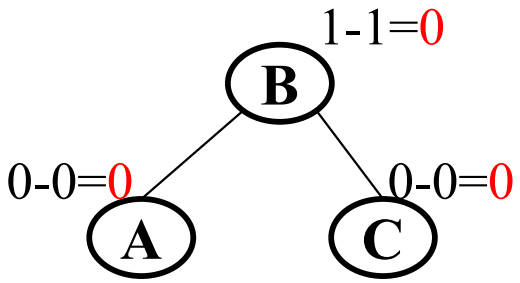
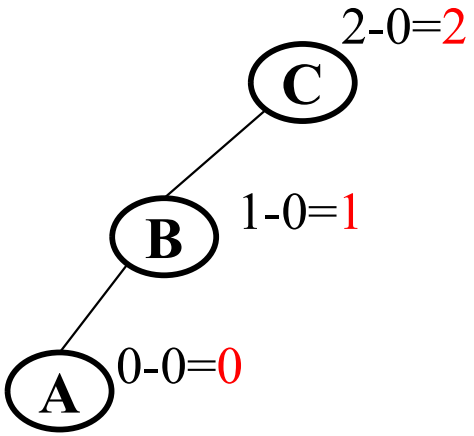
AVL Tree

- It is a **height balanced binary search tree**
- Each node is associated with a balance factor
- **Balance Factor of a node**

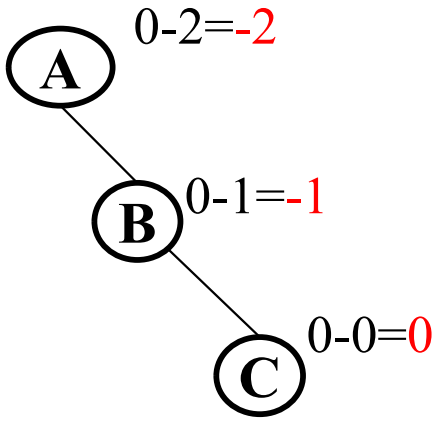
  = height of left subtree – height of right subtree
- In an AVL tree balance factor of every node is **-1,0 or +1**
- Otherwise the tree will be unbalanced and need to be balanced.



Tree is Balanced.



Tree is not Balanced.
Balance factor of C is 2



Tree is not Balanced.
Balance factor of A is -2

Why AVL Tree?

- Most of the Binary Search Tree operations (eg: search, insertion, deletion etc) take O(h) time where h is the height of the BST
- The minimum height of the BST is log n

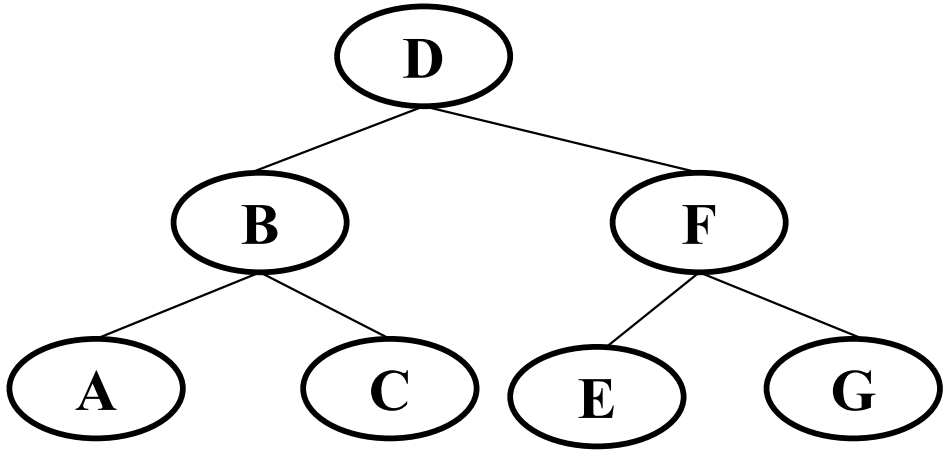
- The height of an AVL tree is always O(log n) where n is the number of nodes in the tree.

- So the time complexity of all AVL tree operations are **O(log n)**

Find minimum and maximum height of any AVL tree with 7 nodes? Assume that the height of a tree with a single node is 0

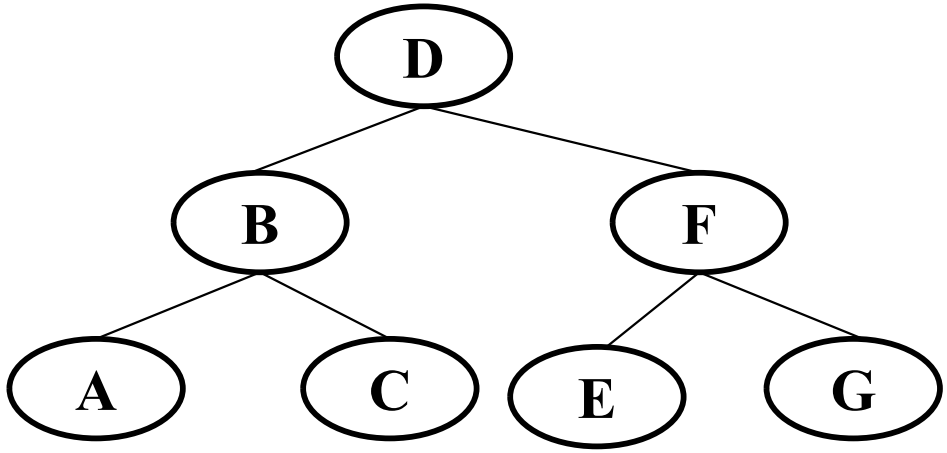
Find minimum and maximum height of any AVL tree with 7 nodes? Assume that the height of a tree with a single node is 0

- Example of minimum height AVL Tree with 7 nodes



Find minimum and maximum height of any AVL tree with 7 nodes? Assume that the height of a tree with a single node is 0

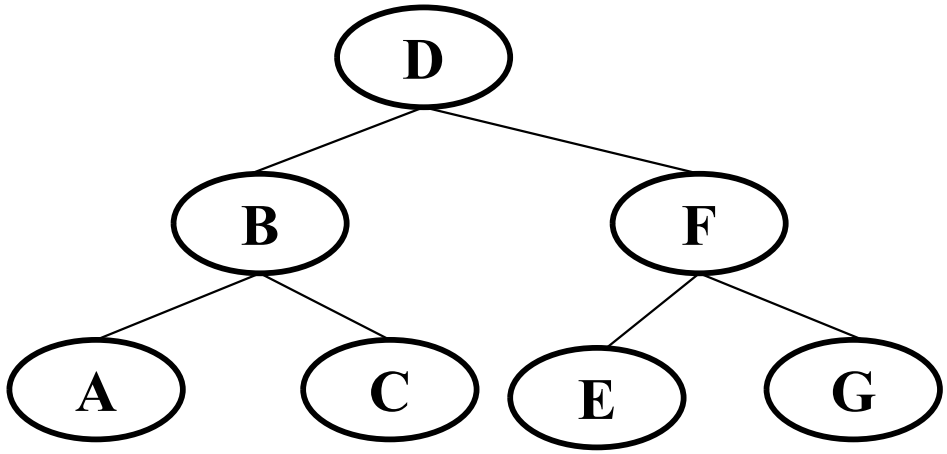
- Example of minimum height AVL Tree with 7 nodes



Height = 2

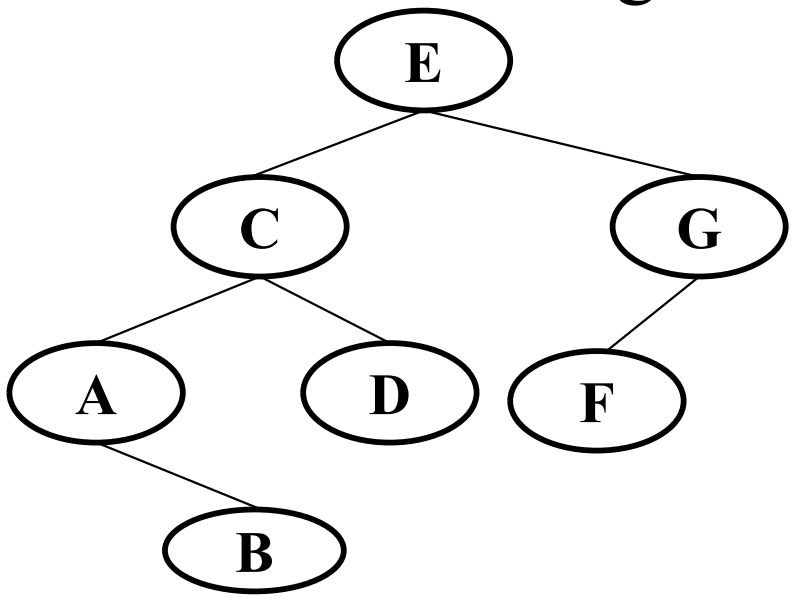
Find minimum and maximum height of any AVL tree with 7 nodes? Assume that the height of a tree with a single node is 0

- Example of minimum height AVL Tree with 7 nodes



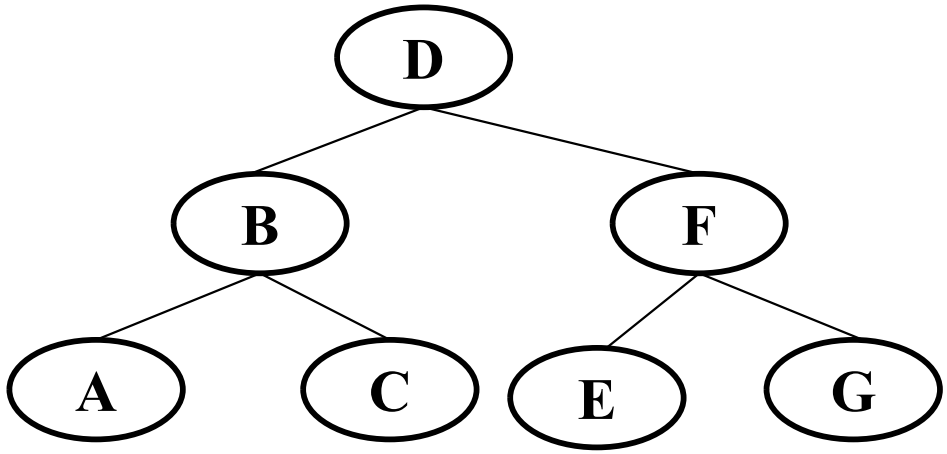
Height = 2

- Example of maximum height AVL Tree with 7 nodes



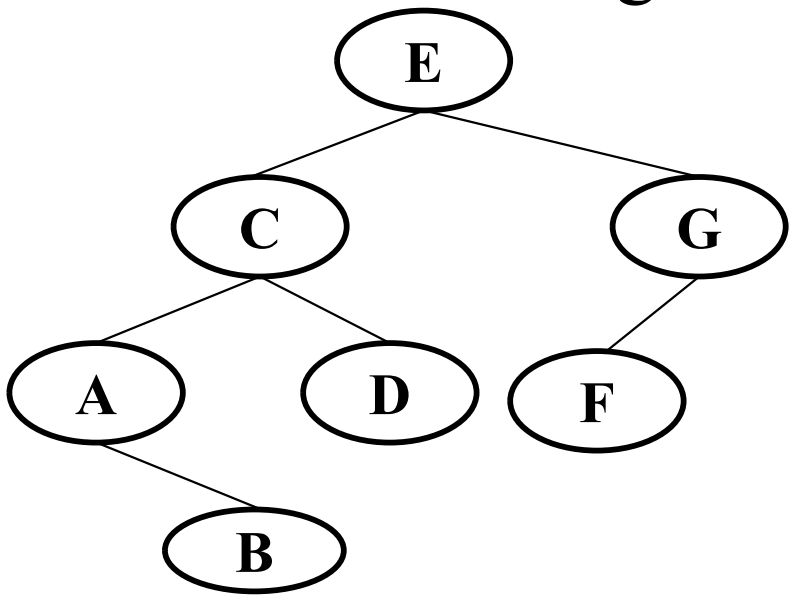
Find minimum and maximum height of any AVL tree with 7 nodes? Assume that the height of a tree with a single node is 0

