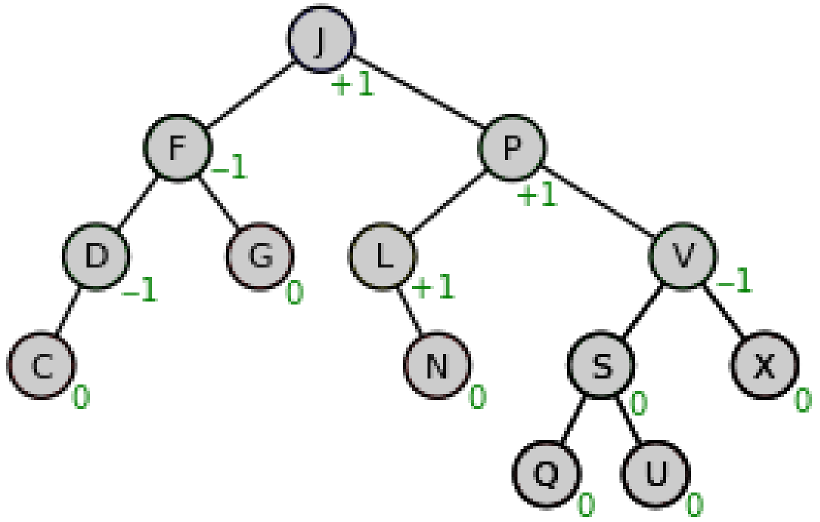
(1) We hope |BalanceFactor| < 2

(2) Rotation operations

$$Rotations \begin{cases} RR, & Left\ rot \\ LL, & Right\ rot \\ RL, & Right + Left\ rot \\ LR, & Left + Right\ rot \end{cases}$$

The gif is from https://en.wikipedia.org/wiki/AVL_tree

- ## Tips of AVL Tree



(1)* Classification (RR, LL, RL, LR)

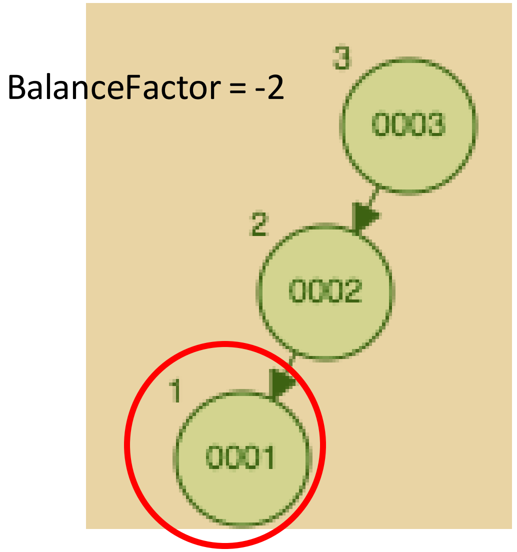
(2) Take over (Left rot, Right rot)

(3) Delegation (RL, LR)

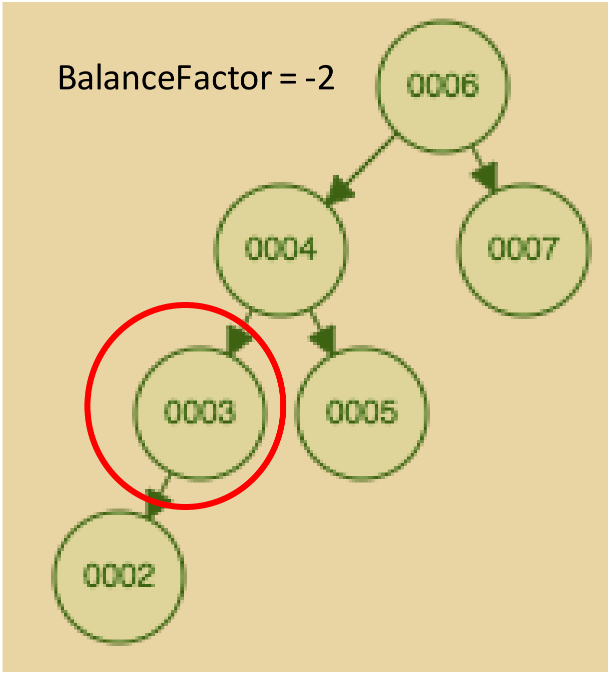
$$Rotations \begin{cases} RR, & Left\ rot \\ LL, & Right\ rot \\ RL, & Right + Left\ rot \\ LR, & Left + Right\ rot \end{cases}$$

- ## LL (Left-Left)

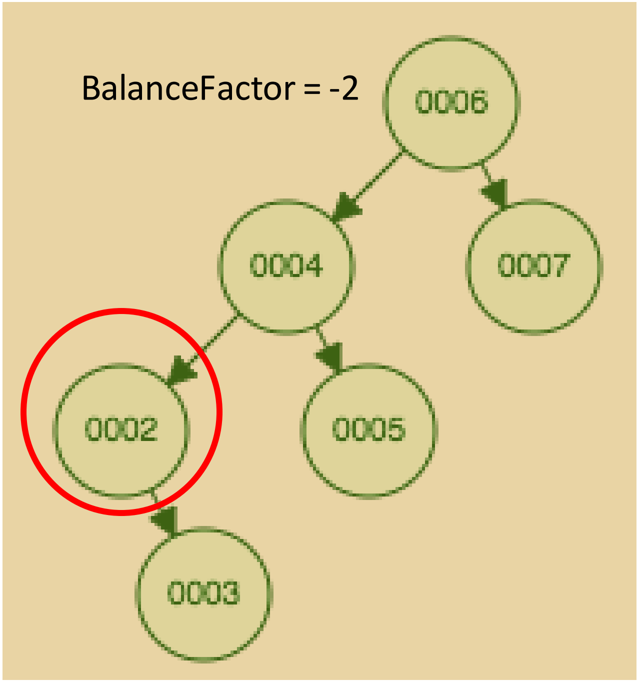
$$Rotations \begin{cases} RR, & Left\ rot \\ LL, & Right\ rot \\ RL, & Right + Left\ rot \\ LR, & Left + Right\ rot \end{cases}$$



(I)



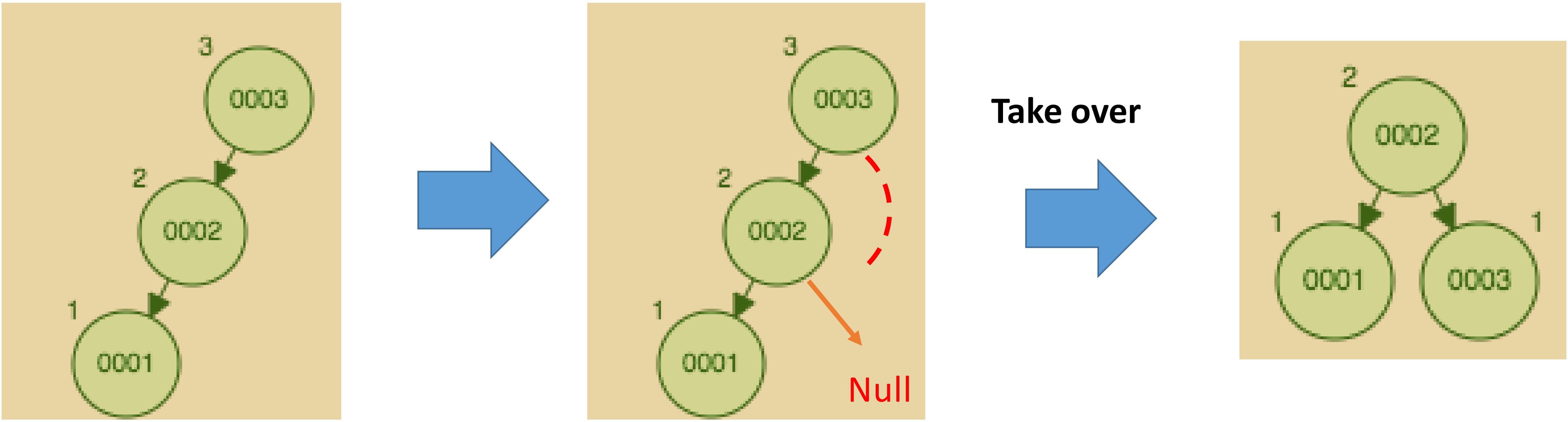
(II)



(III)

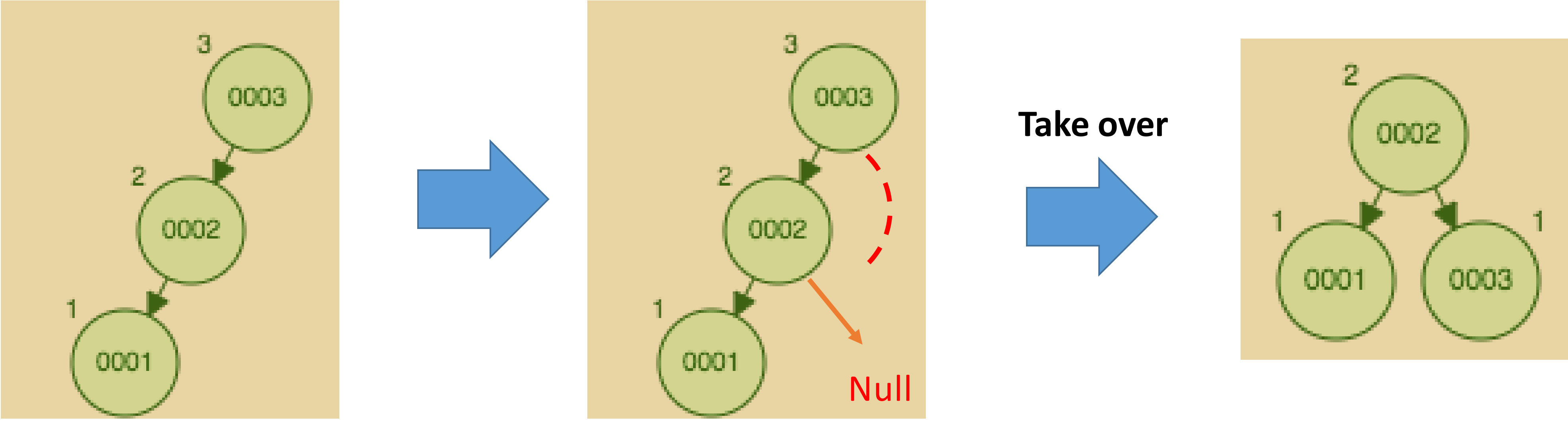
- ## LL (Left-Left) --- I

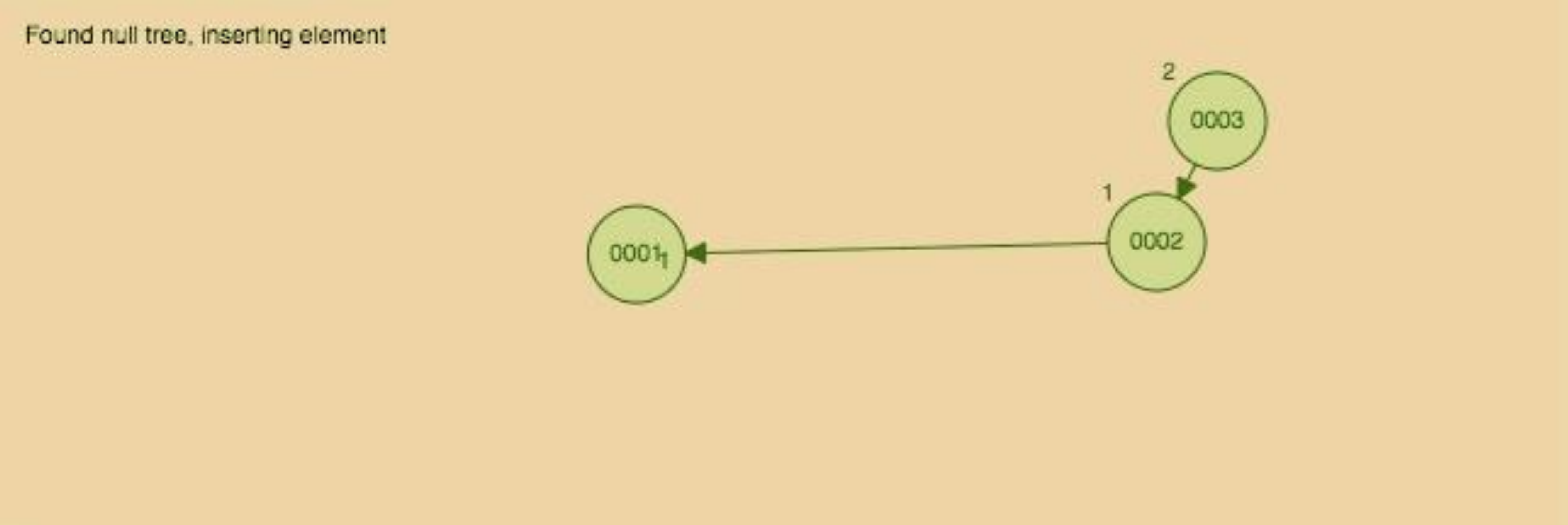
$$Rotations \begin{cases} RR, \ Left \ rot \\ LL, \ Right \ rot \\ RL, \ Right + Left \ rot \\ LR, \ Left + Right \ rot \end{cases}$$