- Example of minimum height AVL Tree with 7 nodes



Height = 2

- Example of maximum height AVL Tree with 7 nodes

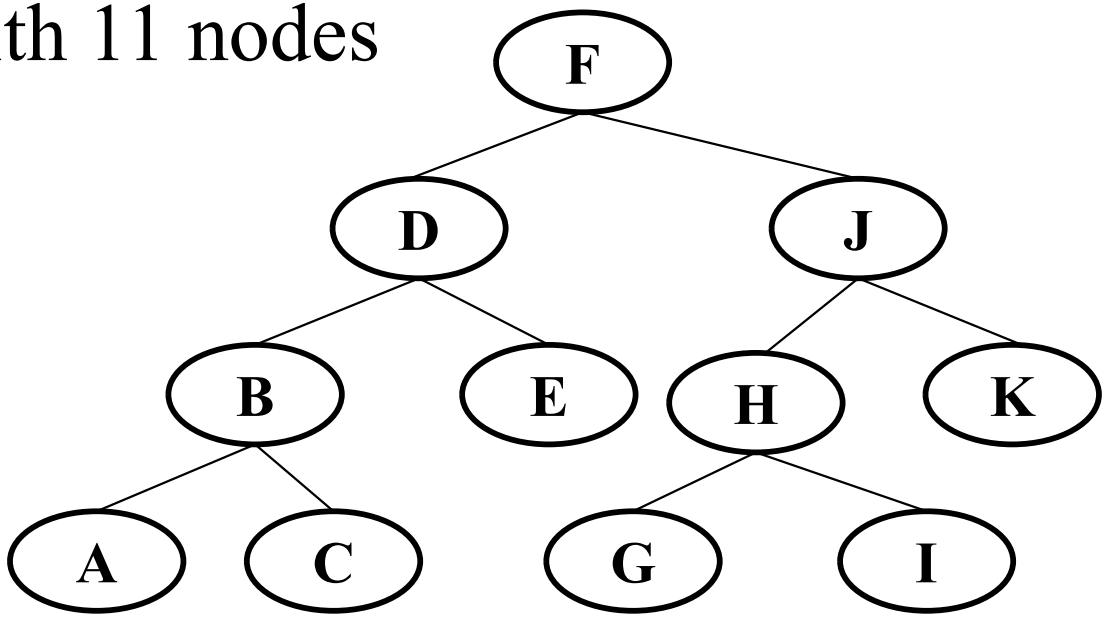


Height = 3

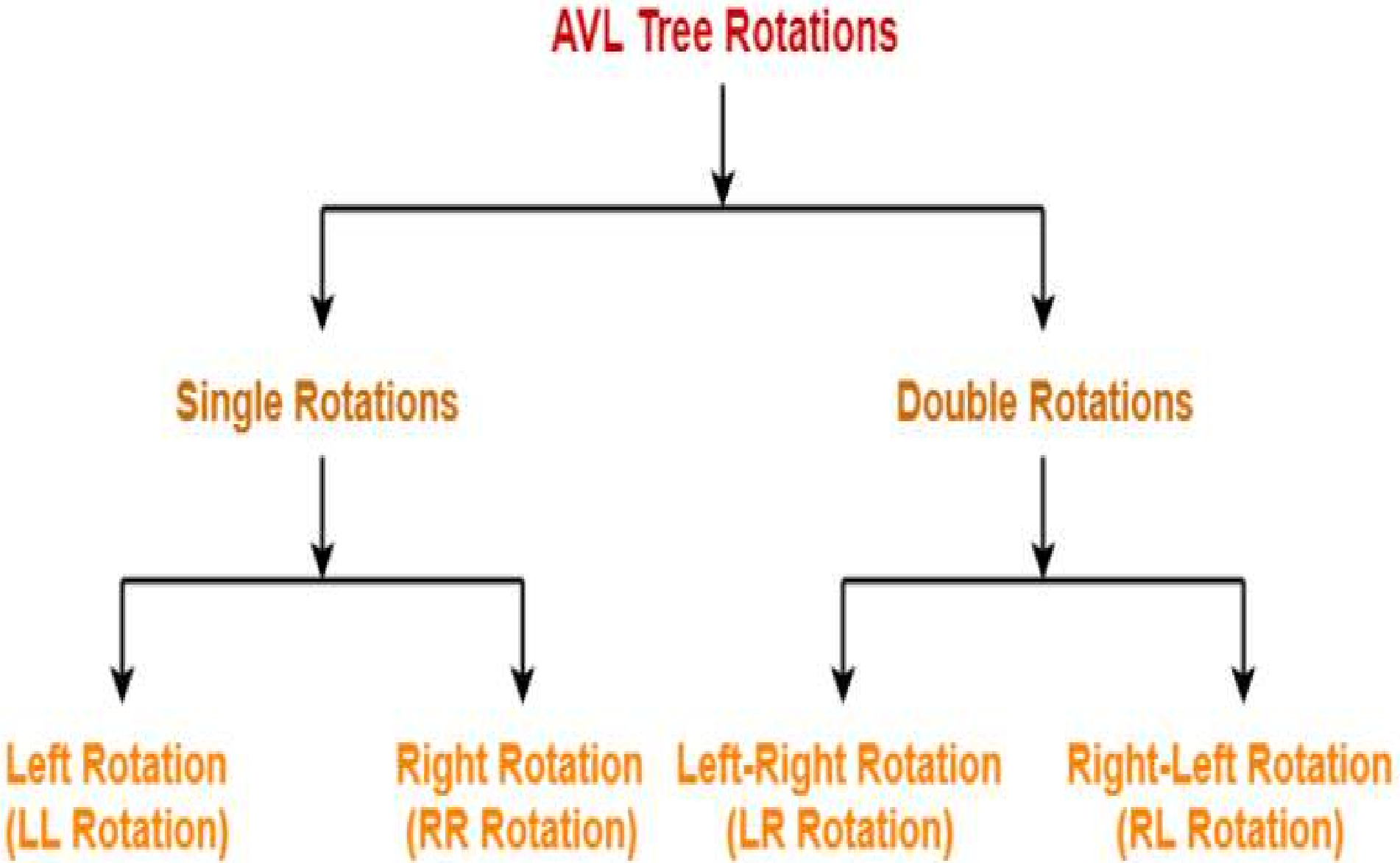
Find minimum and maximum height of any AVL tree with 11 nodes? Assume that the height of a tree with a single node is 0

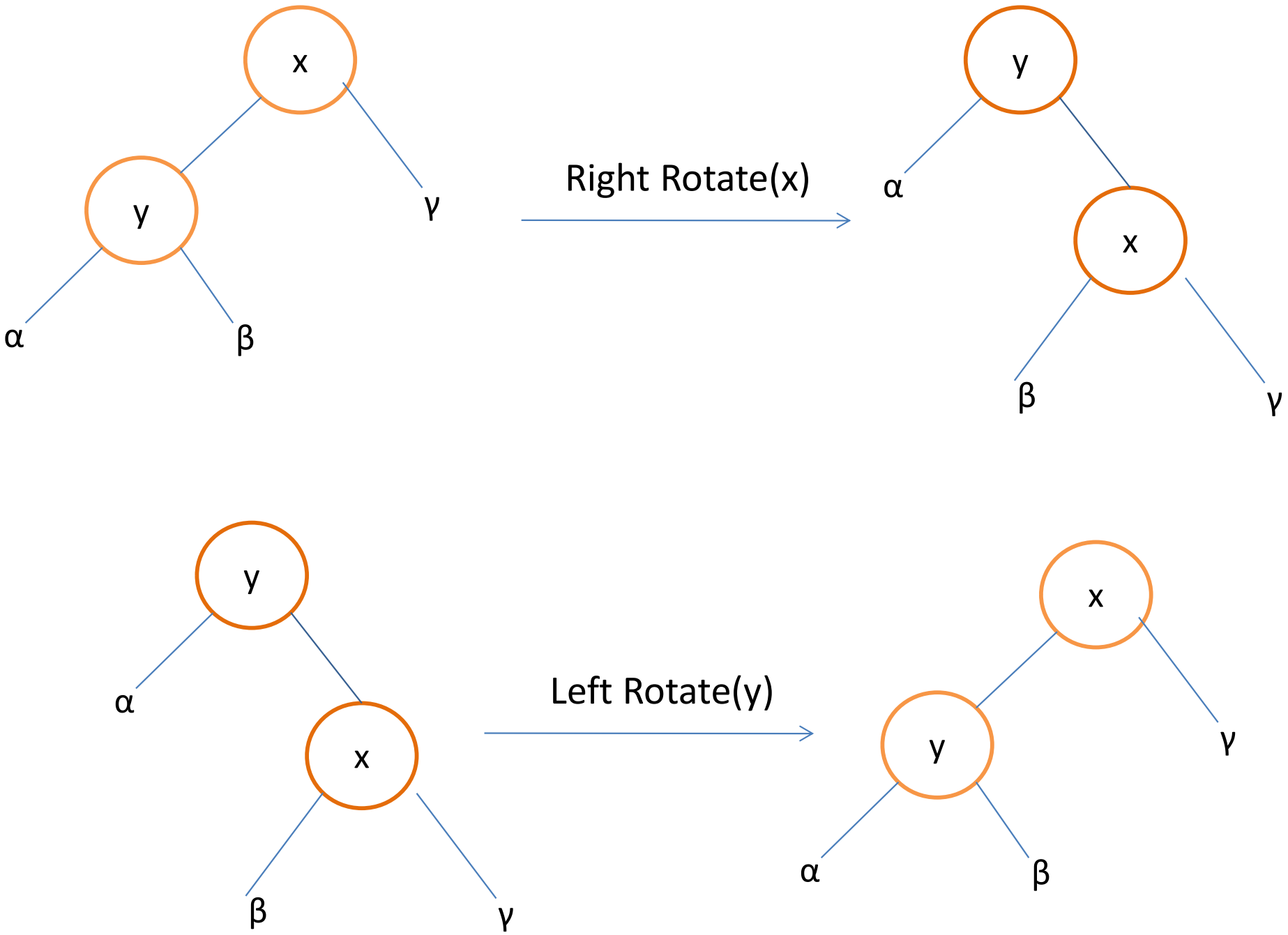
Find minimum and maximum height of any AVL tree with 11 nodes? Assume that the height of a tree with a single node is 0

- Example of minimum and maximum height AVL Tree with 11 nodes

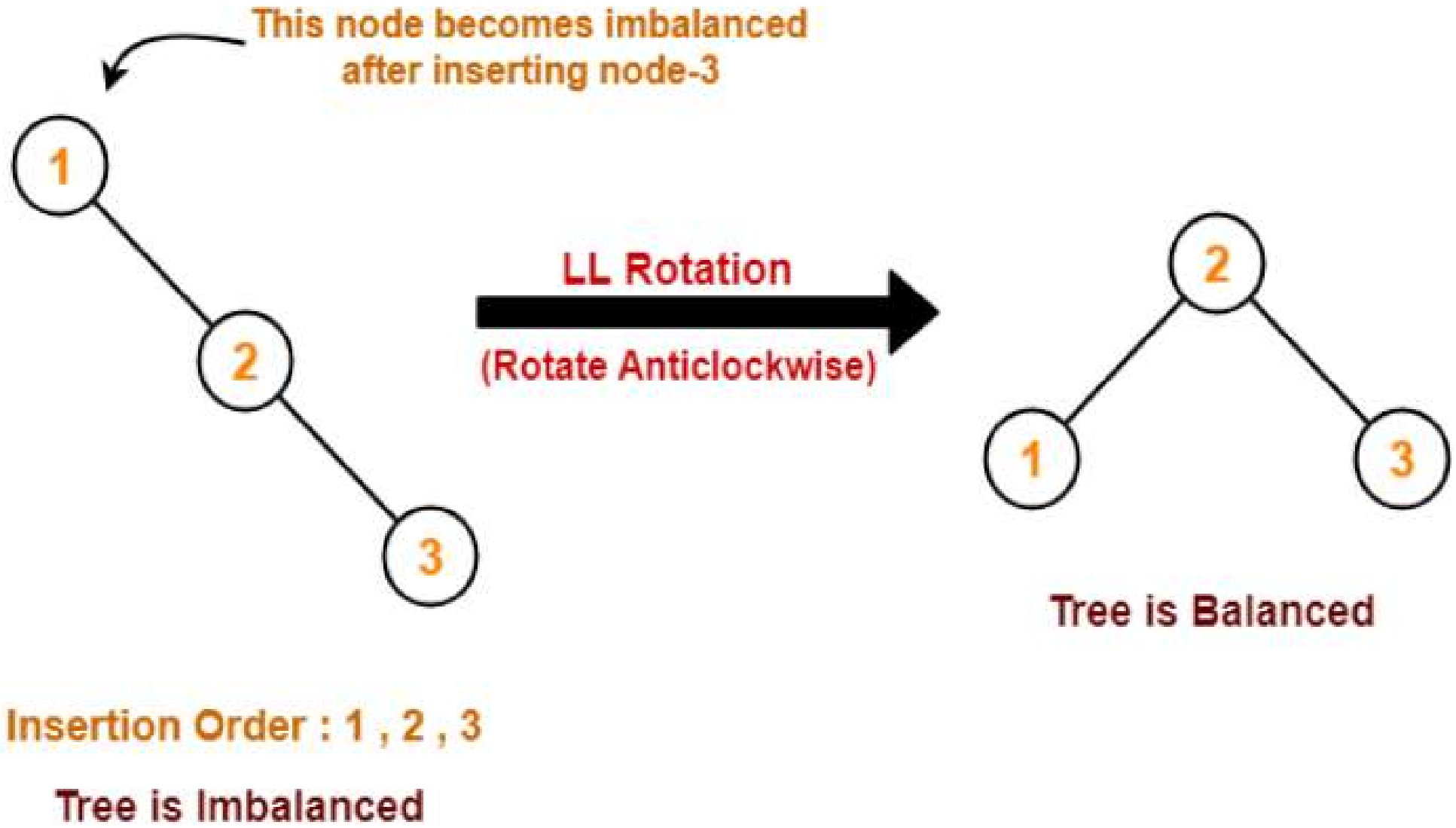


Height = 3

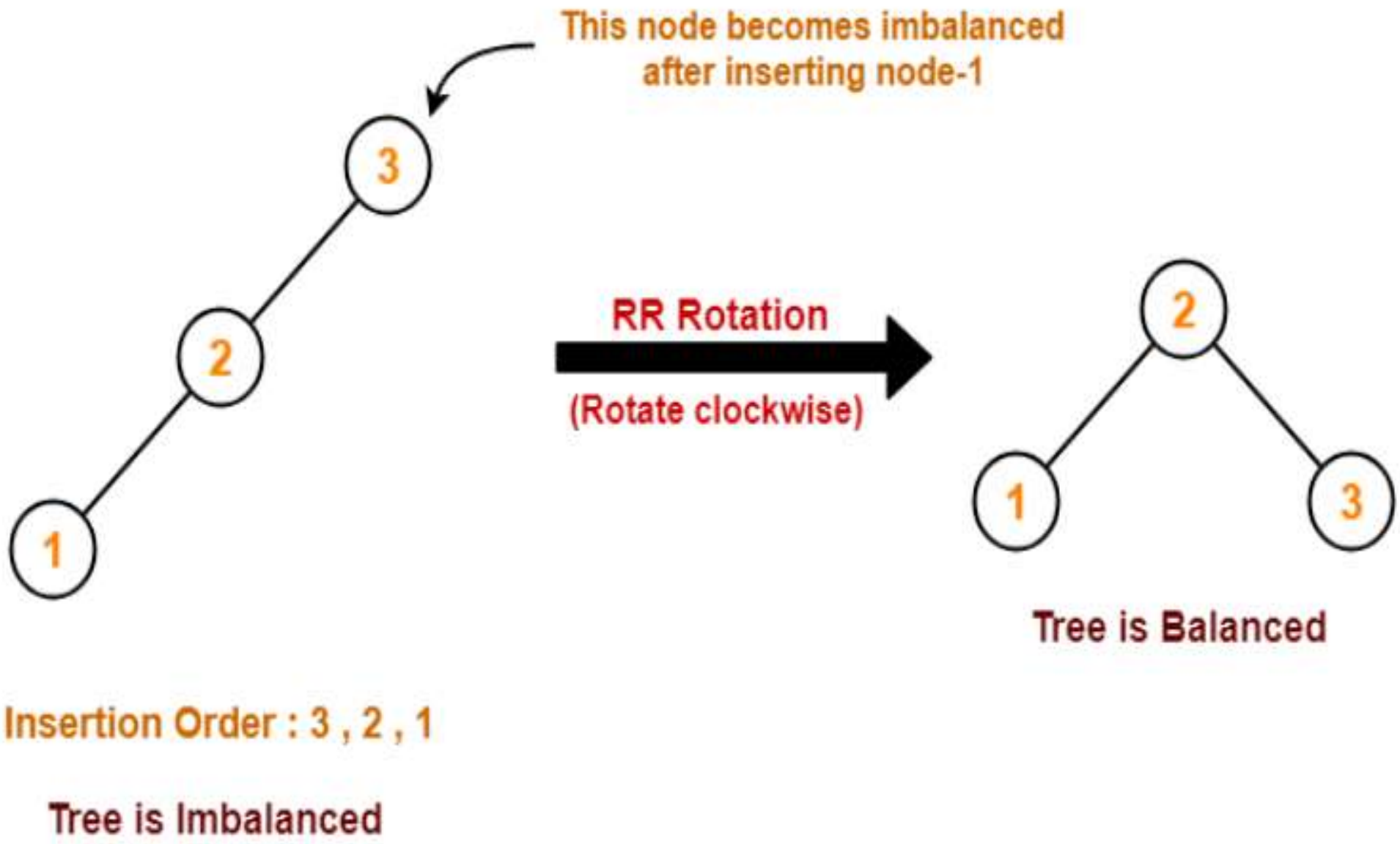

AVL Tree Rotations
Single Rotations
Double Rotations
Left Rotation
(LL Rotation)
Right Rotation
(RR Rotation)
Left-Right Rotation
(LR Rotation)
Right-Left Rotation
(RL Rotation)


x
y
α
β
γ
Right Rotate(x)
y
α
x
β
γ
y
α
x
β
γ
Left Rotate(y)
x
y
γ
α
β

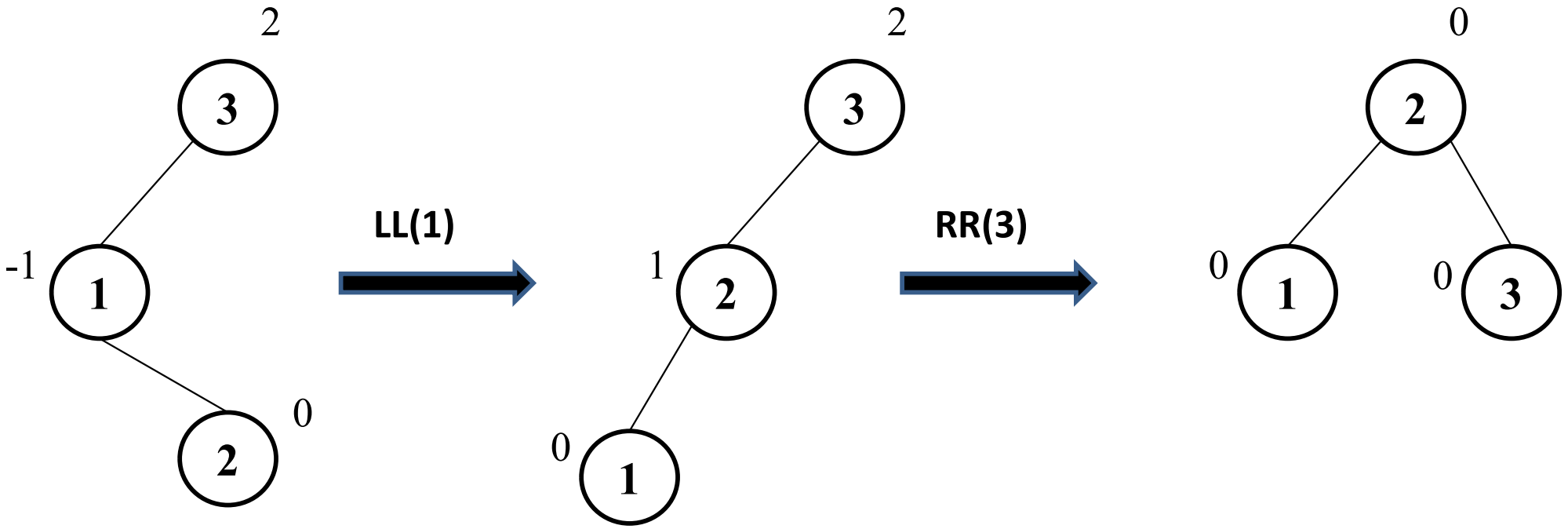
1)Single Left Rotation(LL Rotation)



2)Single Right Rotation(RR Rotation)

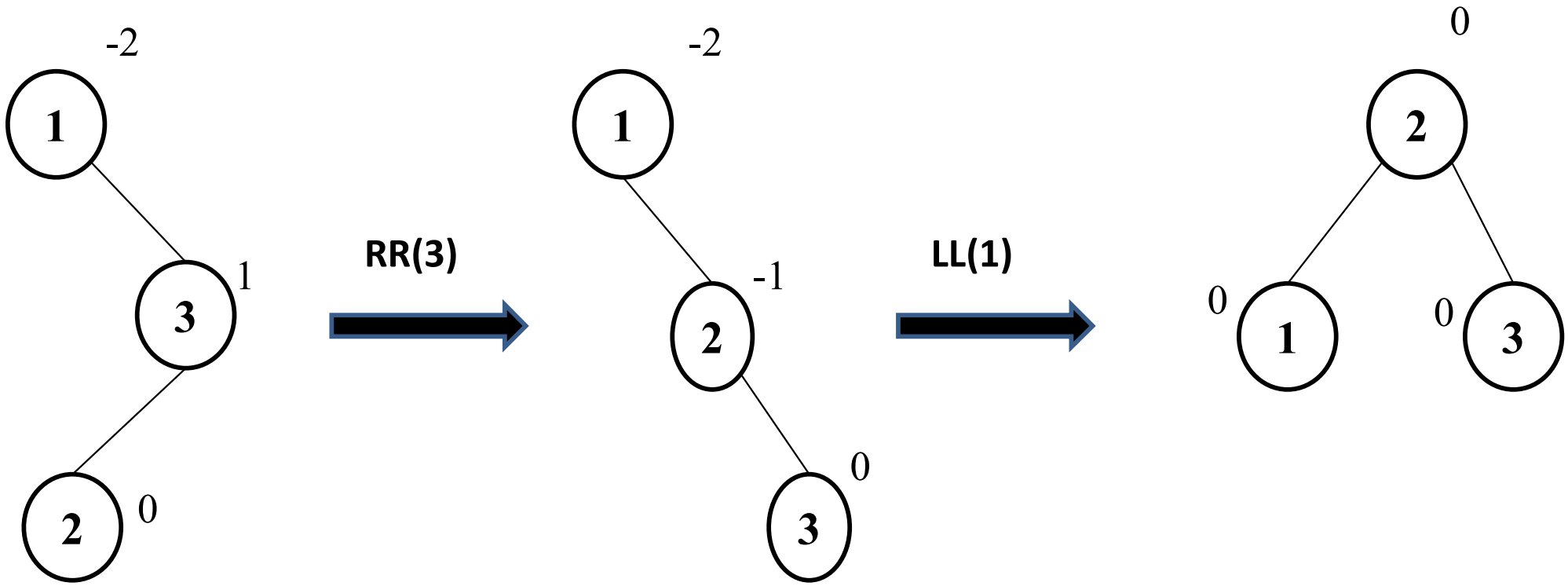


3)Left-Right Rotation(LR Rotation)



Insertion Order: 3,1,2

4) Right-Left Rotation(RL Rotation)



Insertion Order: 1,3,2

AVL Tree Insertion

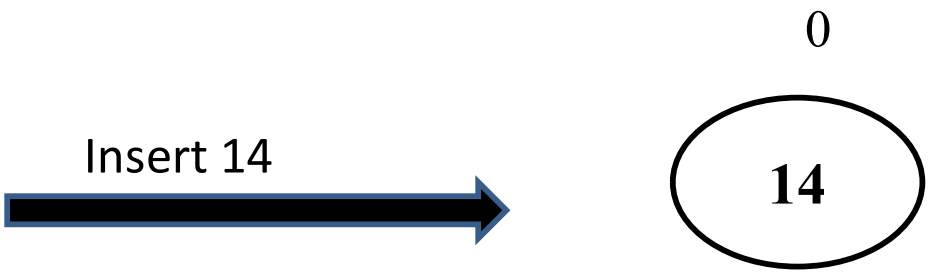
1. Insert the node as the leaf node. Use BST insertion procedure
2. After insertion check the balance factor of every node
3. If the tree is imbalanced, perform the suitable rotation. Let z be the newly inserted node. x be the first unbalanced node on the path from z to root. y be the child of x on the path from z to root
   a) If y is the left child of x and z is in the left subtree of y, then perform RR Rotation with respect to x.
   b) If y is the right child of x and z is in the right subtree of y, then perform LL Rotation with respect to x.
   c) If y is the left child of x and z is in the right subtree of y, then perform LR Rotation
   d) If y is the right child of x and z is in the left subtree of y, then perform RL Rotation

AVL Tree Insertion

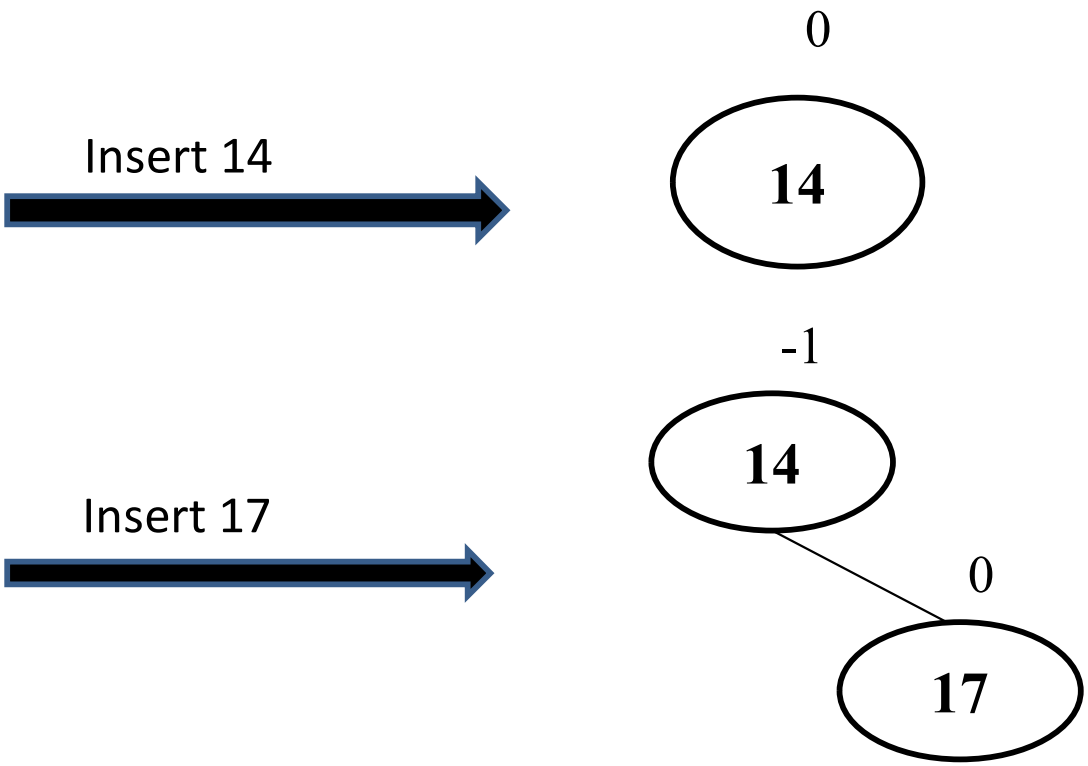
- Time Complexity of AVL tree insertion = O(log n)

Insert 14,17,11,7,53,4 and 13 in to an empty AVL tree

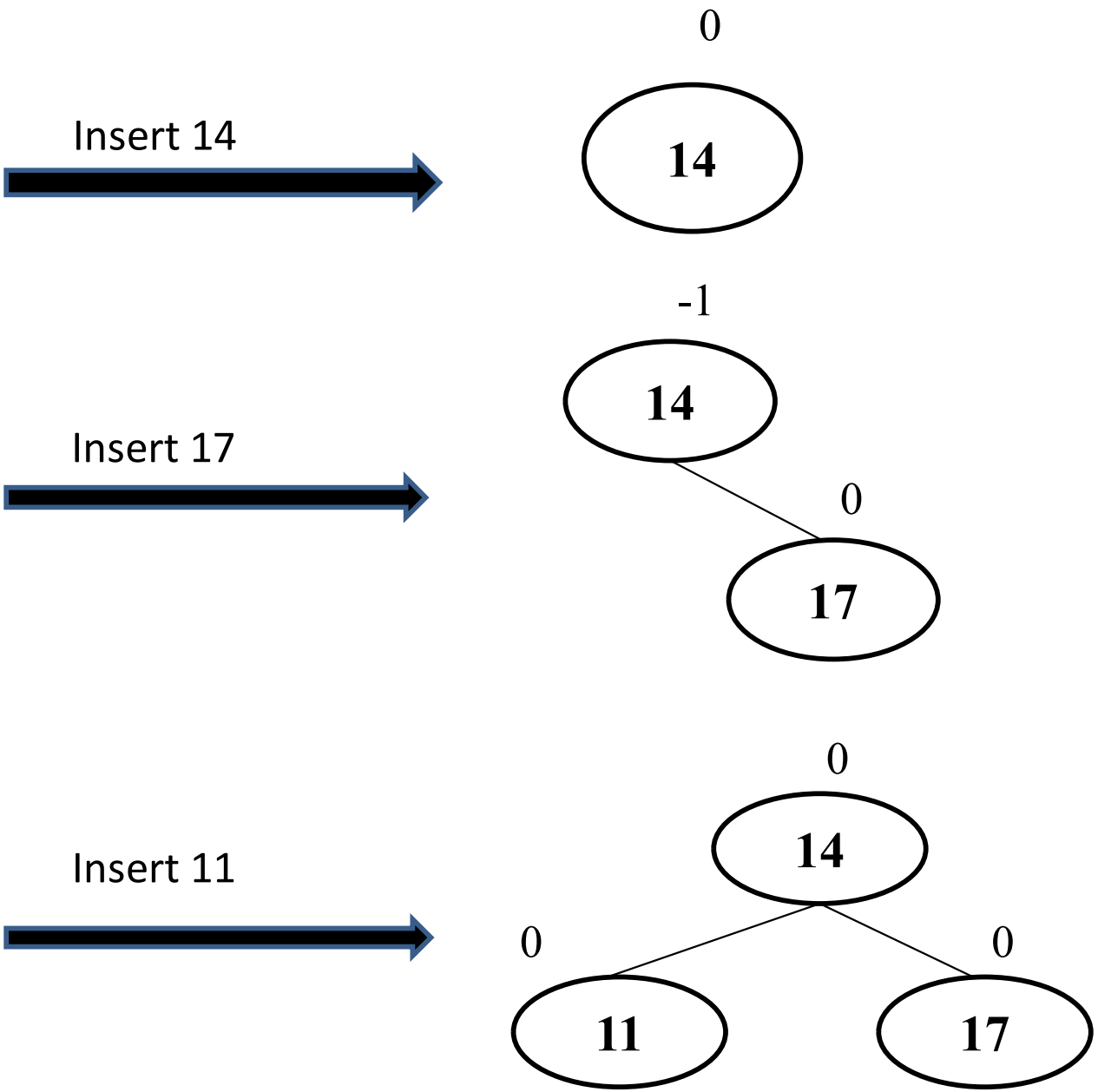
Insert **14**,17,11,7,53,4 and 13 in to an empty AVL tree



Insert 14,17,11,7,53,4 and 13 in to an empty AVL tree

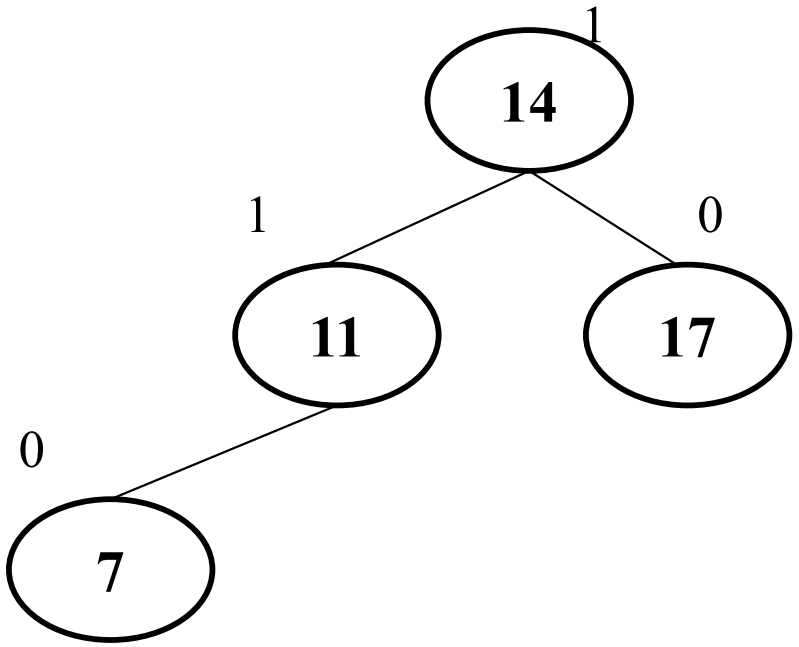


Insert 14,17,11,7,53,4 and 13 in to an empty AVL tree



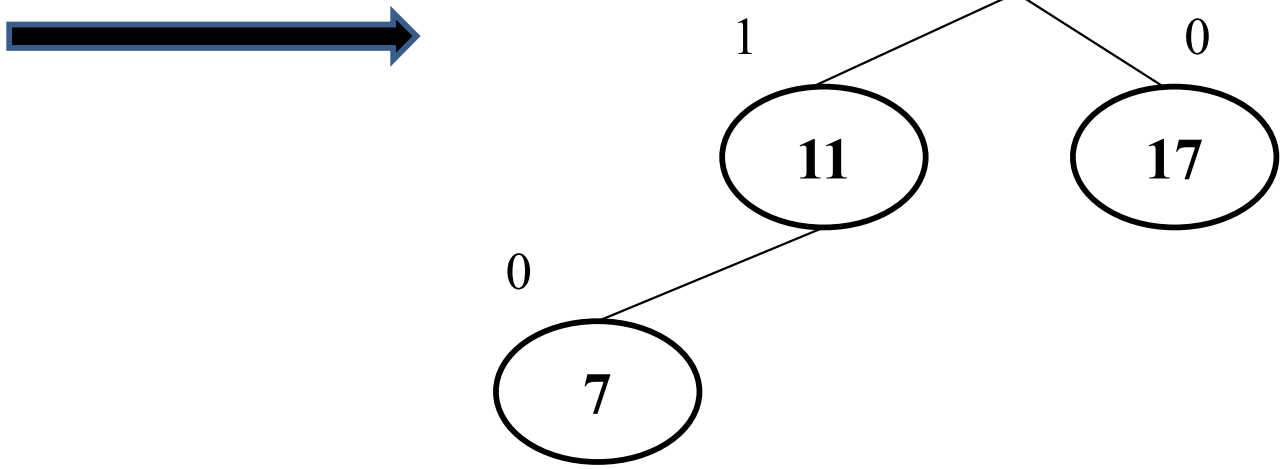
Insert 14,17,11,7,53,4 and 13 in to an empty AVL tree