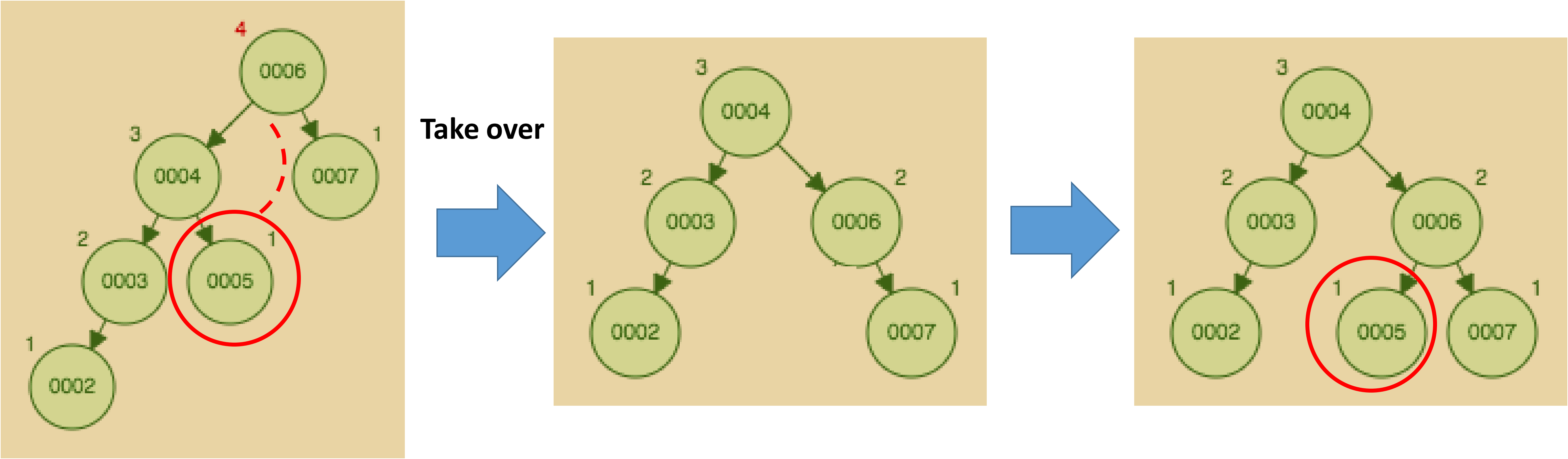
(I)

- **LL (Left-Left) --- I**



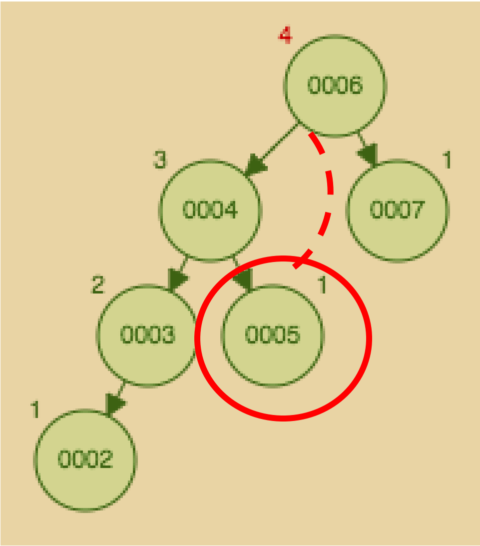


- **LL (Left-Left) --- II**

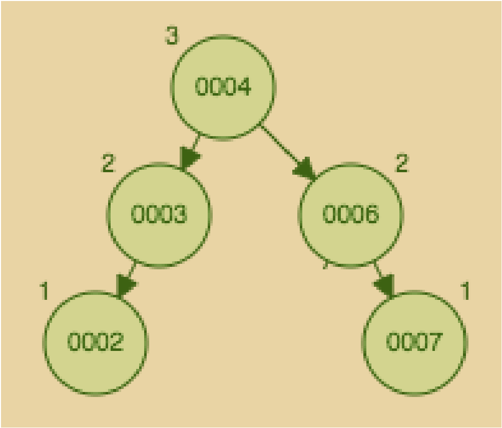


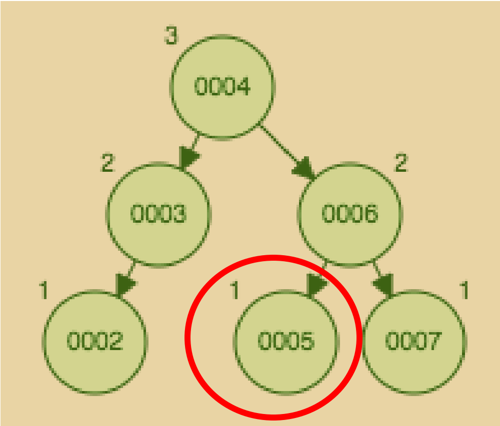
(II)

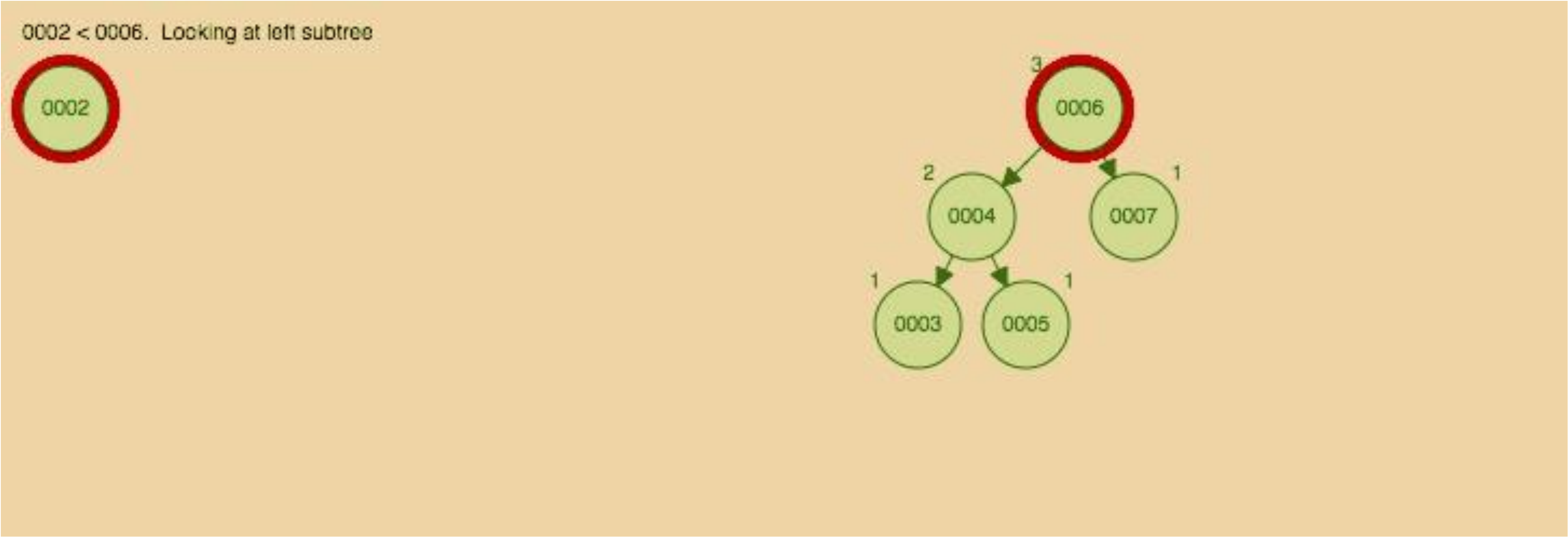
LL (Left-Left) --- II



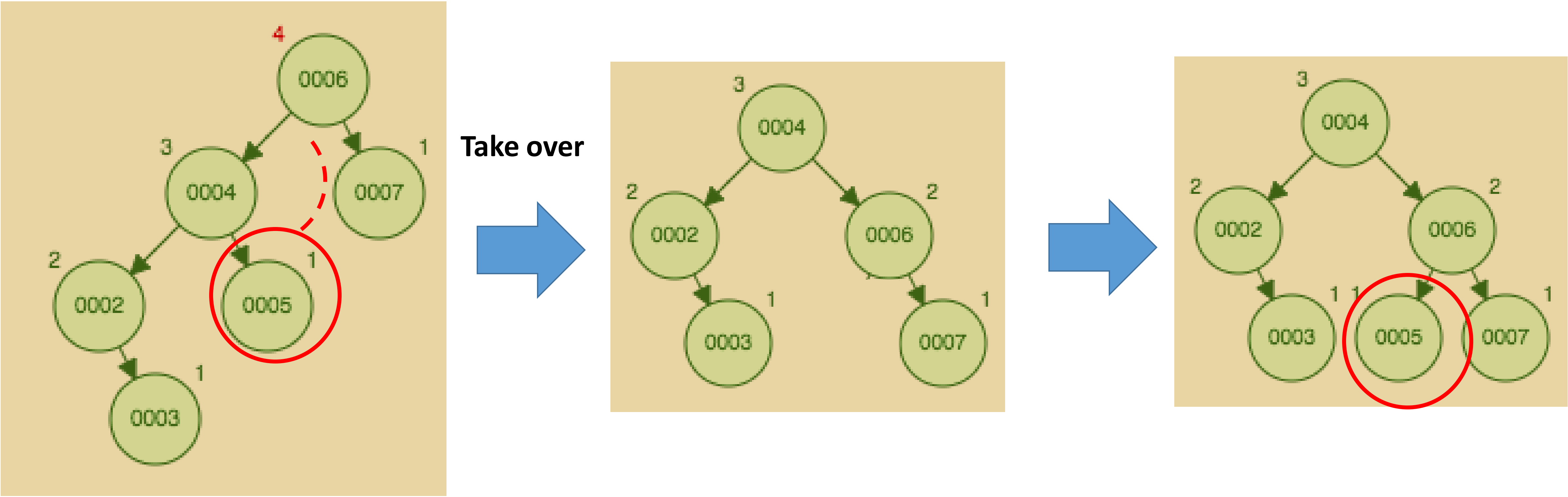
Take over





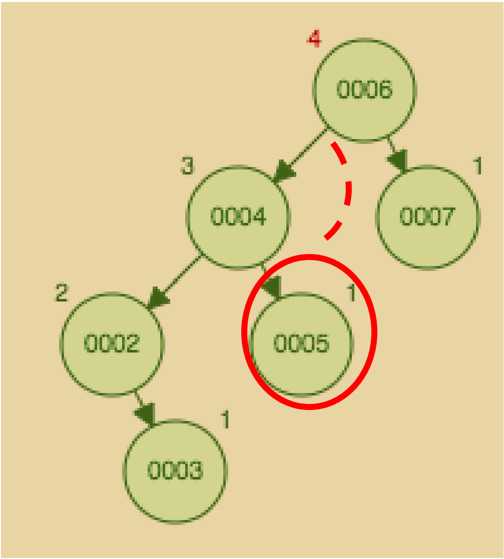


- **LL (Left-Left) --- III**

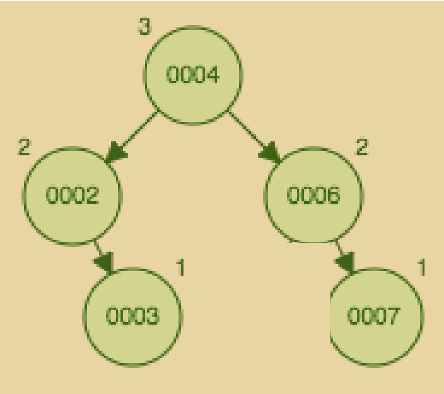


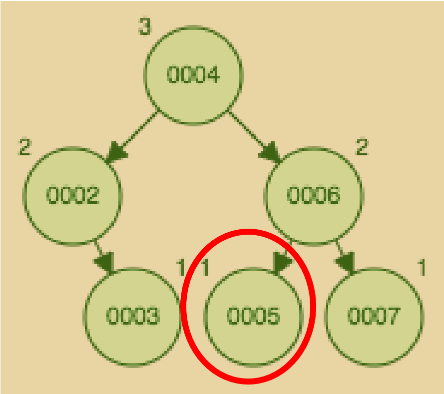
(III)

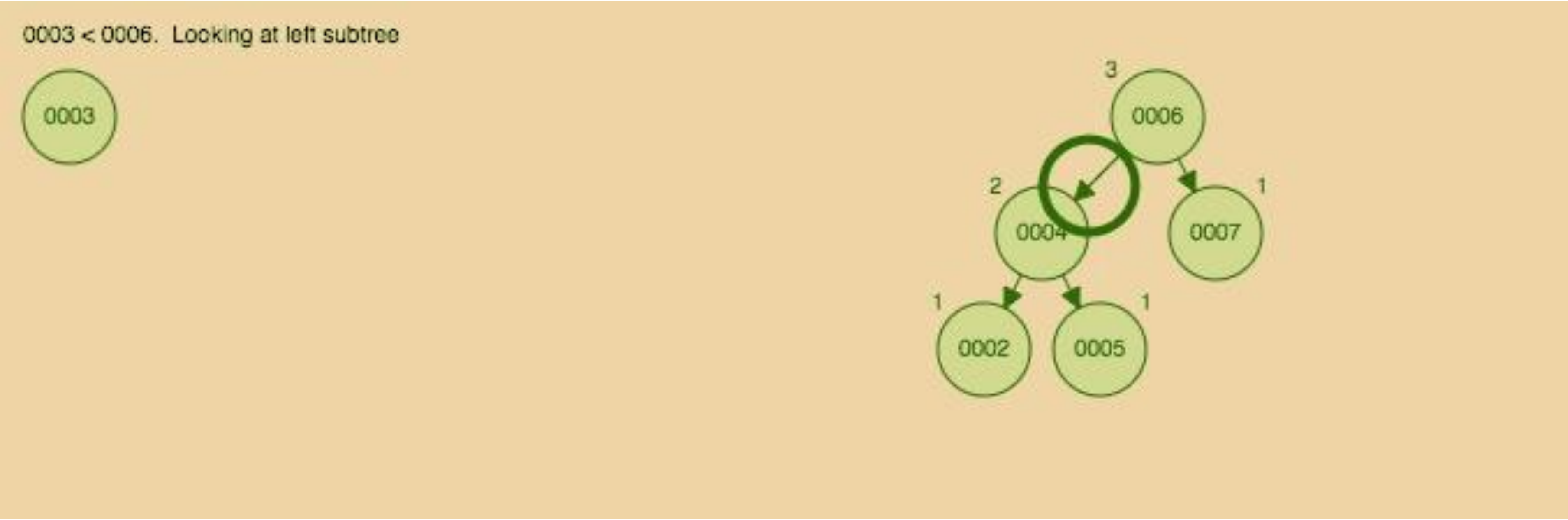
- **LL (Left-Left) --- III**



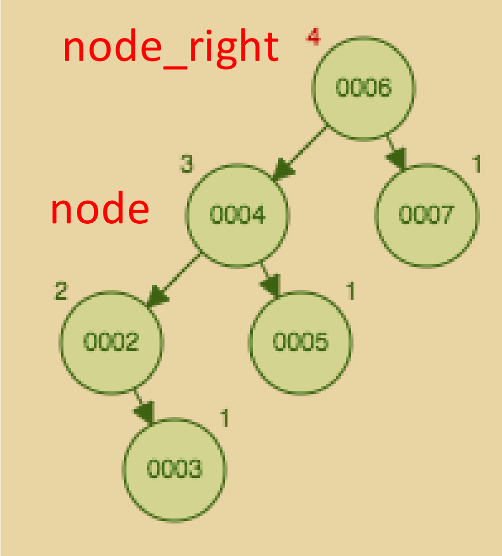
Take over





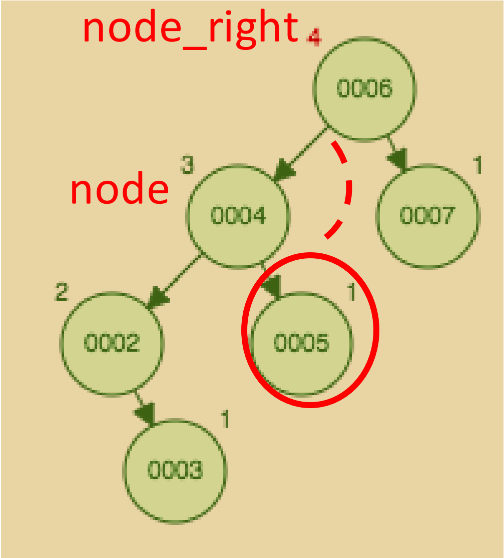


- **LL (Left-Left) --- Code**

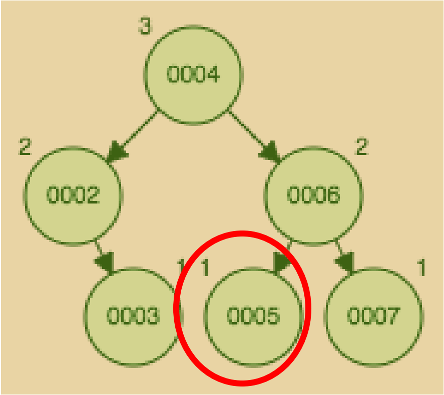



```python
"""LL"""
def right_rotate(self, node):
    node_right = node
    node = node.left
    node_right.left = node.right
    node.right = node_right

    self.update_height(node_right)
    self.update_height(node)

    return node
```

- **LL (Left-Left) --- Code**



Take over

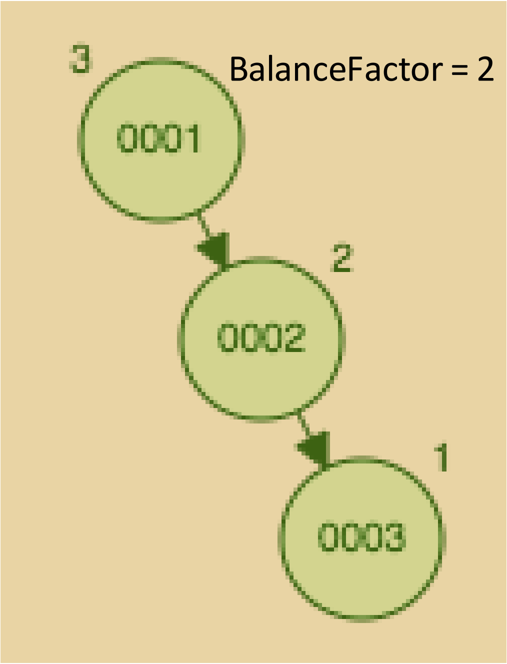



```python
"""LL"""
def right_rotate(self, node):
    node_right = node
    node = node.left
    node_right.left = node.right
    node.right = node_right

    self.update_height(node_right)
    self.update_height(node)

    return node
```

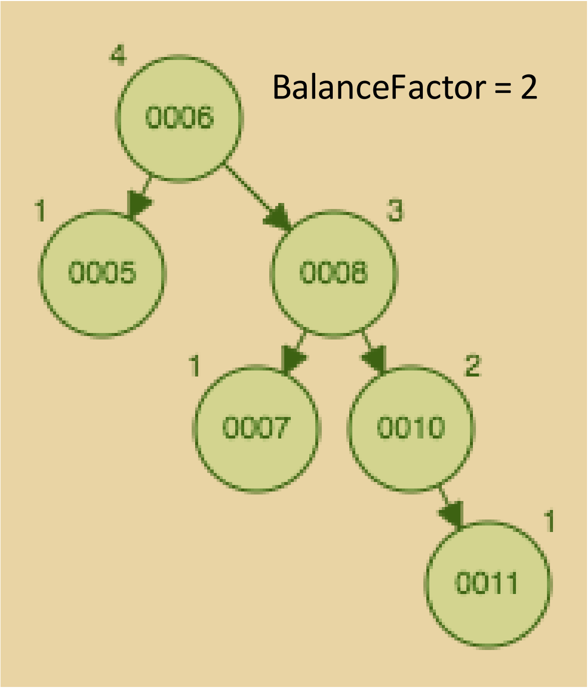
- ## RR (Right-Right)

$$Rotations \begin{cases} RR, & Left\ rot \\ LL, & Right\ rot \\ RL, & Right + Left\ rot \\ LR, & Left + Right\ rot \end{cases}$$

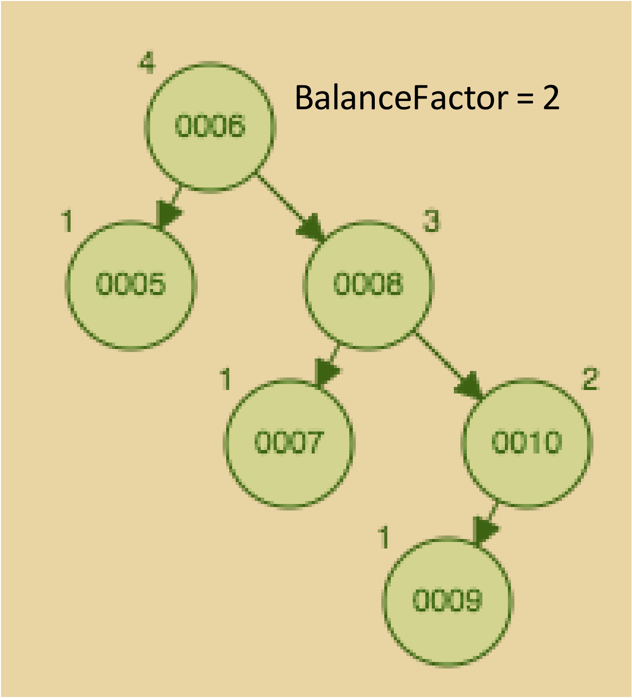
RR is the **MIRROR** case of LL



(I)



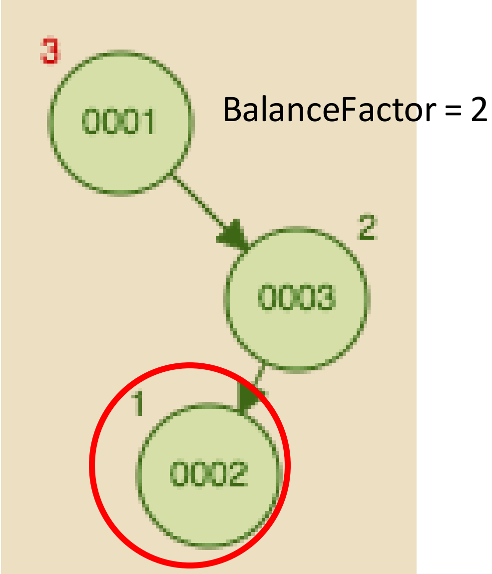
(II)



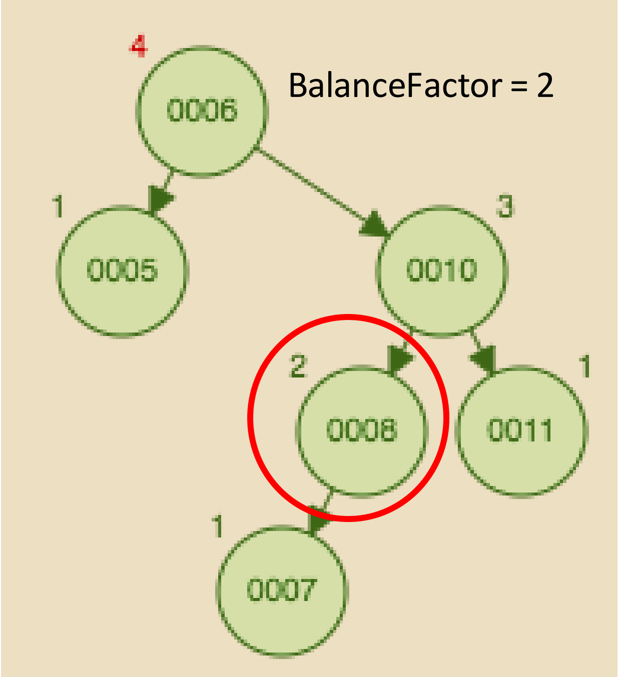
(III)

- **RL (Right-Left)**

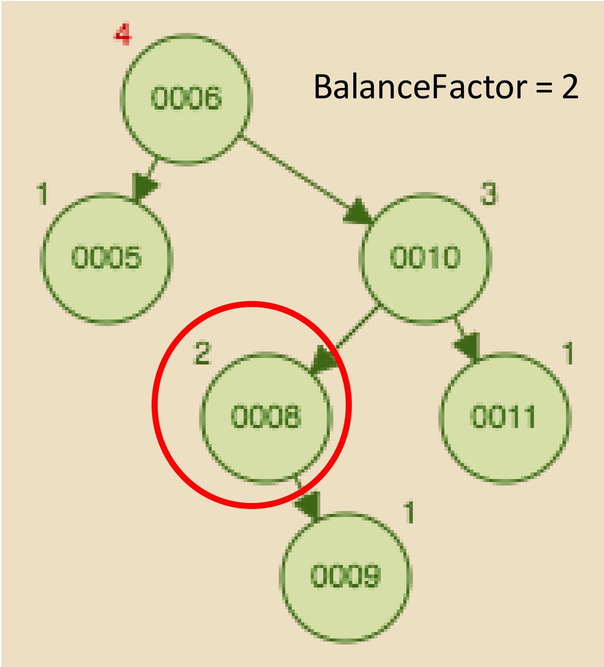
$$Rotations \begin{cases} RR, & Left\ rot \\ LL, & Right\ rot \\ RL, & Right + Left\ rot \\ LR, & Left + Right\ rot \end{cases}$$



(I)