Insert 7

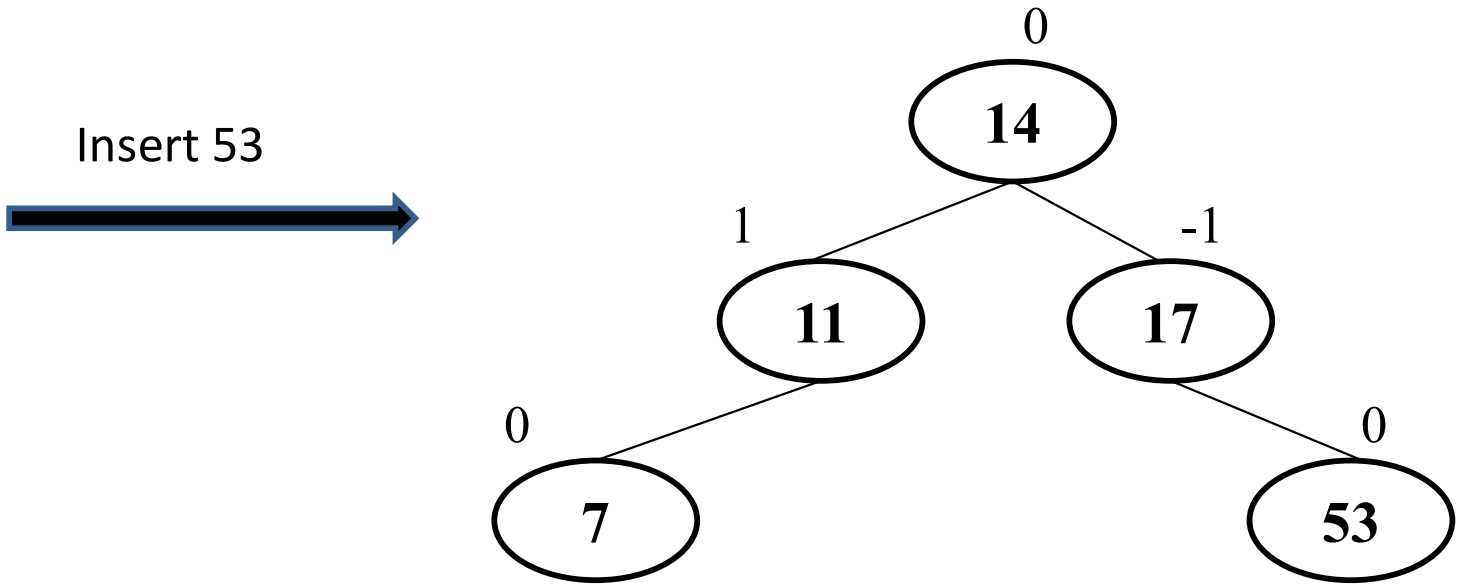


Insert 14,17,11,7,53,4 and 13 in to an empty AVL tree

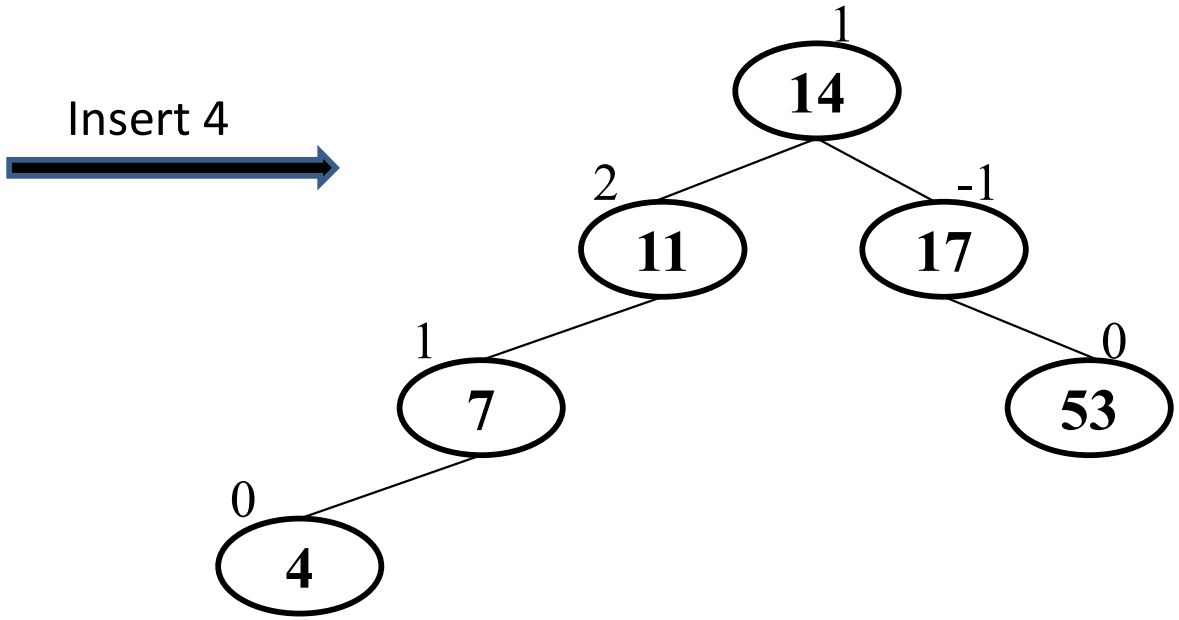
Insert 7



Insert 53

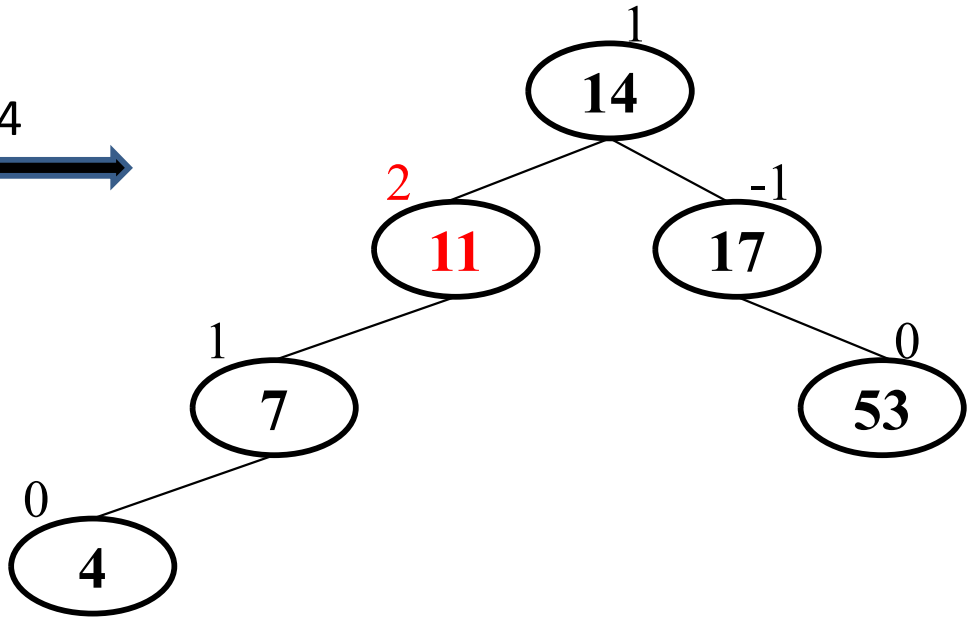


Insert 14,17,11,7,53,**4** and 13 in to an empty AVL tree



Insert 14,17,11,7,53,4 and 13 in to an empty AVL tree

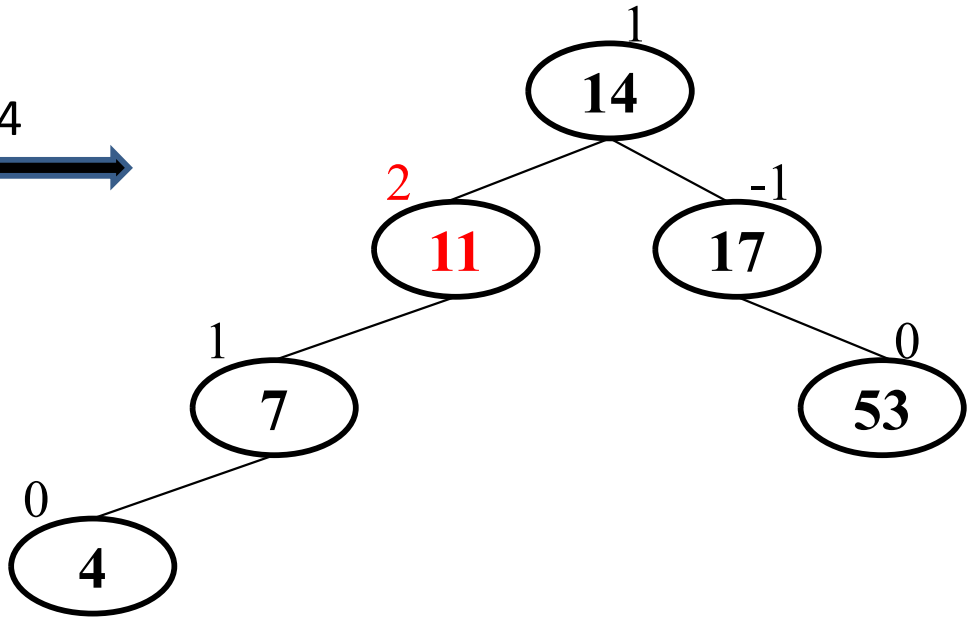
Insert 4



The tree is now imbalanced because the balance factor of node 11 is 2.
Perform RR Rotation with respect to 11

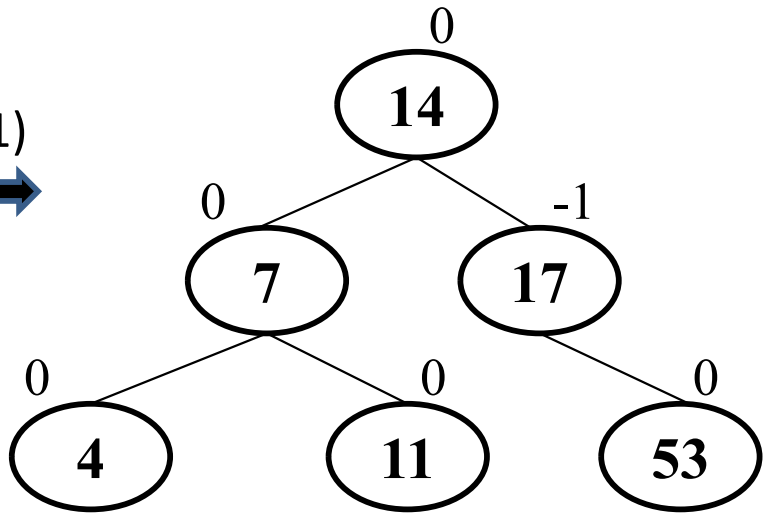
Insert 14,17,11,7,53,4 and 13 in to an empty AVL tree

Insert 4



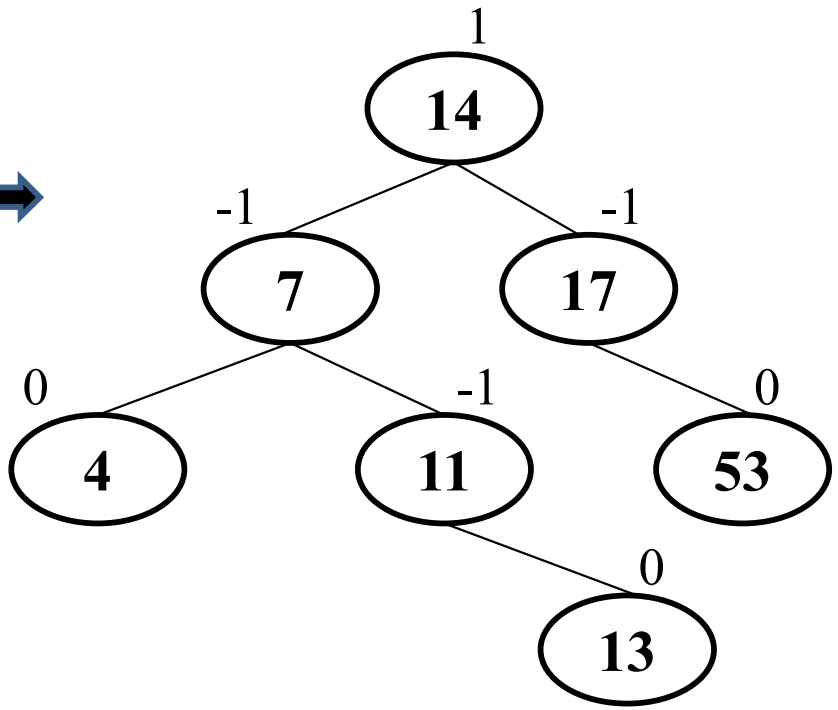
The tree is now imbalanced because the balance factor of node 11 is 2.
Perform RR Rotation with respect to 11

RR Rotation(11)

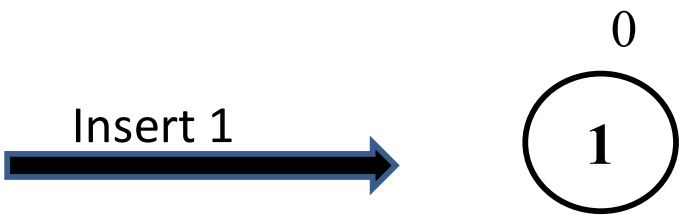


Insert 14,17,11,7,53,4 and **13** in to an empty AVL tree

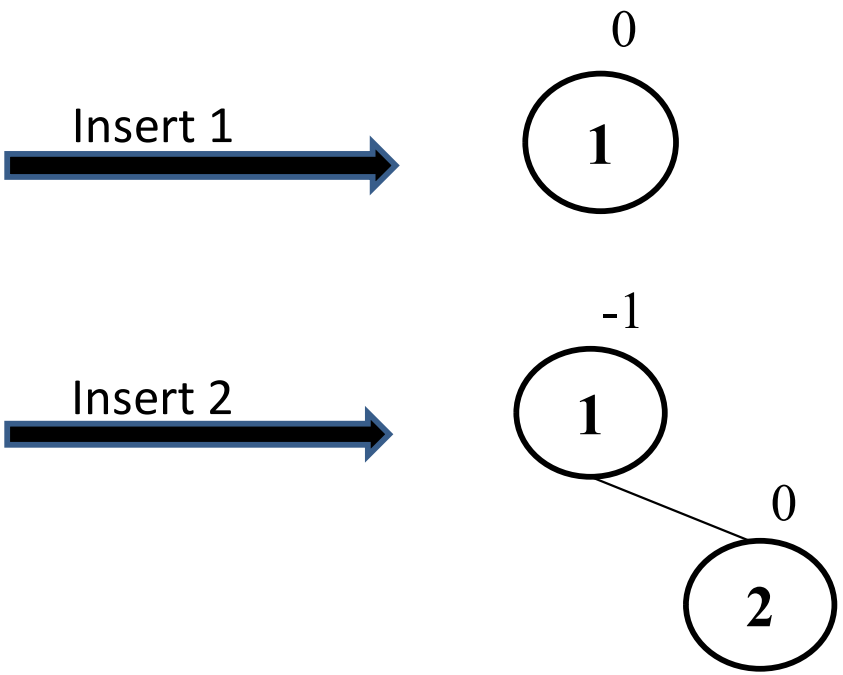
Insert 13



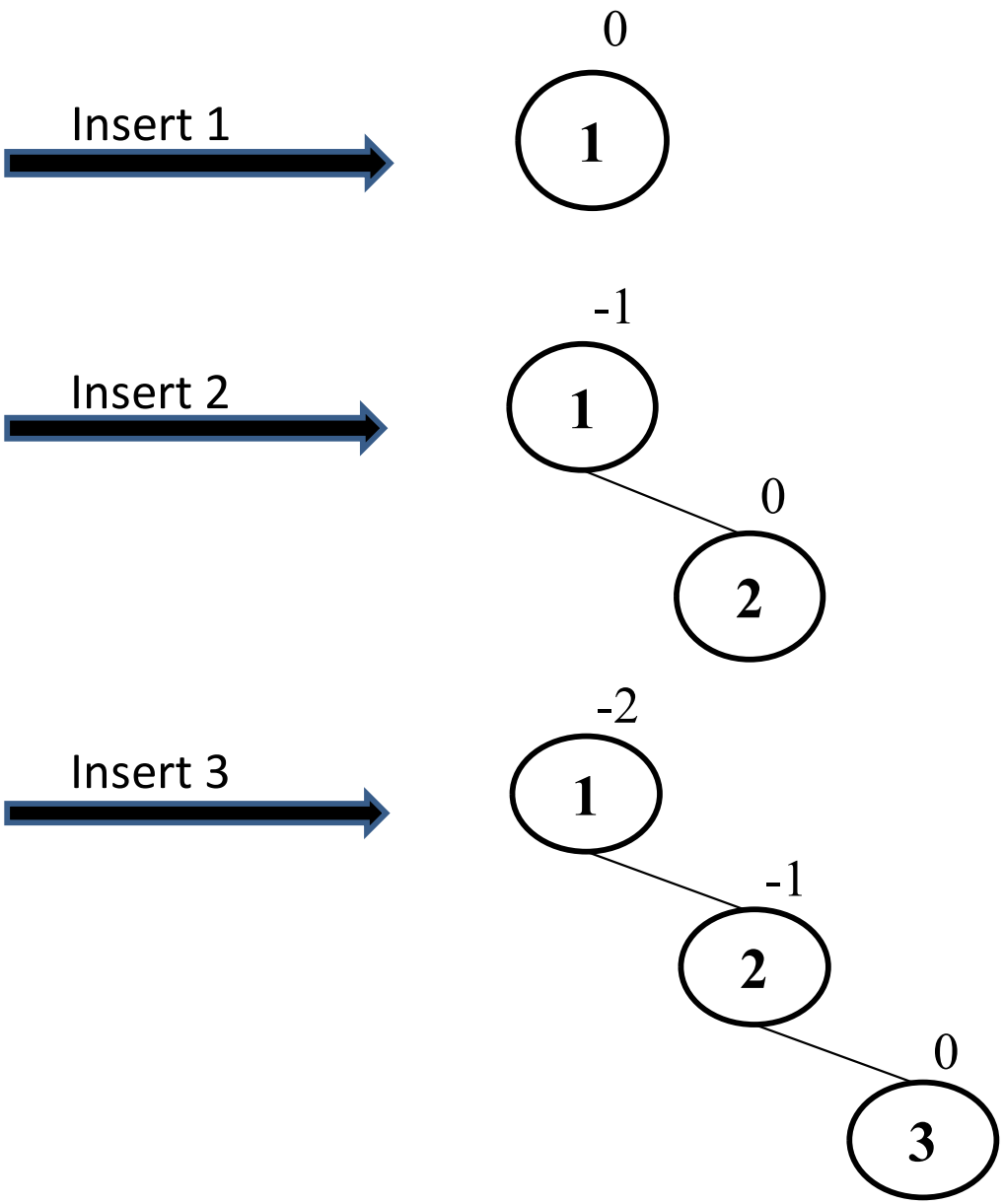
Insert numbers from 1 to 8 into an empty AVL tree



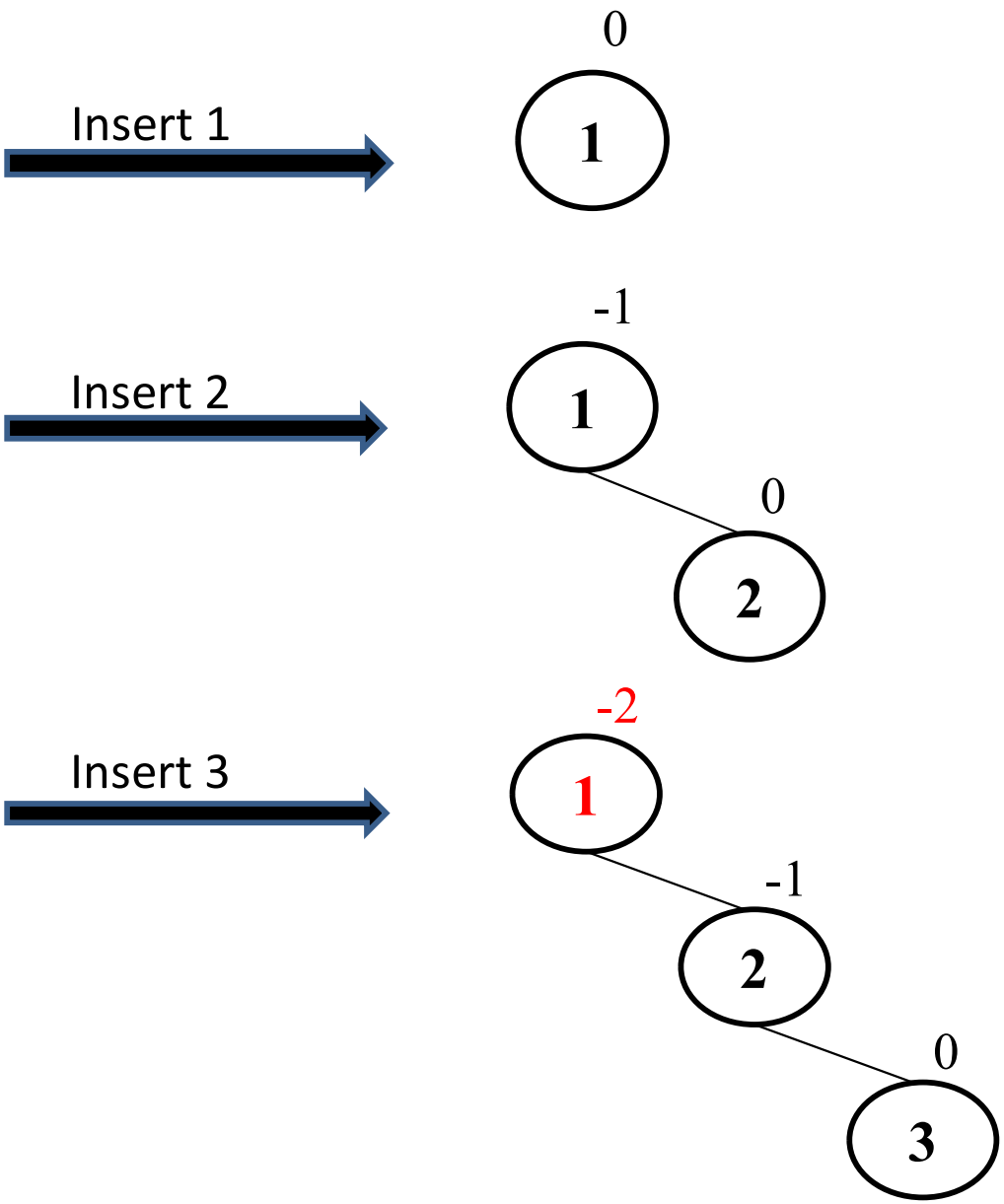
Insert numbers from 1 to 8 into an empty AVL tree



Insert numbers from 1 to 8 into an empty AVL tree

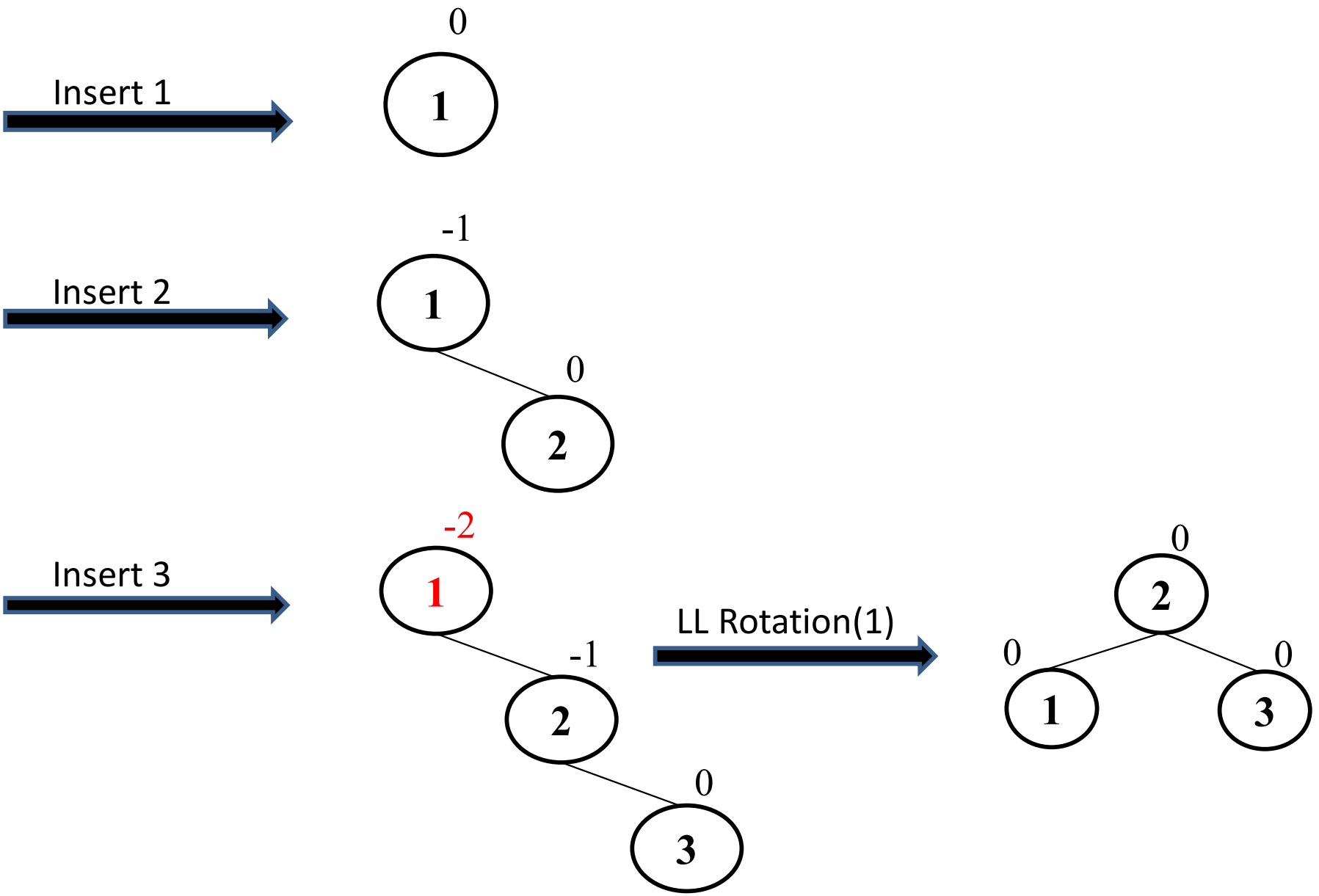


Insert numbers from 1 to 8 into an empty AVL tree



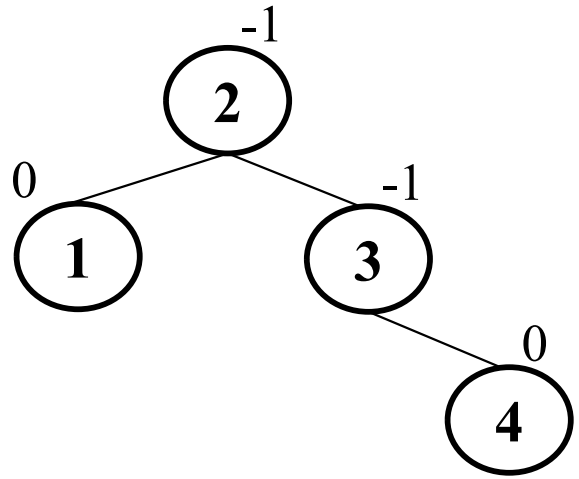
The tree is now imbalanced because the balance factor of node 1 is -2.
Perform LL Rotation with respect to 1

Insert numbers from 1 to 8 into an empty AVL tree



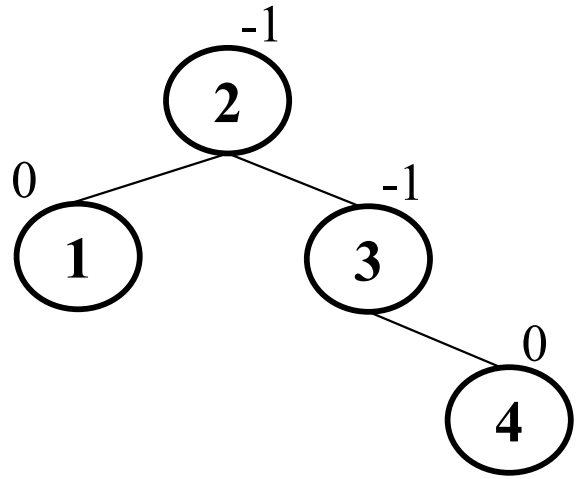
Insert numbers from 1 to 8 into an empty AVL tree