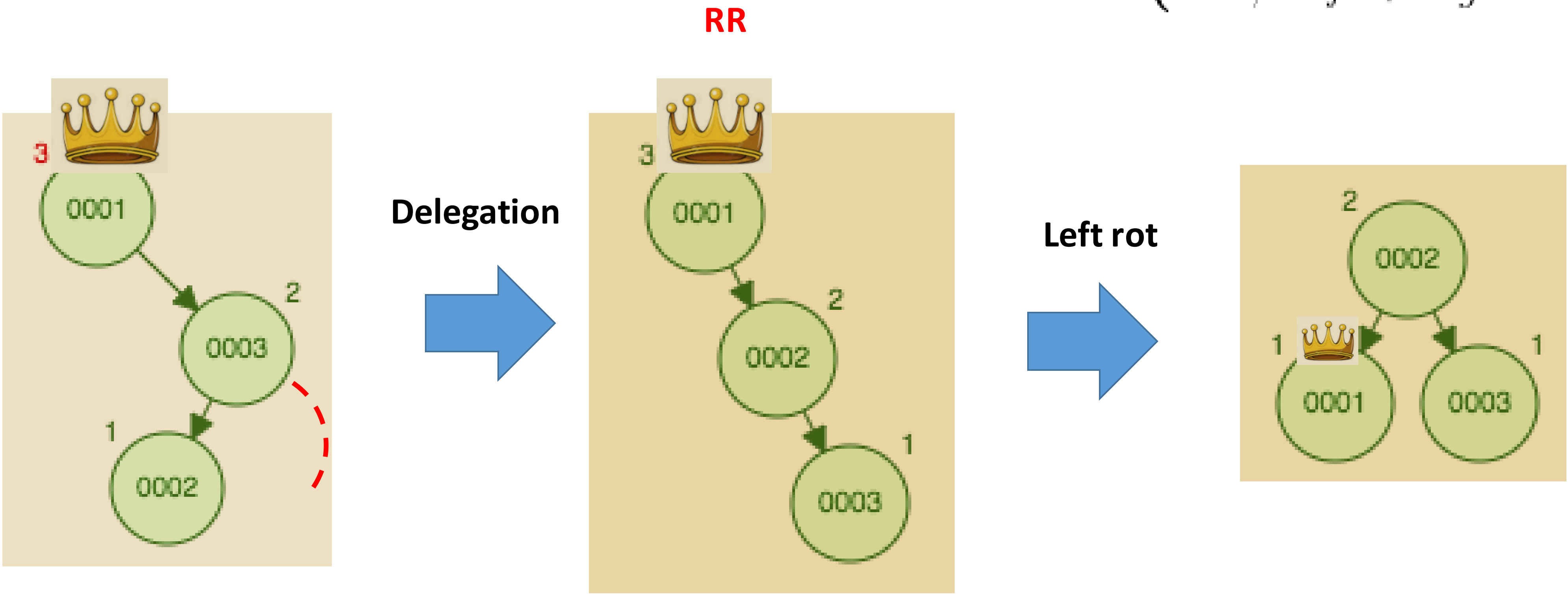
(II)



(III)

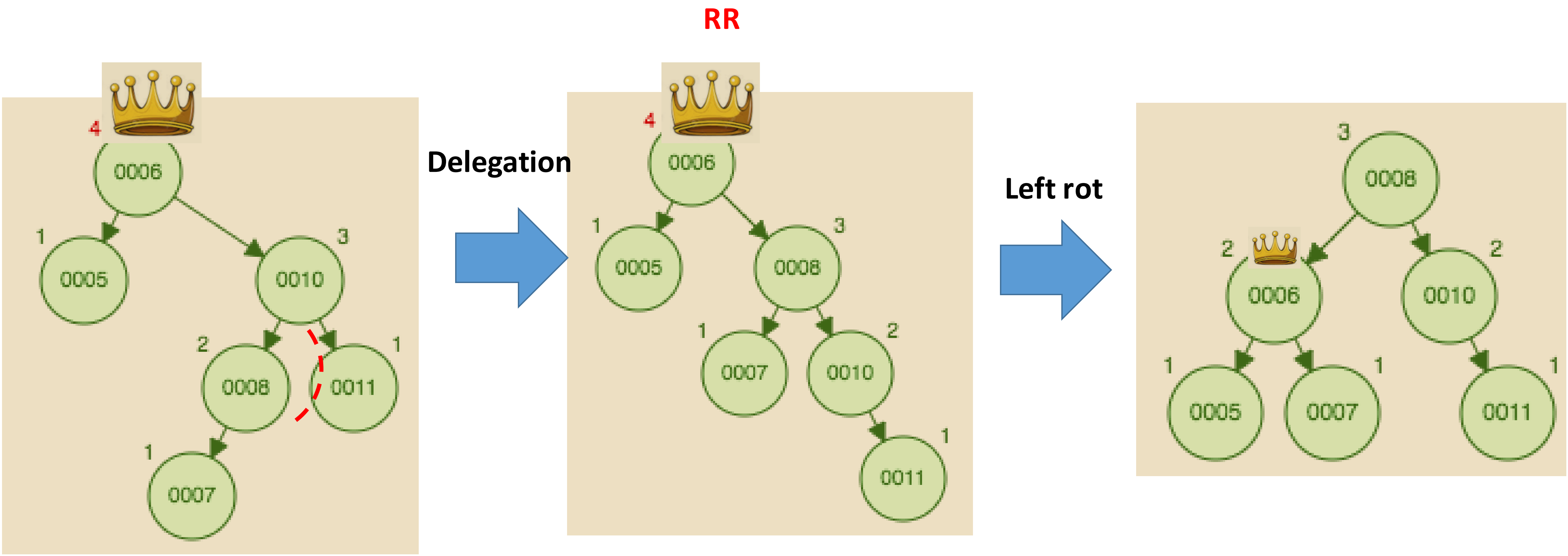
- **RL (Right-Left) --- I**

$$Rotations \begin{cases} RR, & Left\ rot \\ LL, & Right\ rot \\ RL, & Right + Left\ rot \\ LR, & Left + Right\ rot \end{cases}$$



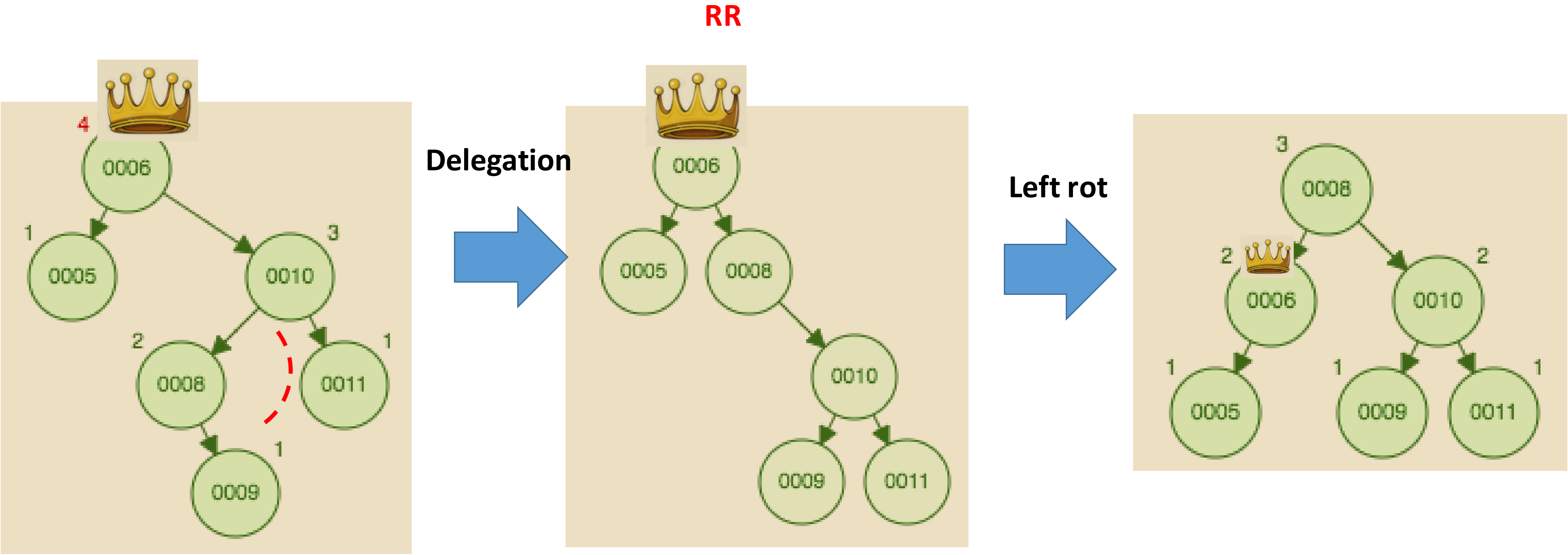
(I)

- **RL (Right-Left) --- II**



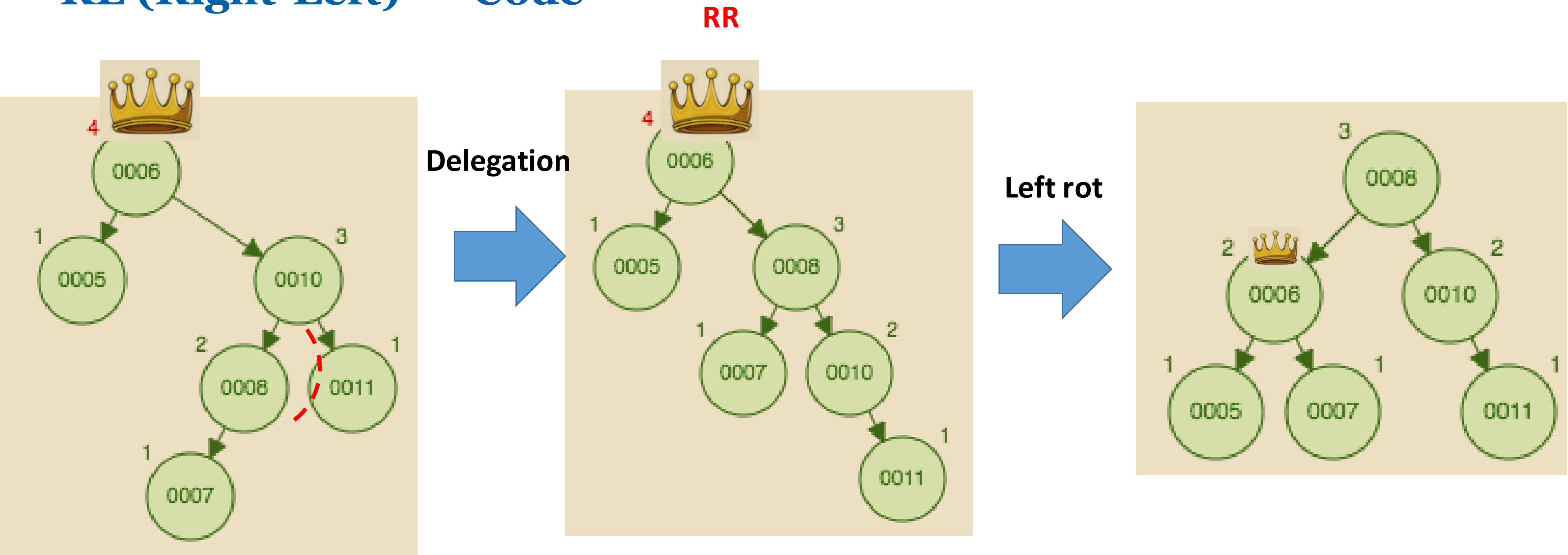
(II)

- **RL (Right-Left) --- III**



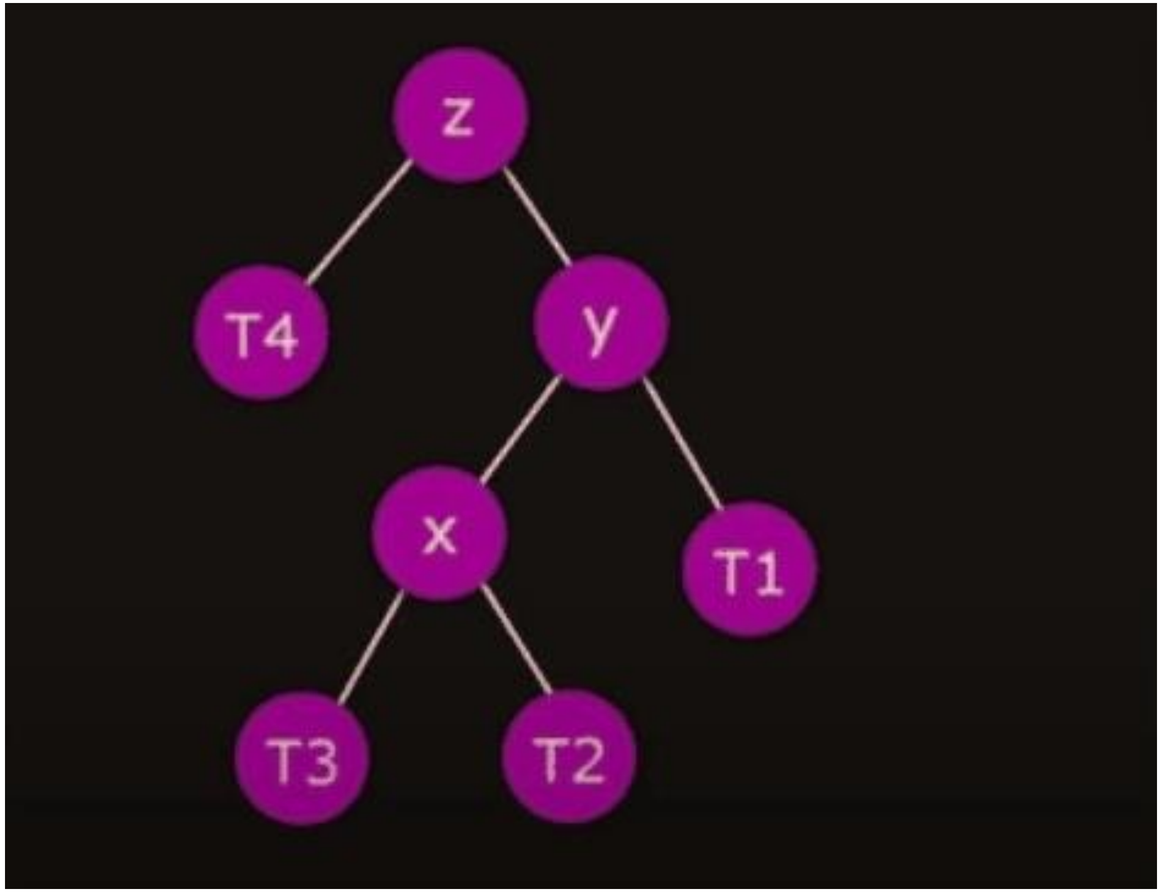
(III)