Insert 4

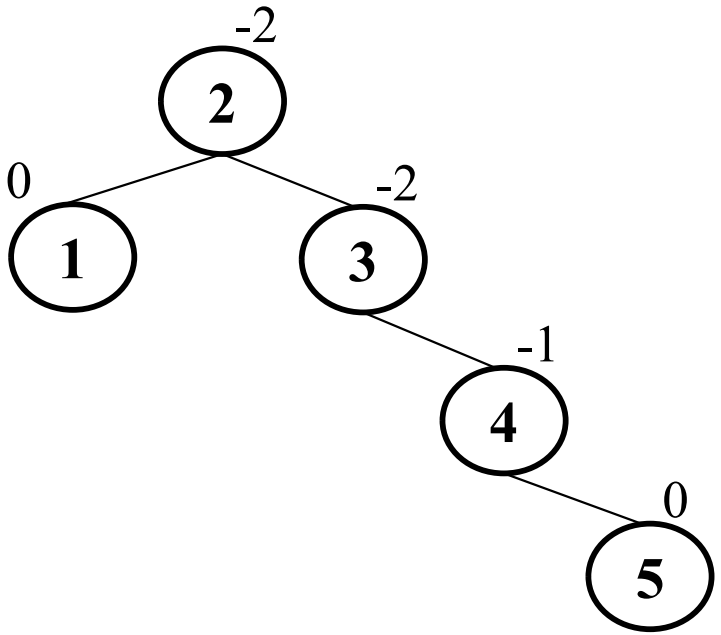


Insert numbers from 1 to 8 into an empty AVL tree

Insert 4

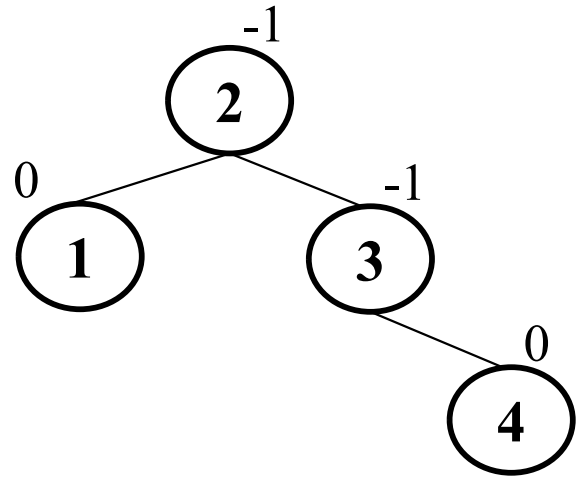


Insert 5

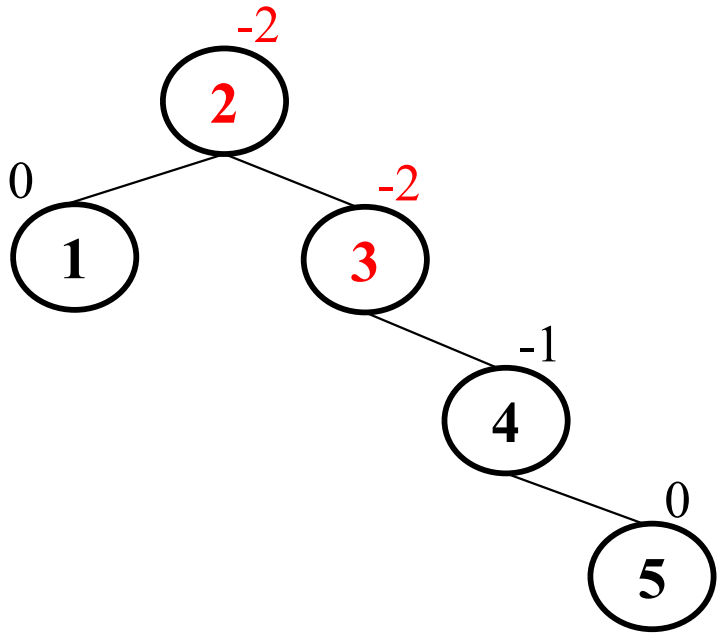


Insert numbers from 1 to 8 into an empty AVL tree

Insert 4

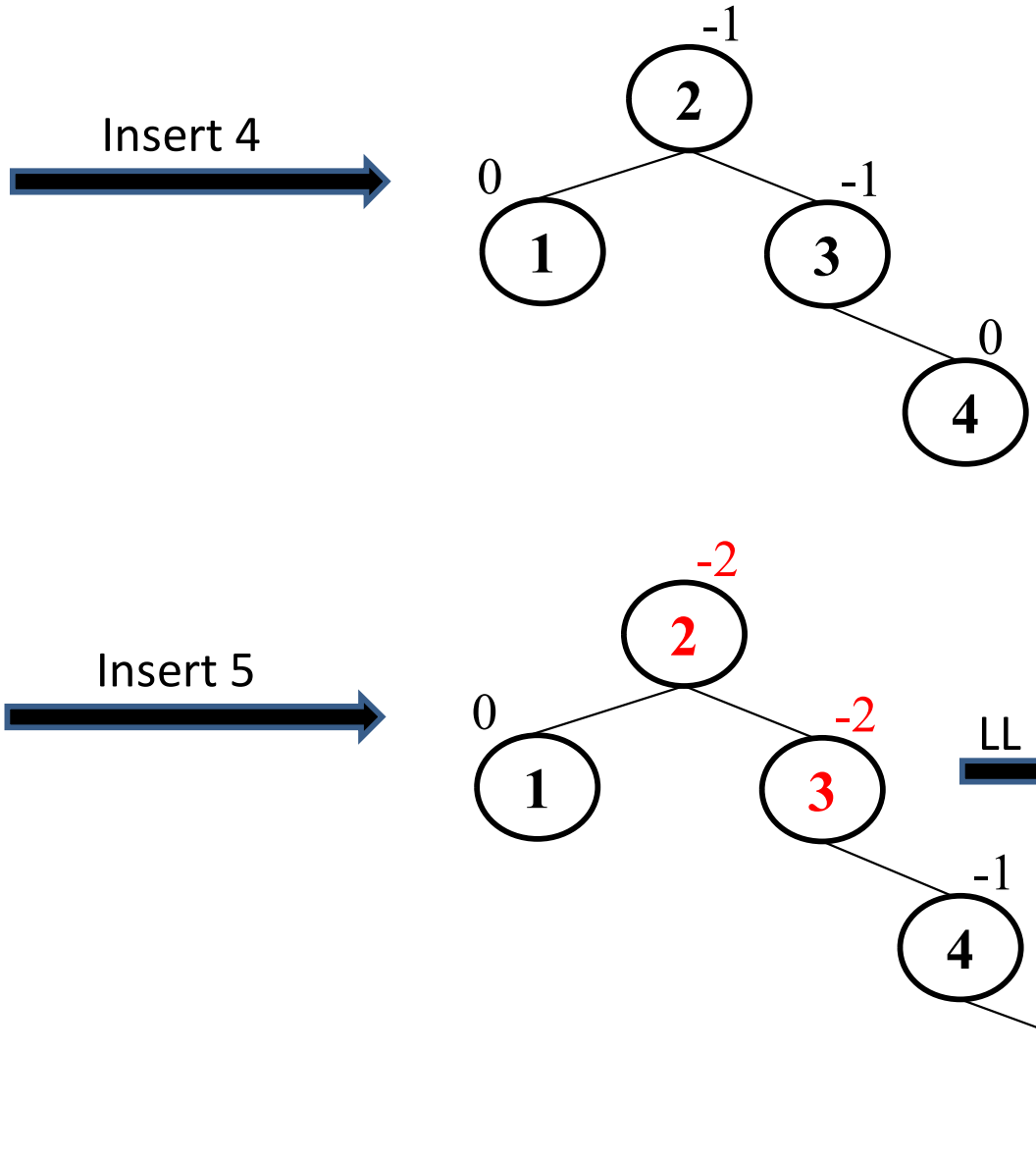


Insert 5

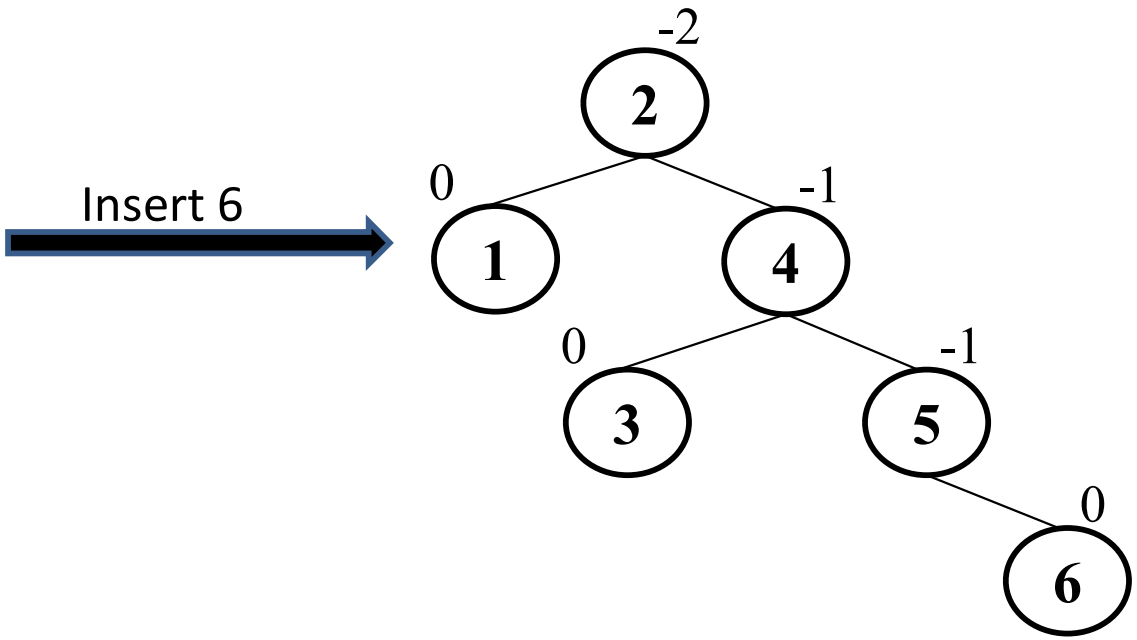


The tree is now imbalanced because the balance factor of node 2 and 3 are -2.
Perform LL Rotation with respect to 3

Insert numbers from 1 to 8 into an empty AVL tree

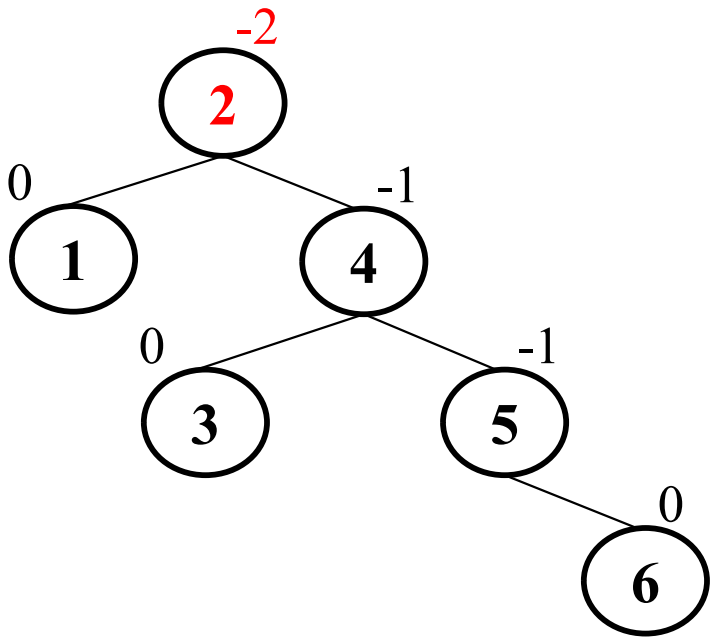


Insert numbers from 1 to 8 into an empty AVL tree



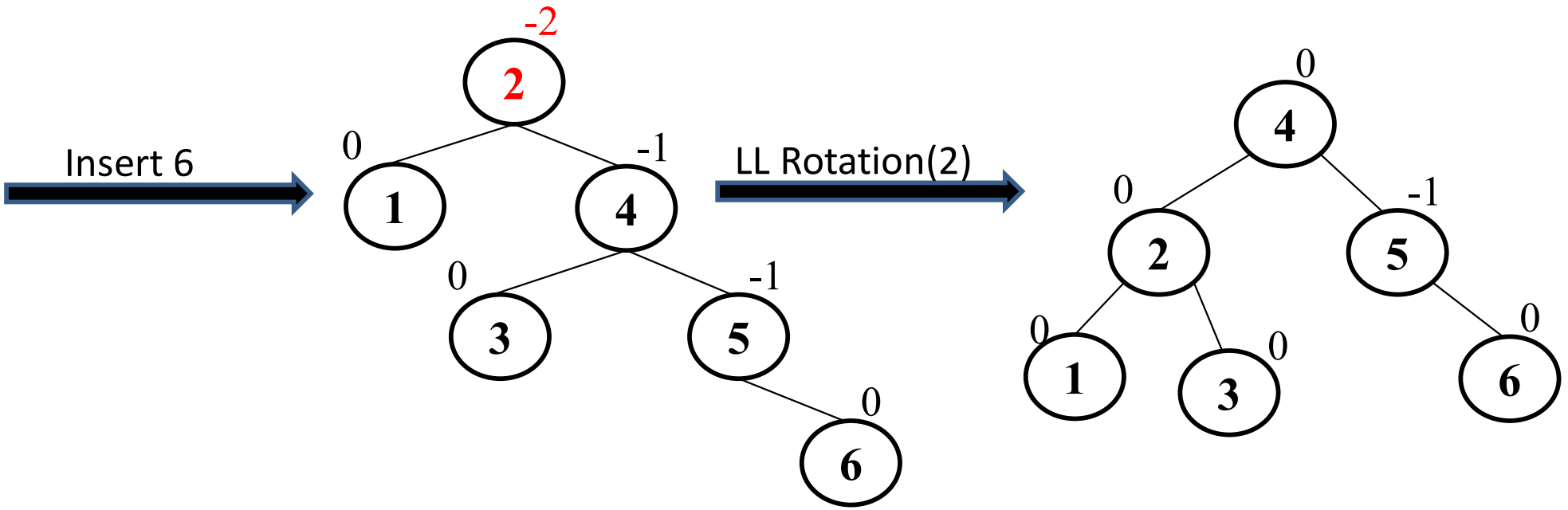
Insert numbers from 1 to 8 into an empty AVL tree

Insert 6

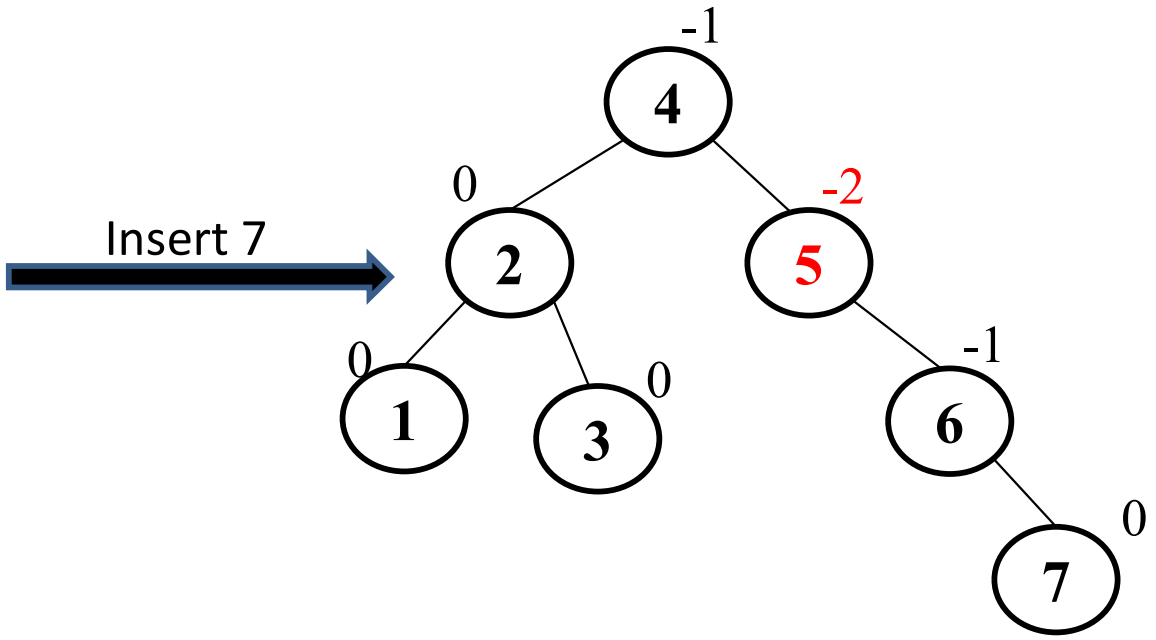


The tree is now imbalanced because the balance factor of node 2 is -2.
Perform LL Rotation with respect to 2

Insert numbers from 1 to 8 into an empty AVL tree

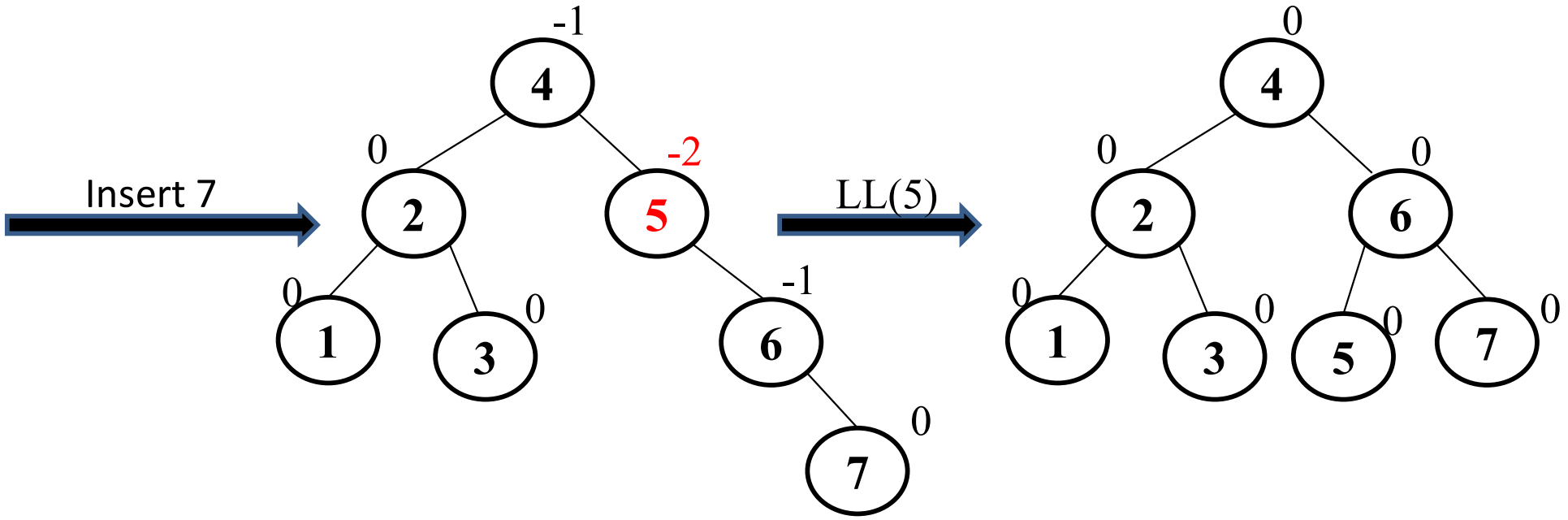


Insert numbers from 1 to 8 into an empty AVL tree

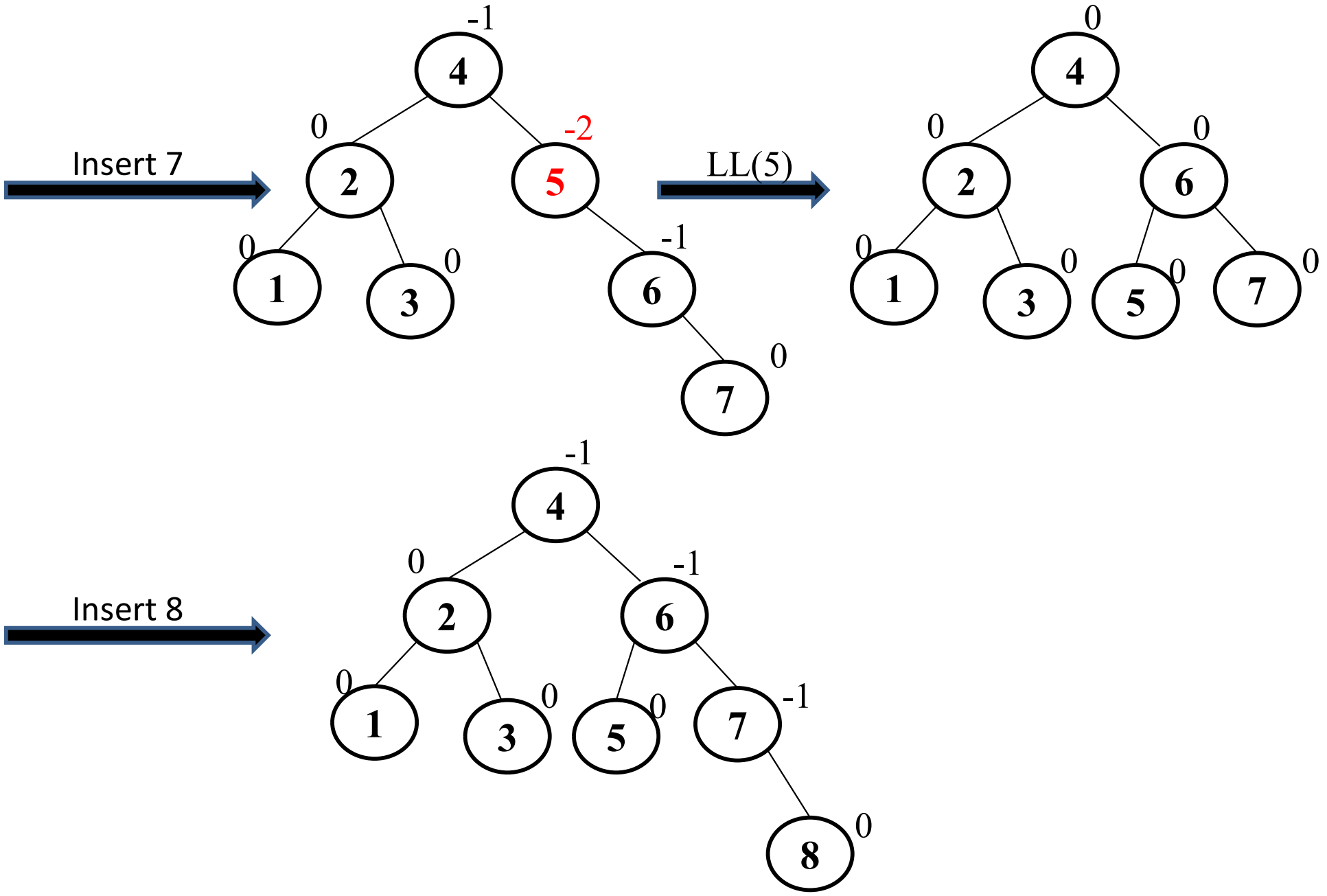


The tree is now imbalanced because the balance factor of node 5 is -2.
Perform LL Rotation with respect to 5

Insert numbers from 1 to 8 into an empty AVL tree

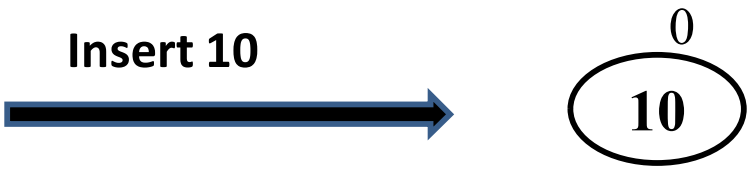


Insert numbers from 1 to 8 into an empty AVL tree

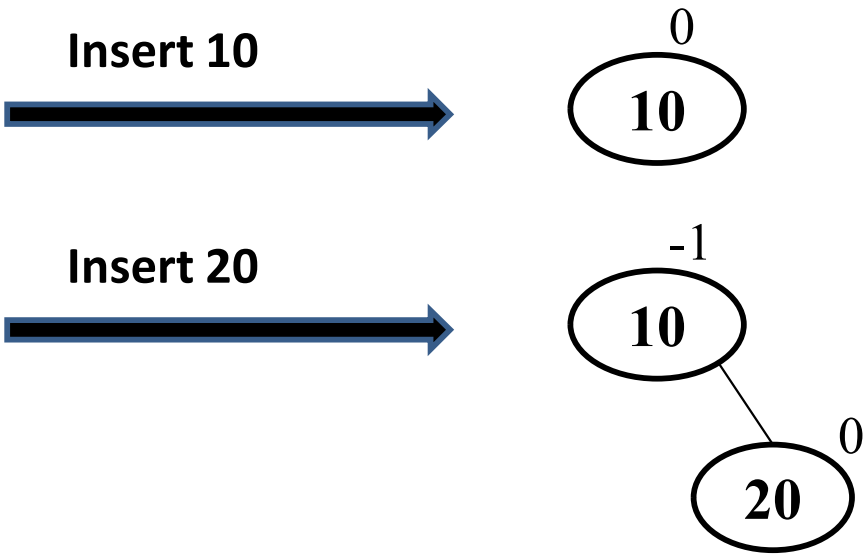


Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL tree

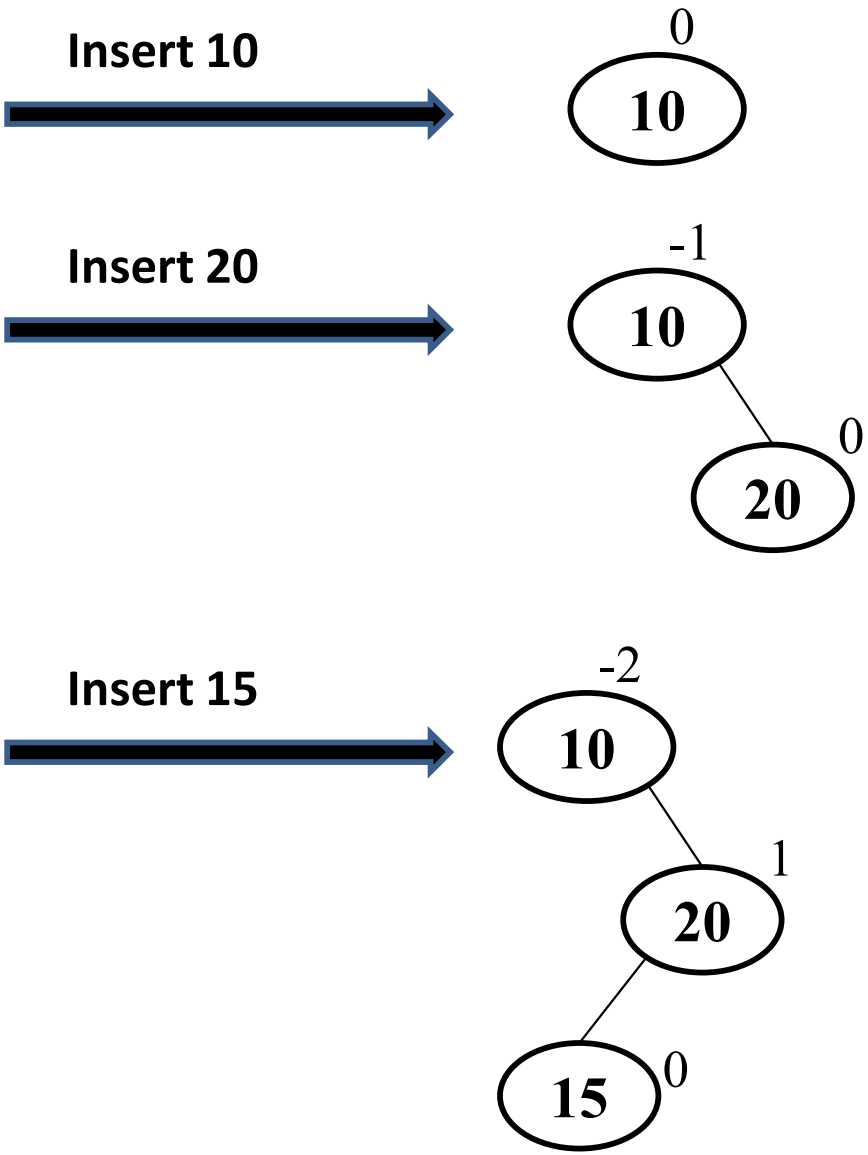
Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL tree



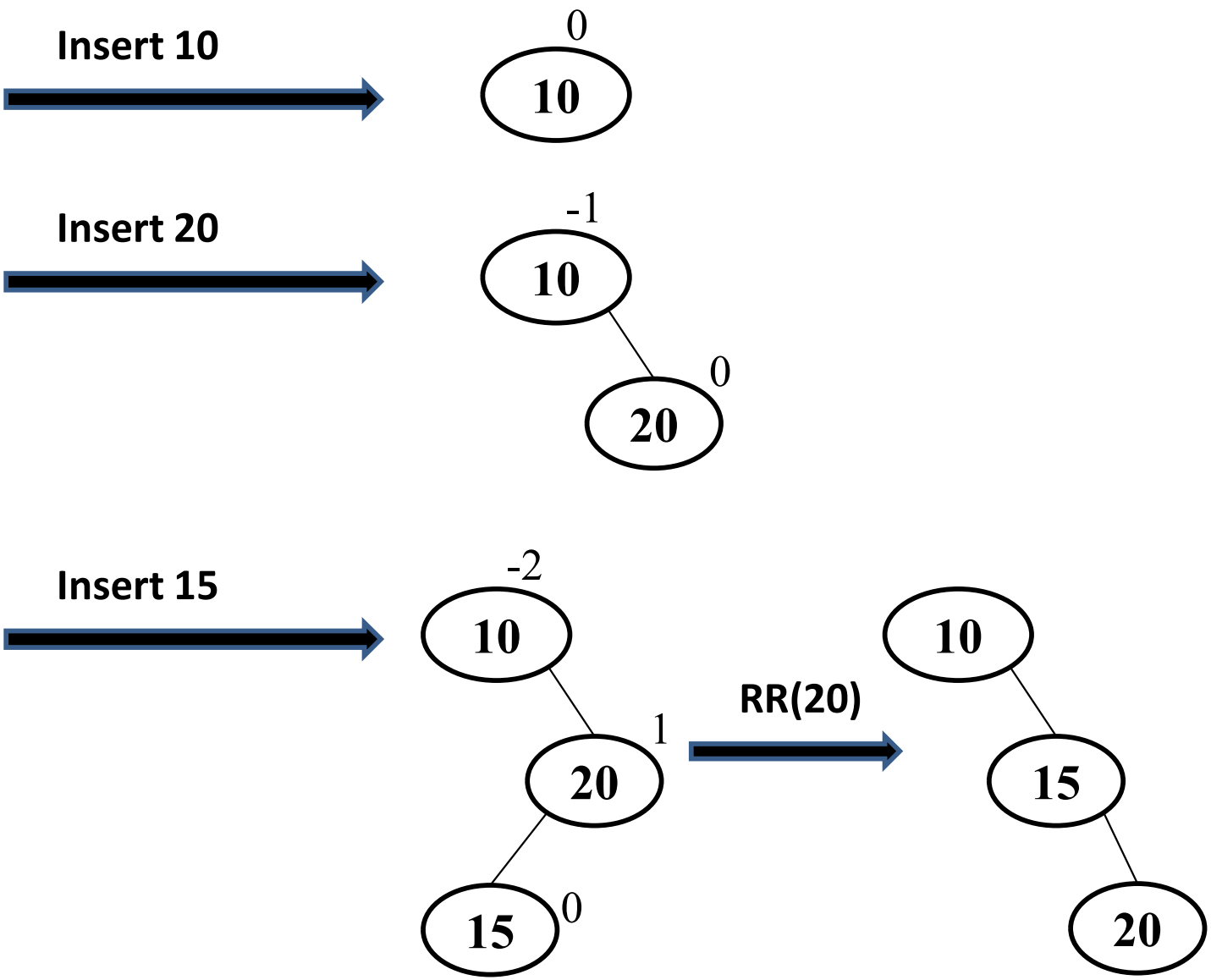
Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL tree