- **RL (Right-Left) --- Code**




```python
"""RL"""
def right_left_rotate(self, node):
    node.right = self.right_rotate(node.right)
    return self.left_rotate(node)
```

- **RL (Right-Left) --- Intuition**

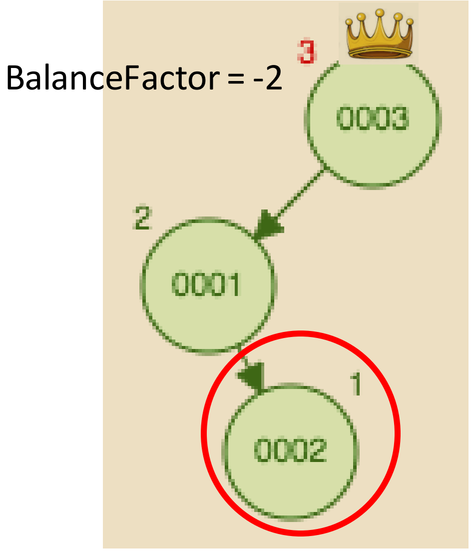


The gif is made from https://www.youtube.com/watch?v=ygZMI2YIcvk&feature=emb_title

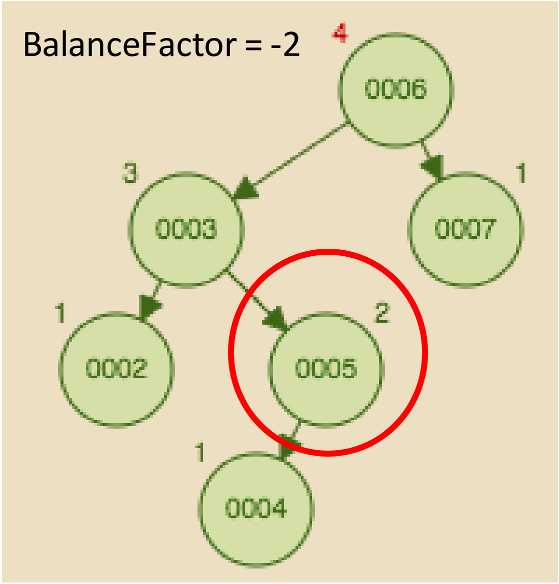
- ## LR (Left-Right)

LR is the **MIRROR** case of RL

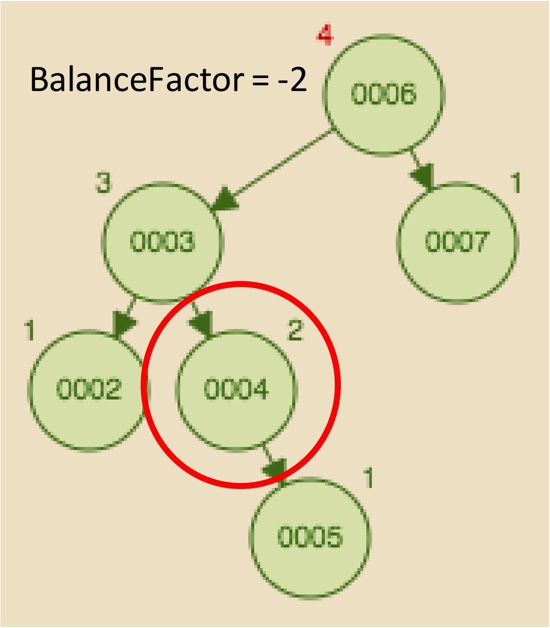
$$Rotations \begin{cases} RR, & Left\ rot \\ LL, & Right\ rot \\ RL, & Right + Left\ rot \\ LR, & Left + Right\ rot \end{cases}$$



(I)



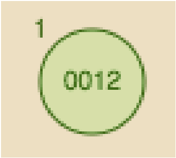
(II)



(III)

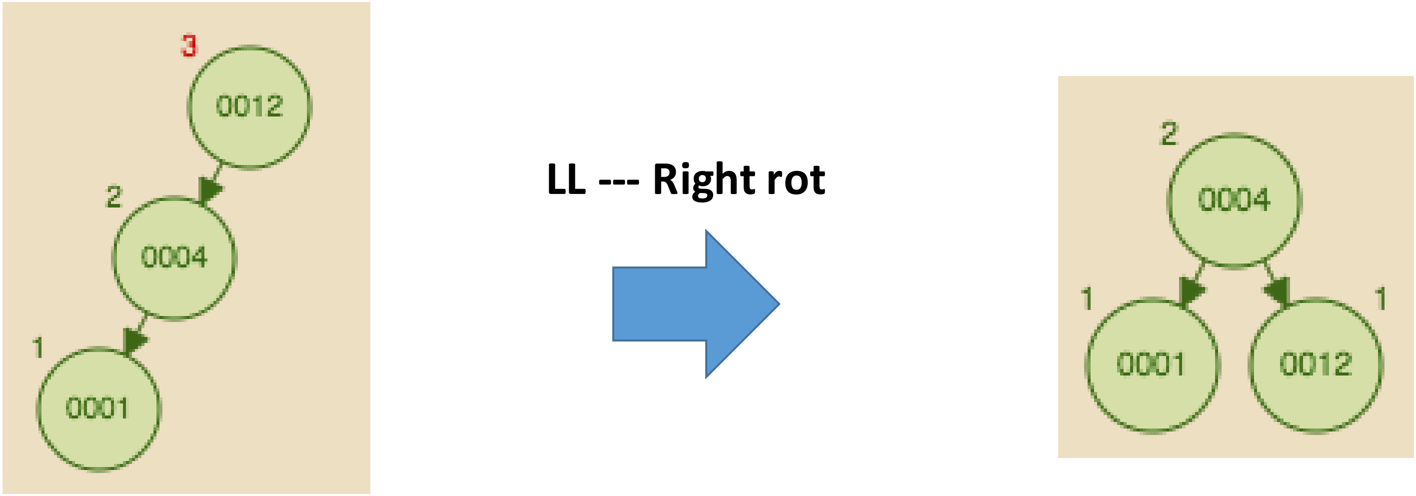
- ## Case Study --- insert

If we have a sequence {12, 4, 1, 3, 7, 8, 10, 9, 2, 11, 6, 5}



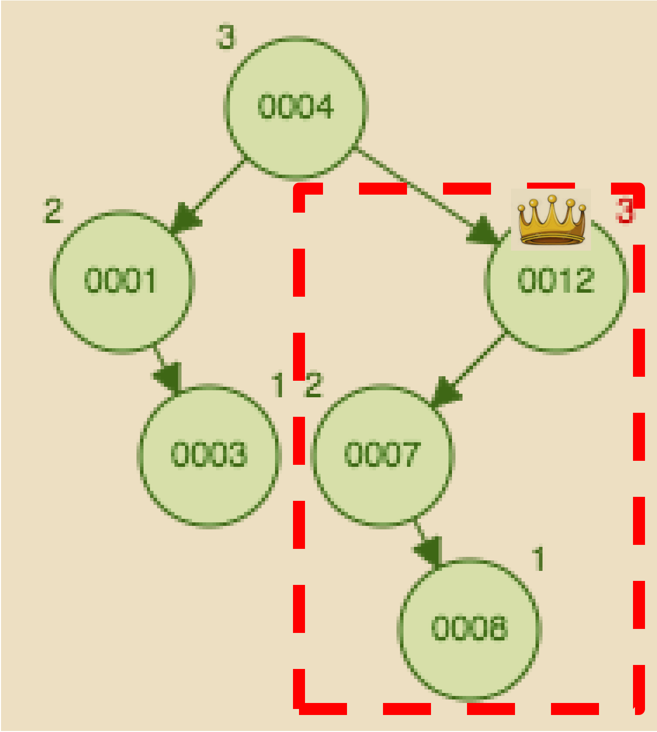
Case Study --- insert

If we have a sequence {~~12, 4, 1~~, 3, 7, 8, 10, 9, 2, 11, 6, 5}

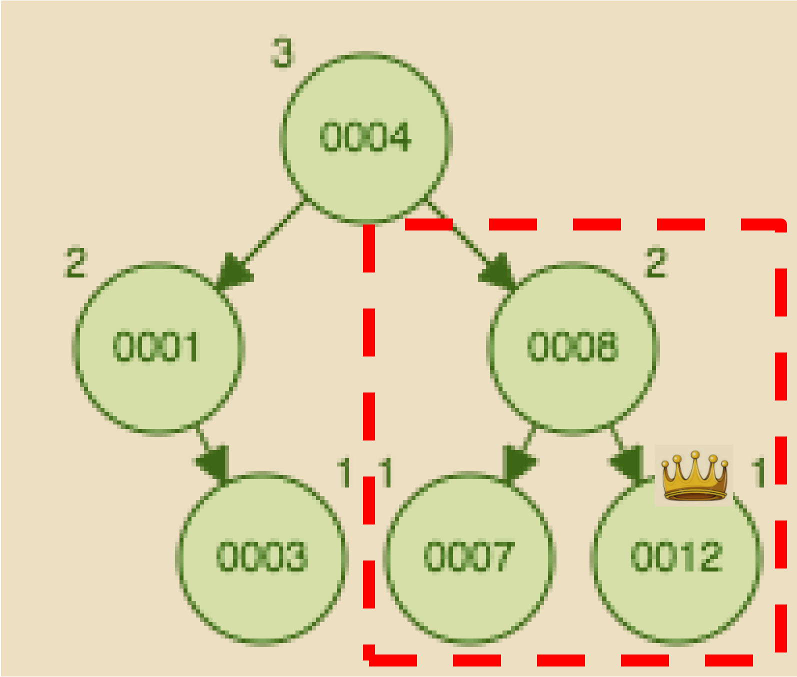


Case Study --- insert

If we have a sequence {~~12, 4, 1, 3, 7, 8~~, 10, 9, 2, 11, 6, 5}

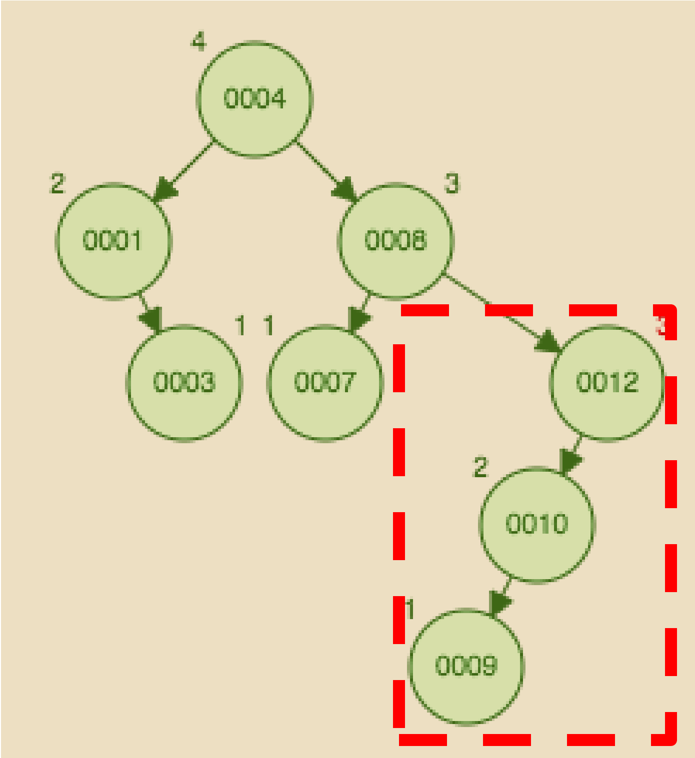


LR --- Left + Right rot

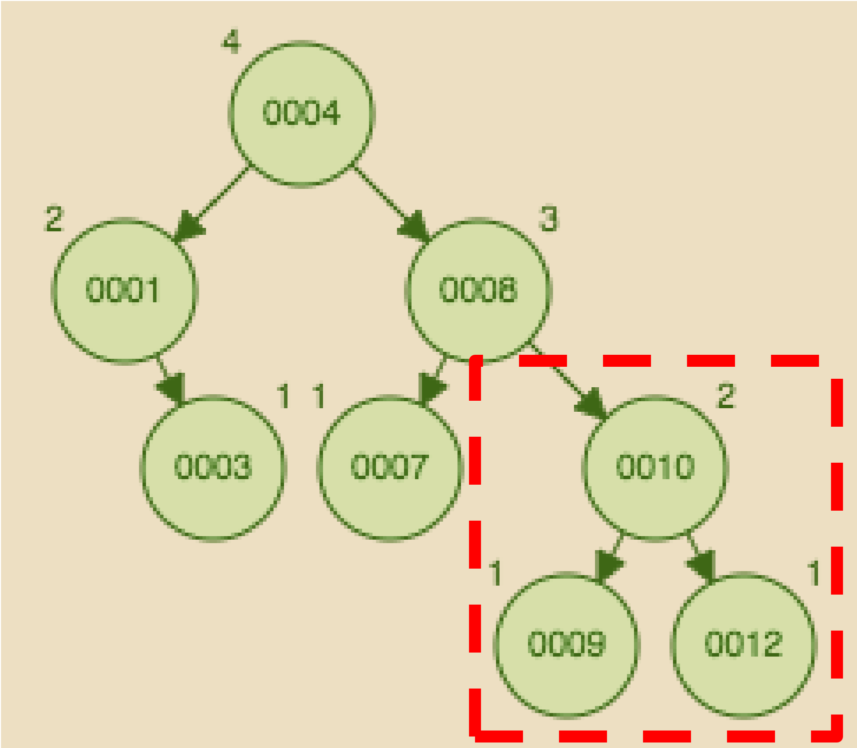


Case Study --- insert

If we have a sequence {~~12, 4, 1, 3, 7, 8, 10, 9~~, 2, 11, 6, 5}

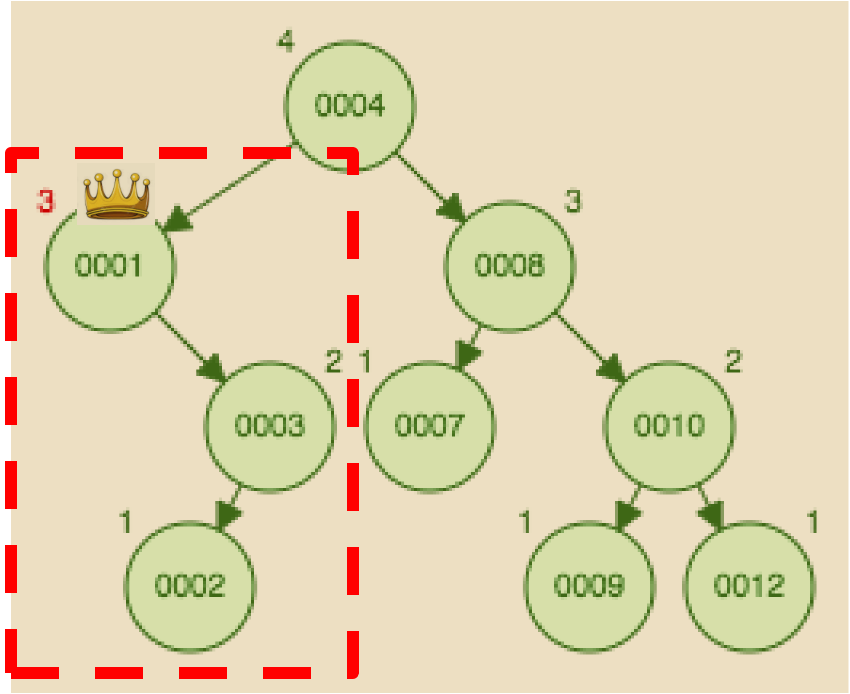


LL --- Right rot

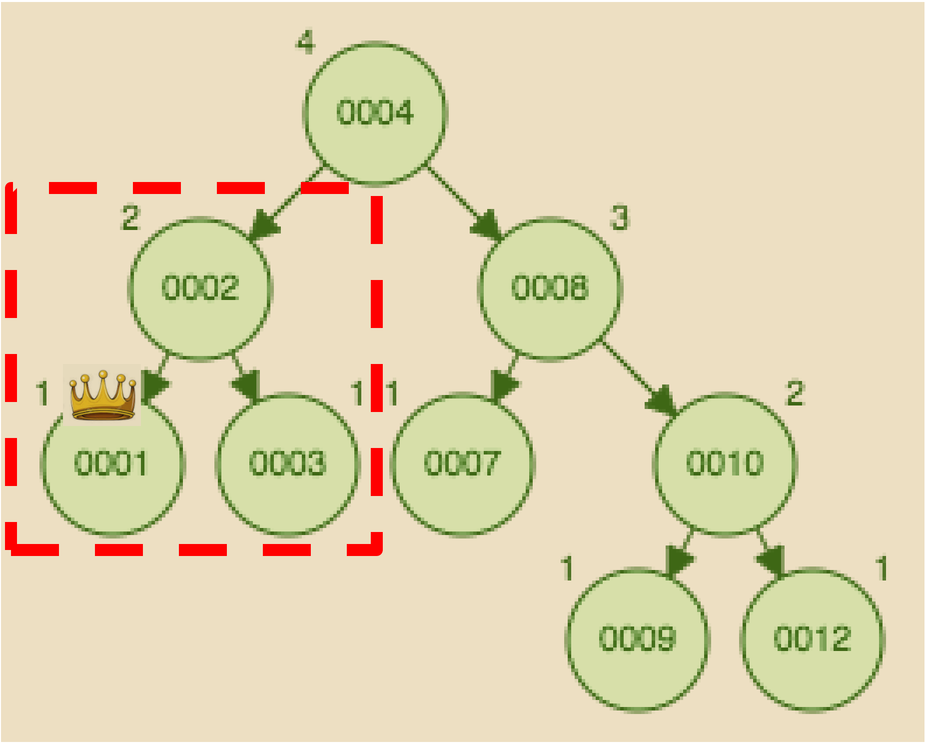


• Case Study --- insert

If we have a sequence {~~12, 4, 1, 3, 7, 8, 10, 9, 2~~, 11, 6, 5}

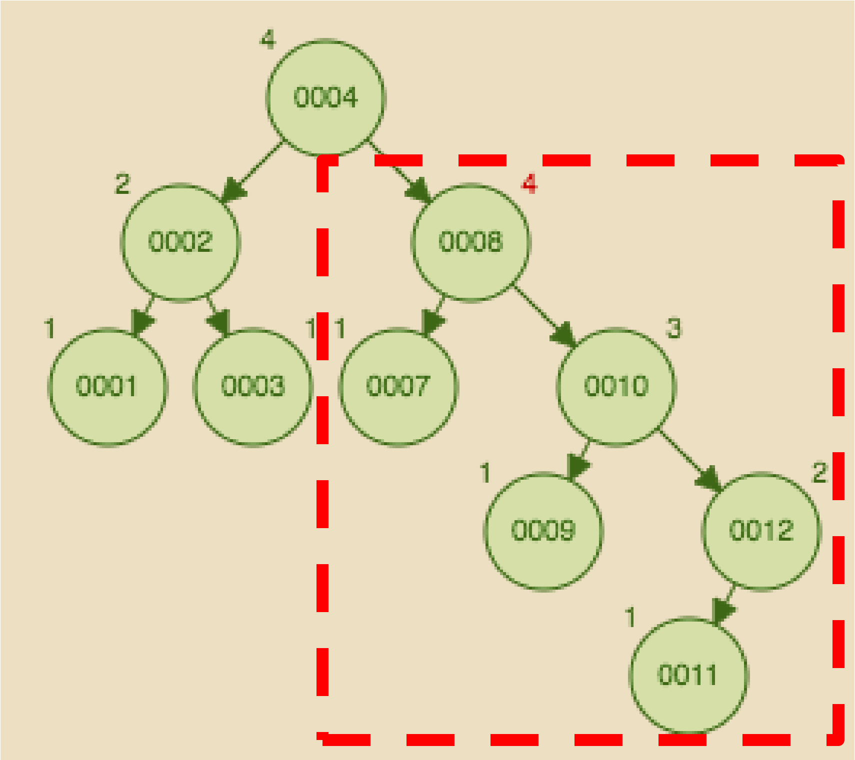


RL --- Right + Left rot

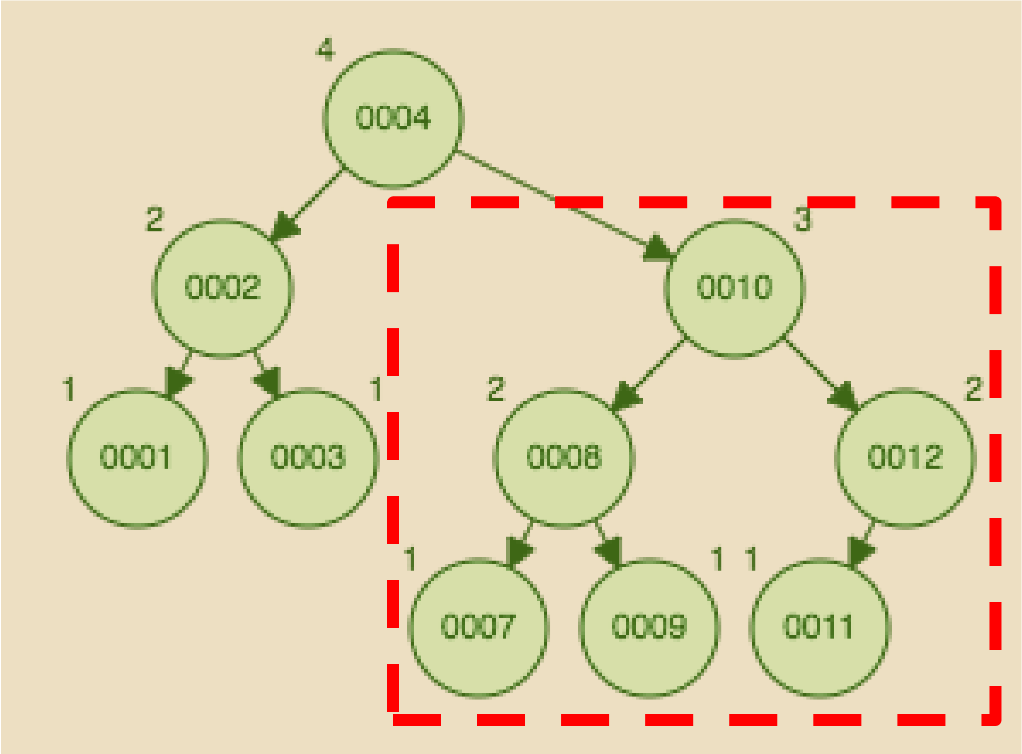


- # Case Study --- insert

If we have a sequence {~~12, 4, 1, 3, 7, 8, 10, 9, 2, 11~~, 6, 5}

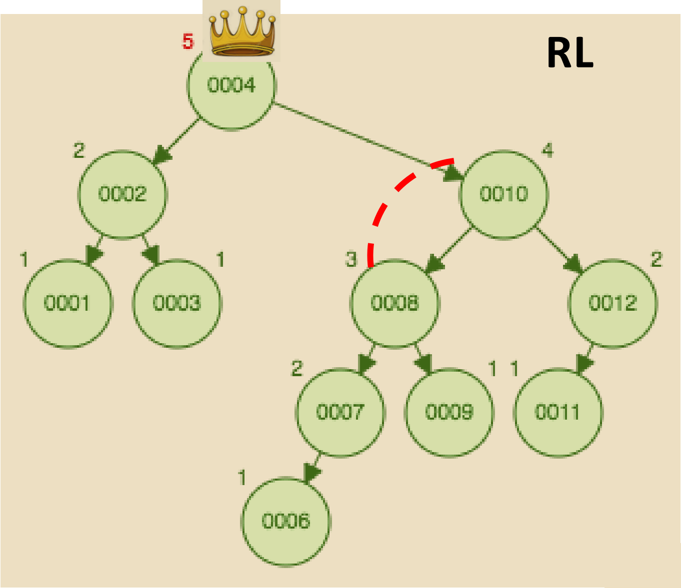


RR --- Left rot

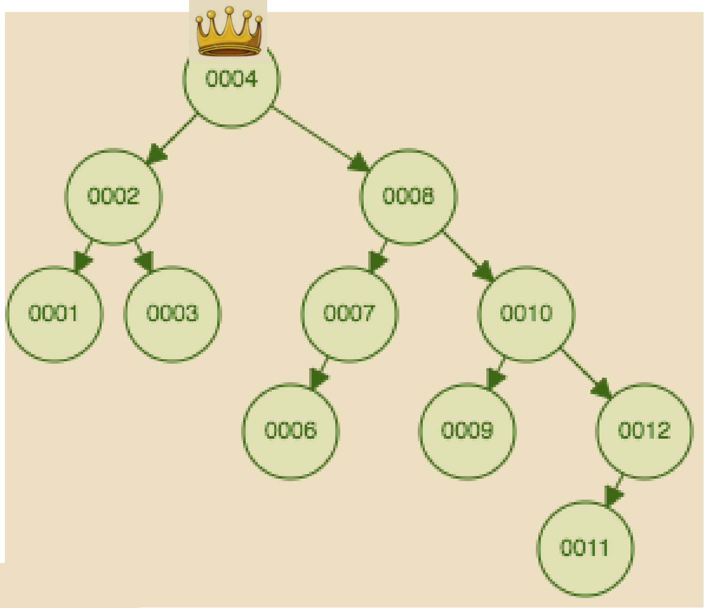


Case Study --- insert

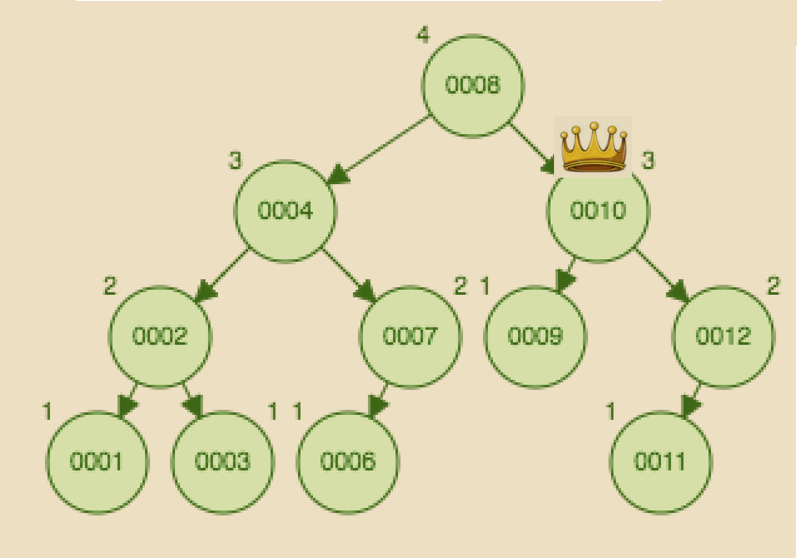
If we have a sequence {~~12, 4, 1, 3, 7, 8, 10, 9, 2, 11, 6~~, 5}



Delegation --- Right rot

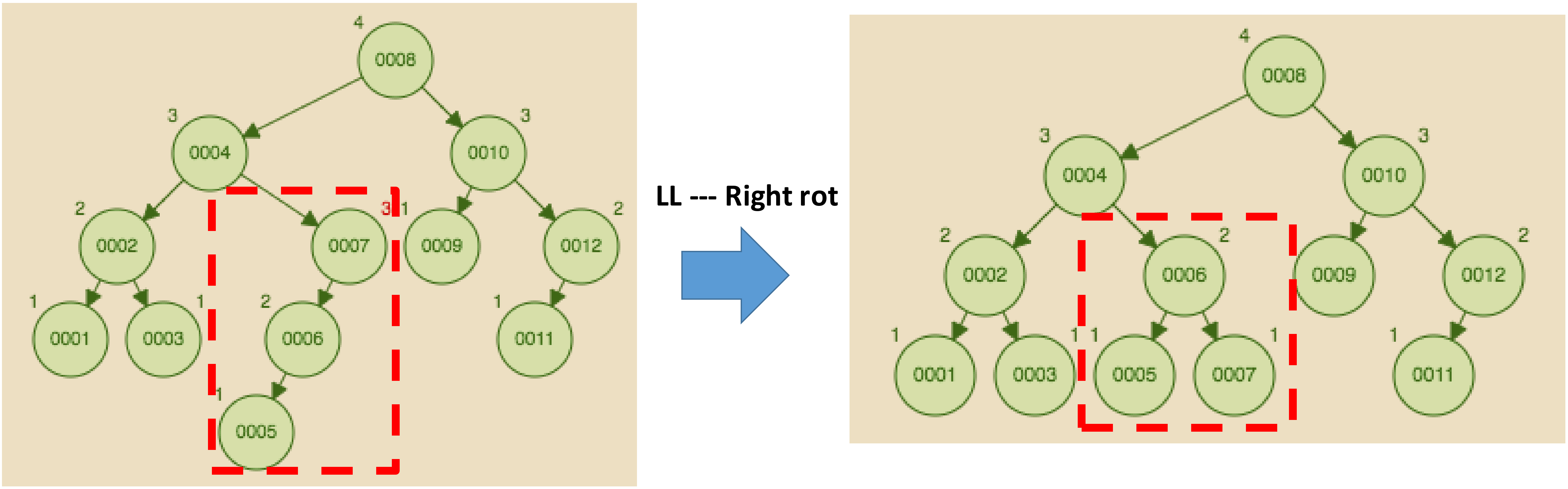


Left rot



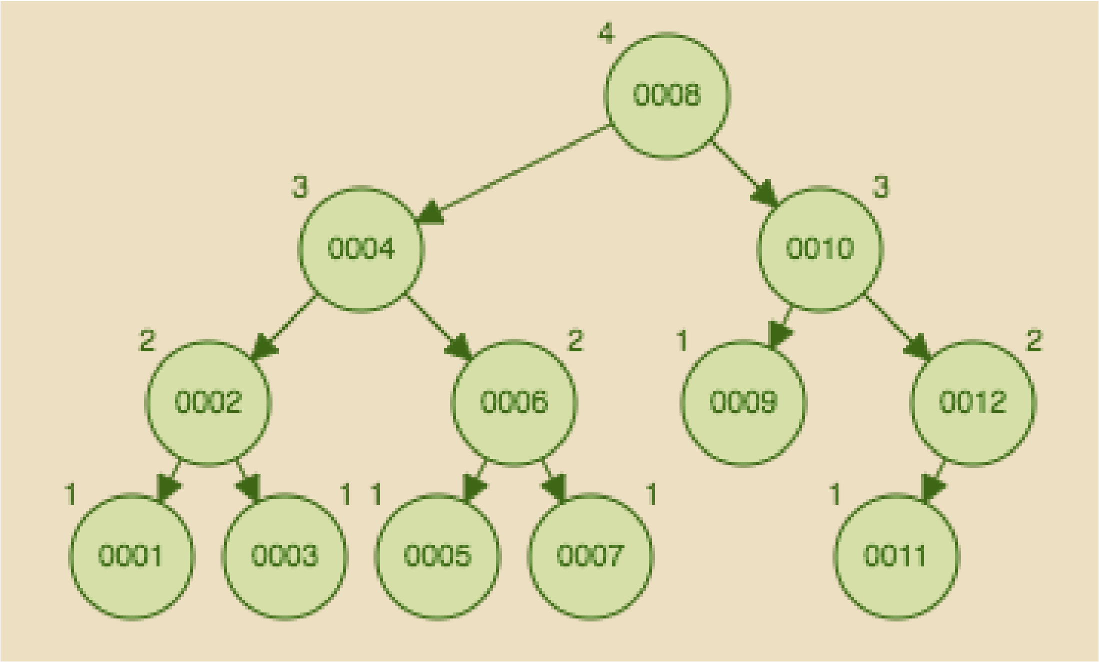
Case Study --- insert

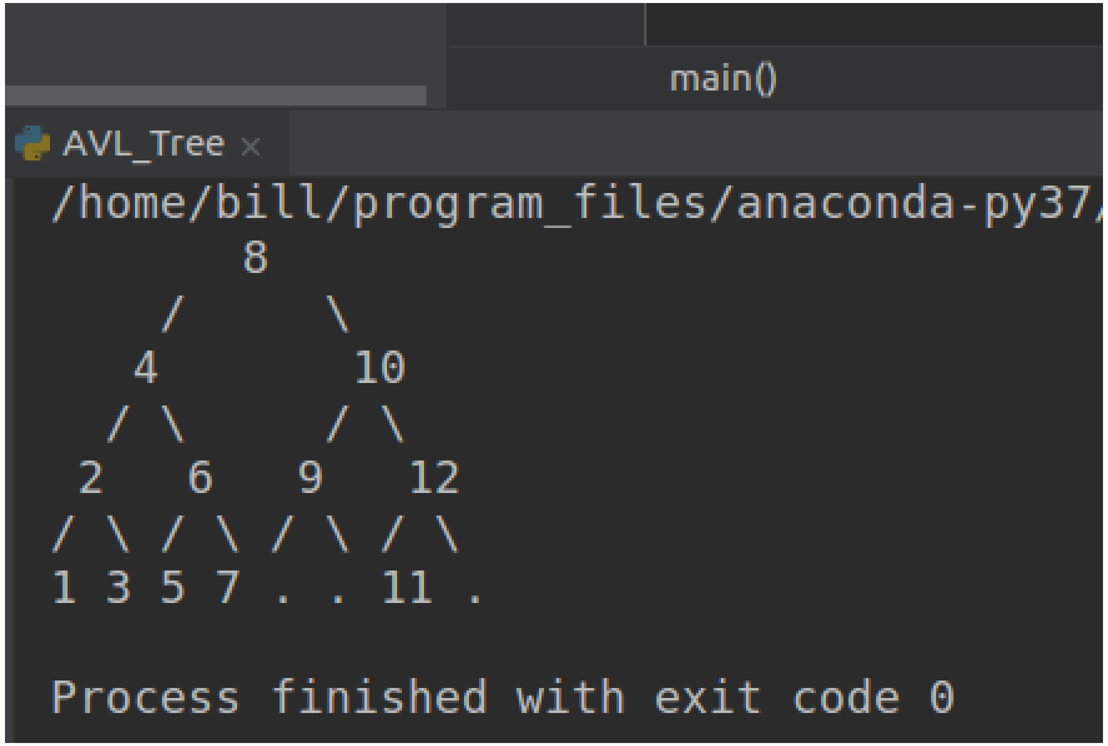
If we have a sequence {~~12, 4, 1, 3, 7, 8, 10, 9, 2, 11, 6, 5~~}



- # Case Study --- insert

If we have a sequence {12, 4, 1, 3, 7, 8, 10, 9, 2, 11, 6, 5}

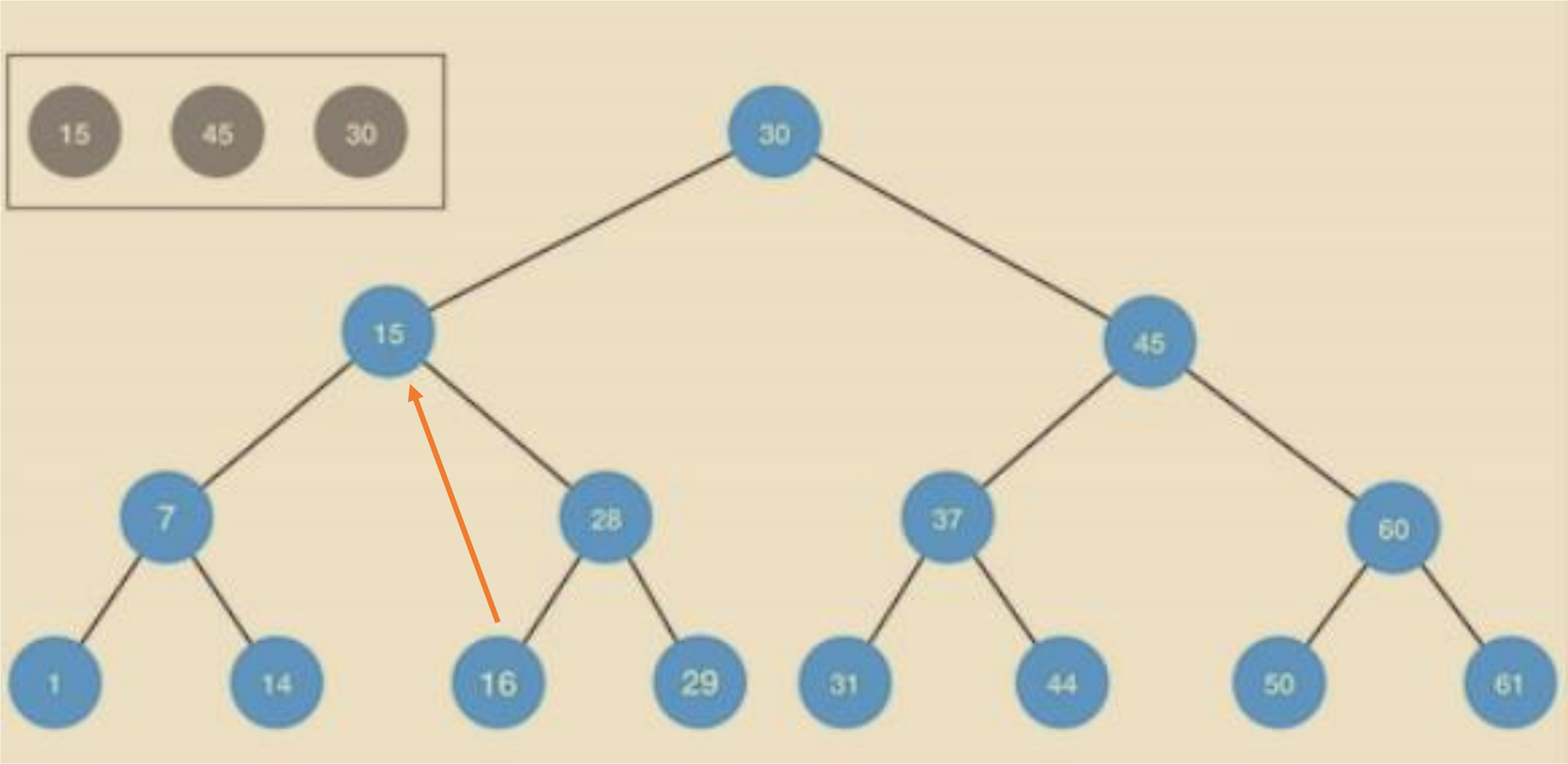




The code link will be put in the description section below

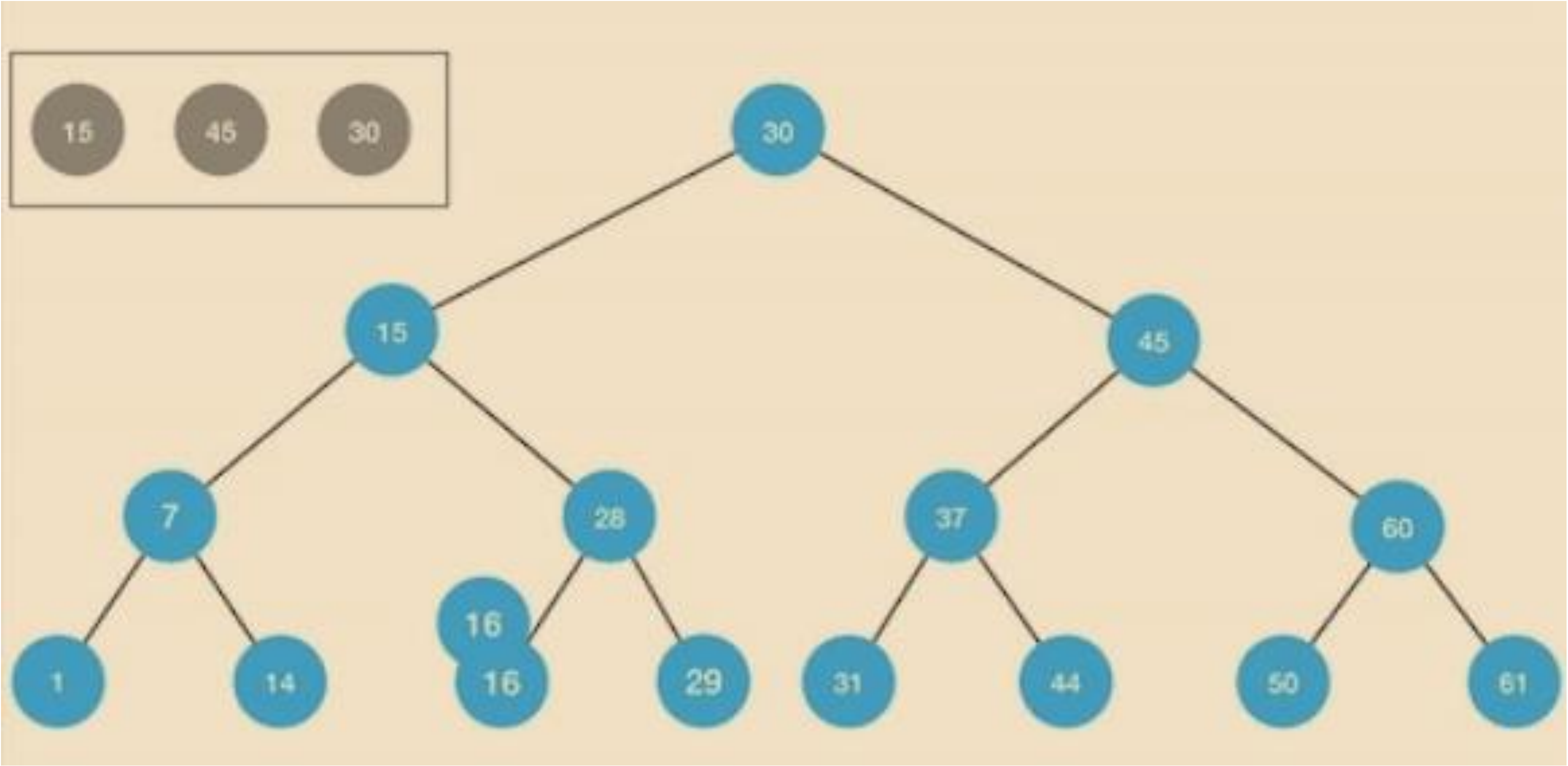
- ## AVL Tree --- delete

Standard BST deletion operation first. For example, if we would like to delete 15 from the tree below



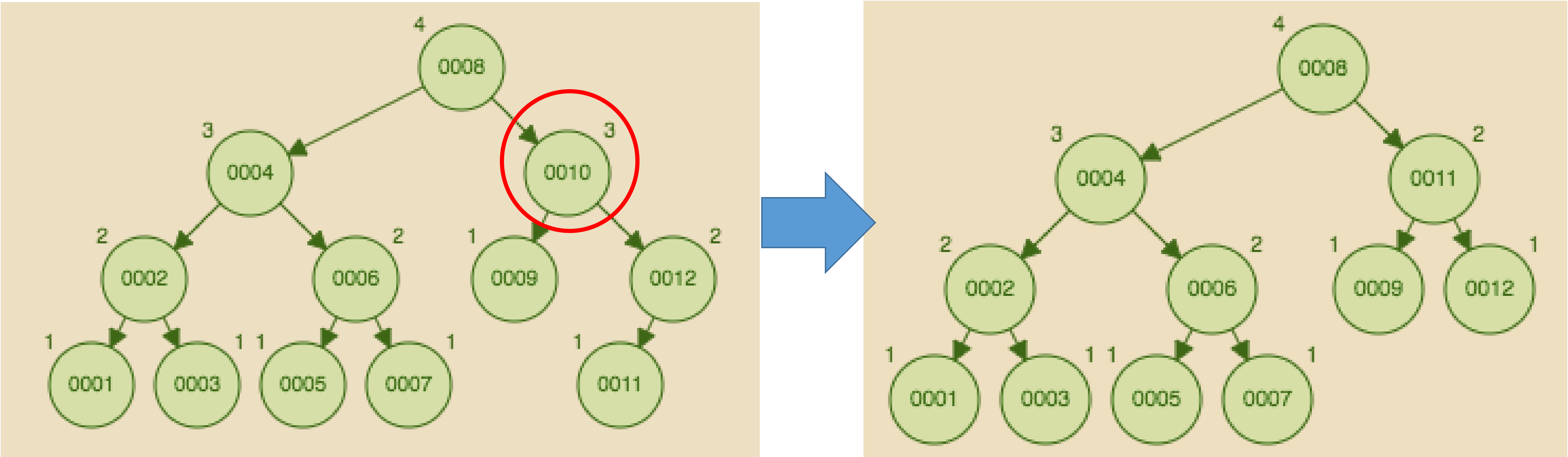
- ## AVL Tree --- delete

If we would like to delete 15 from the tree below



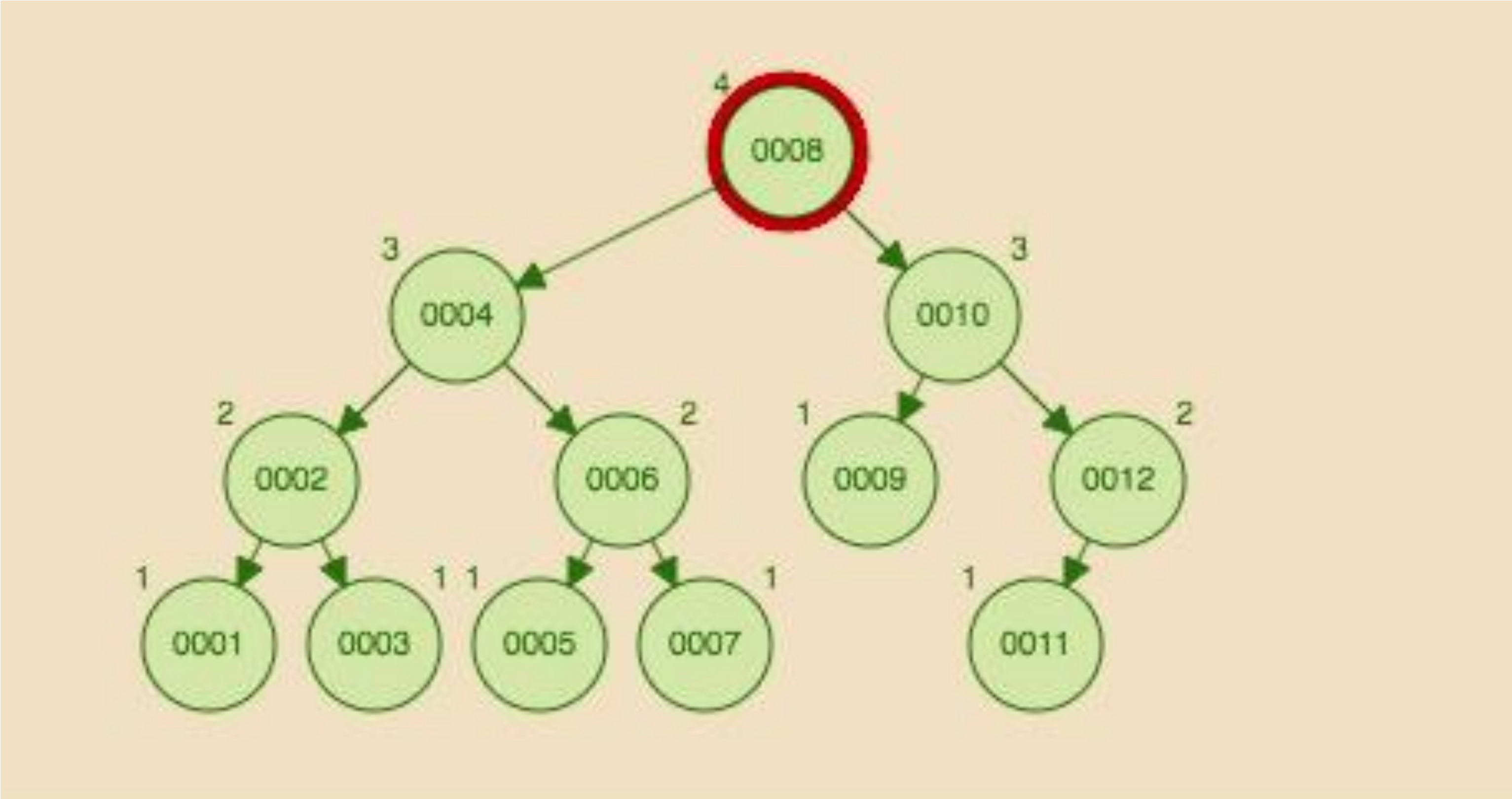
Case Study --- Delete

If we would like to delete 10 from the aforementioned tree



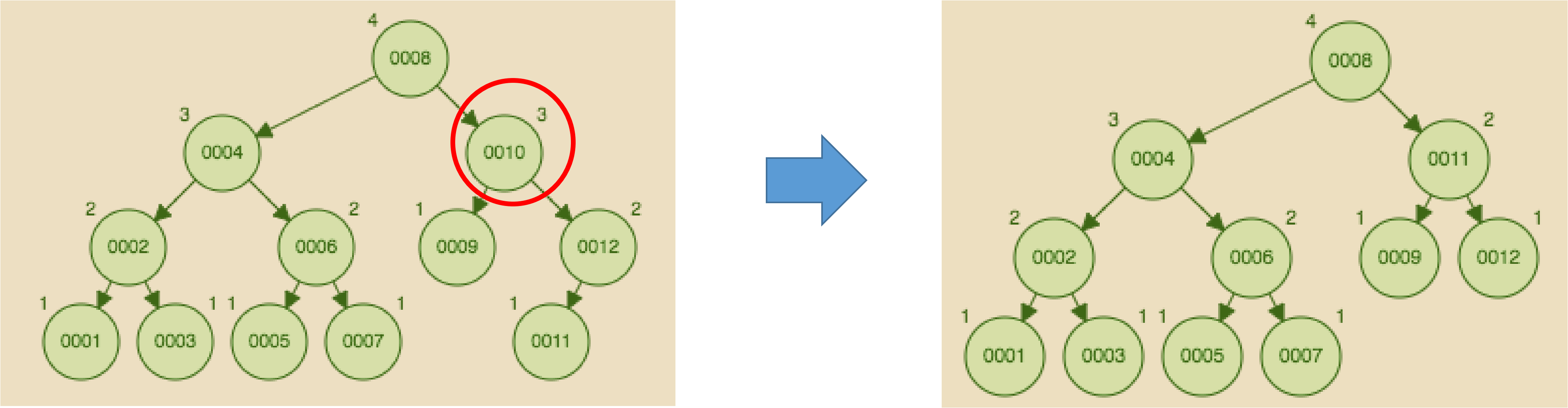
- ## Case Study --- Delete

If we would like to delete 10 from the aforementioned tree



- ## **Case Study --- Delete**

If we would like to delete 10 from the aforementioned tree




```
        8
     /     \
   4         11
  / \       / \
 2    6    9    12
/ \  / \  / \  / \
1  3  5  7  .  .  .  .
```

• AVL Tree --- complexity analysis

The height of AVL tree is O(logn), where n is the node number
Similar as Binary search

	Average	Worst case
height	O(logn)	
Search	O(logn)	O(logn)
Insert	O(logn)	O(logn)
Delete	O(logn)	O(logn)

Thank you!