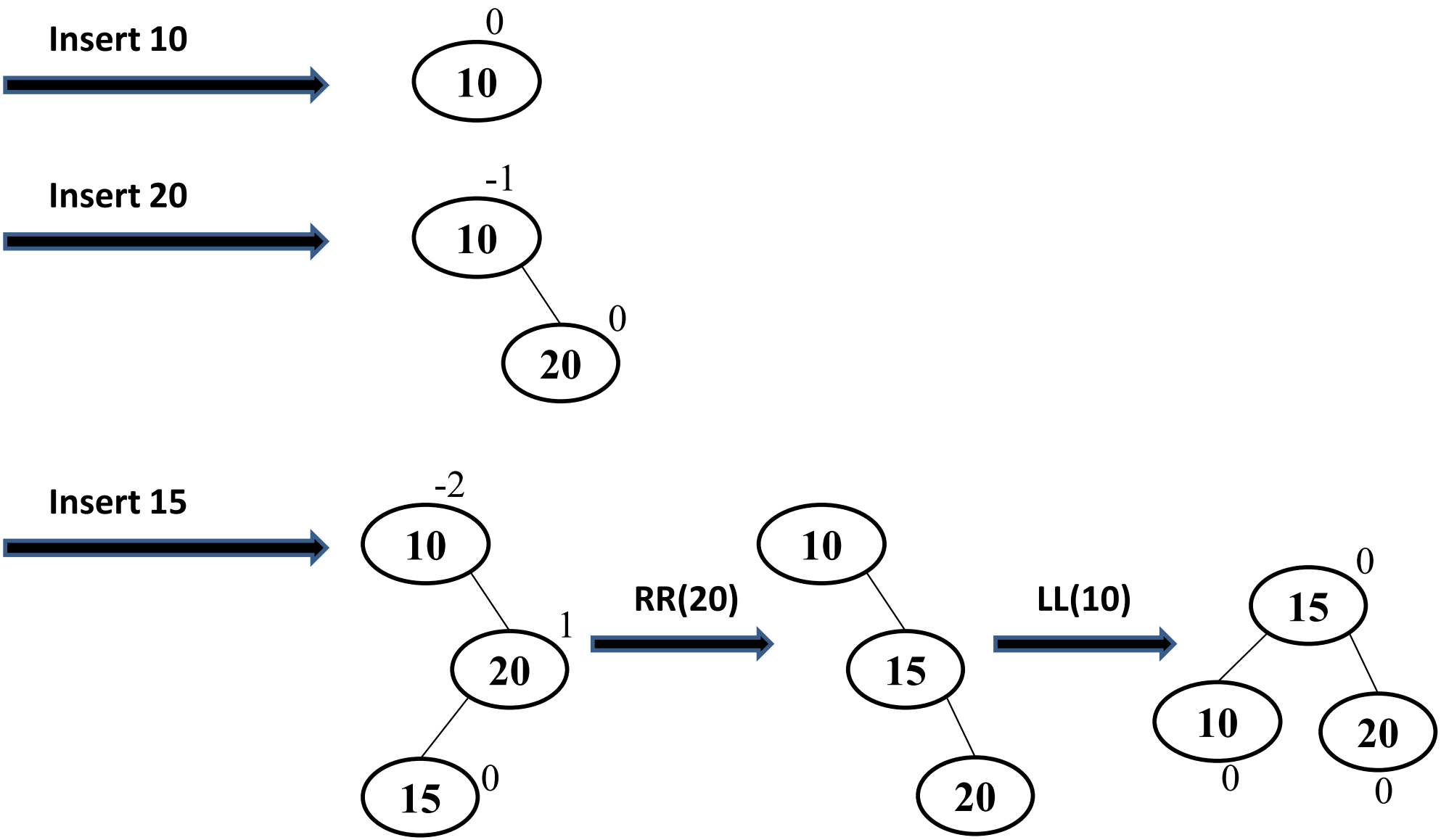
Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL tree



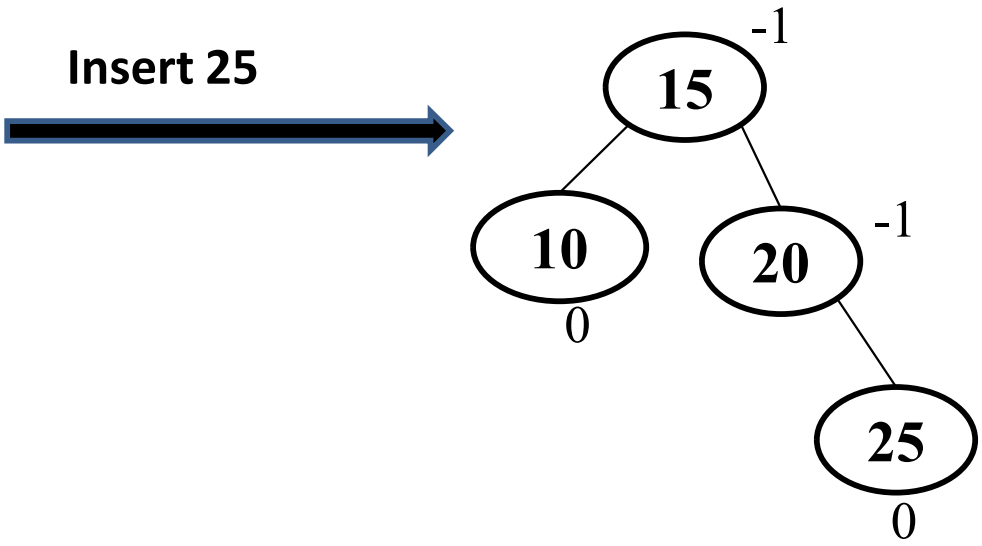
Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL tree



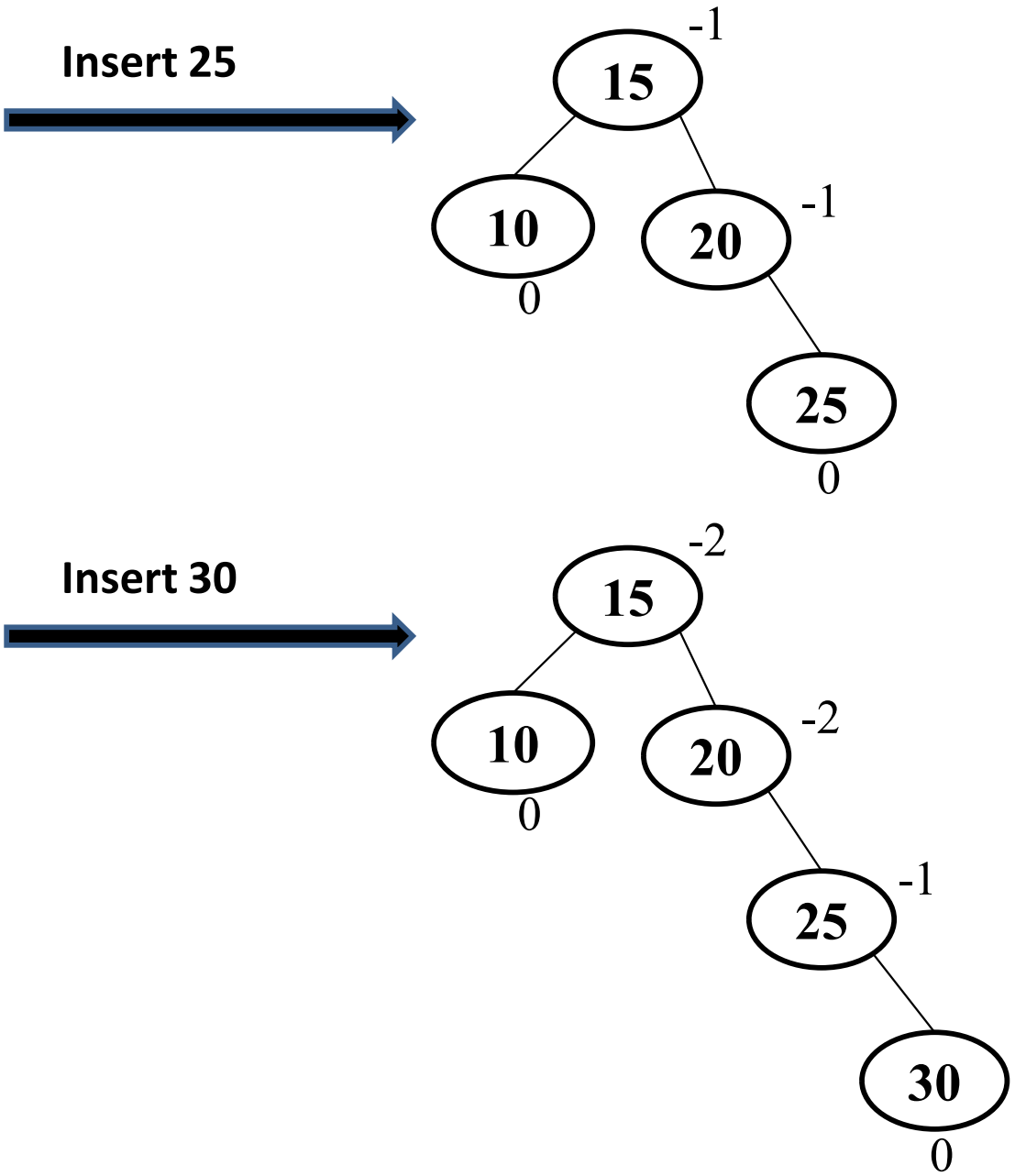
Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL tree



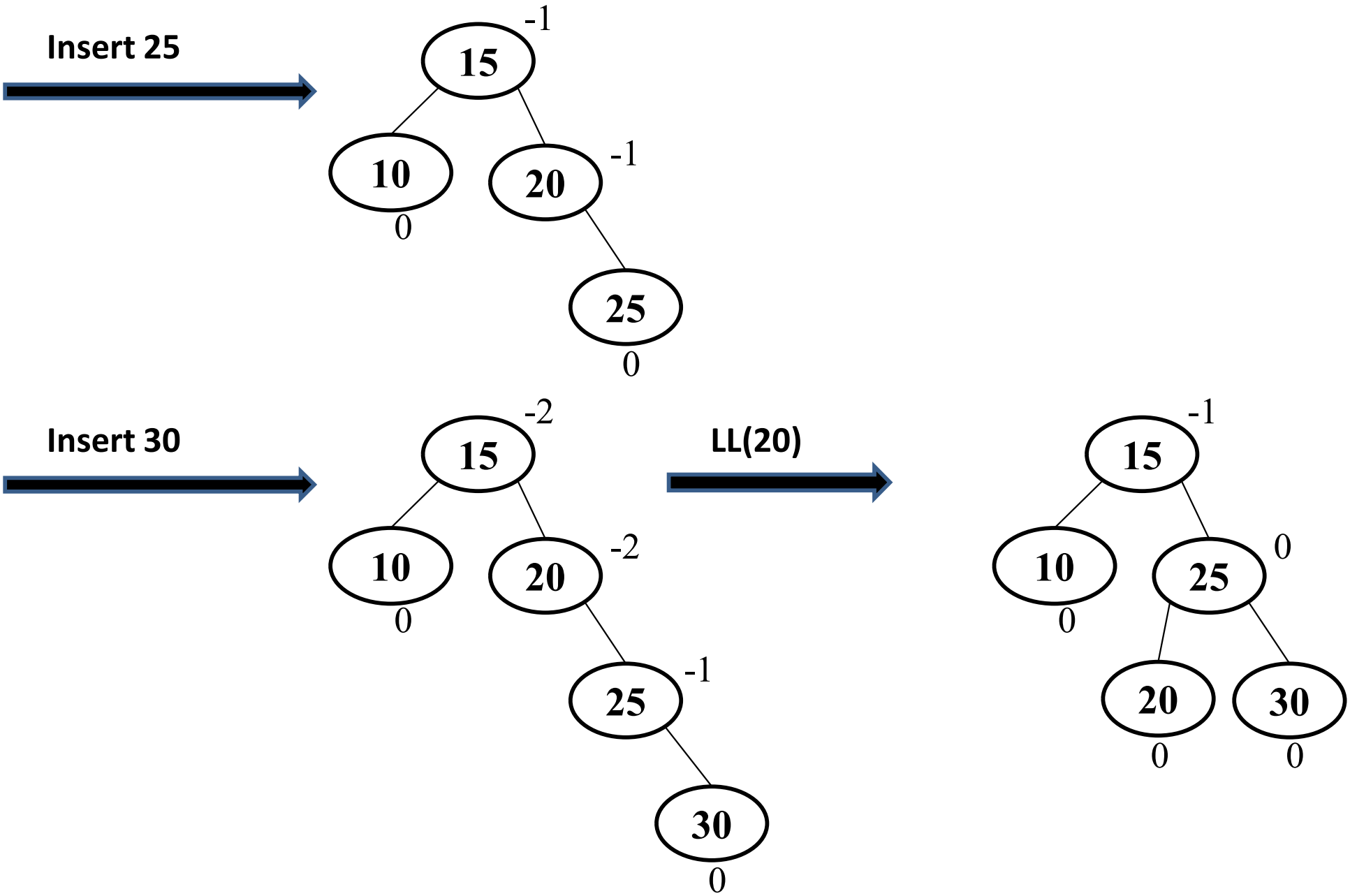
Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL tree



Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL tree

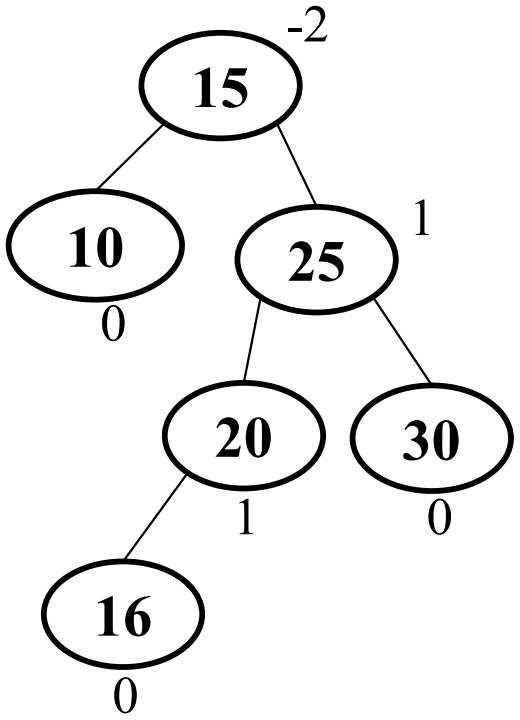


Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL tree

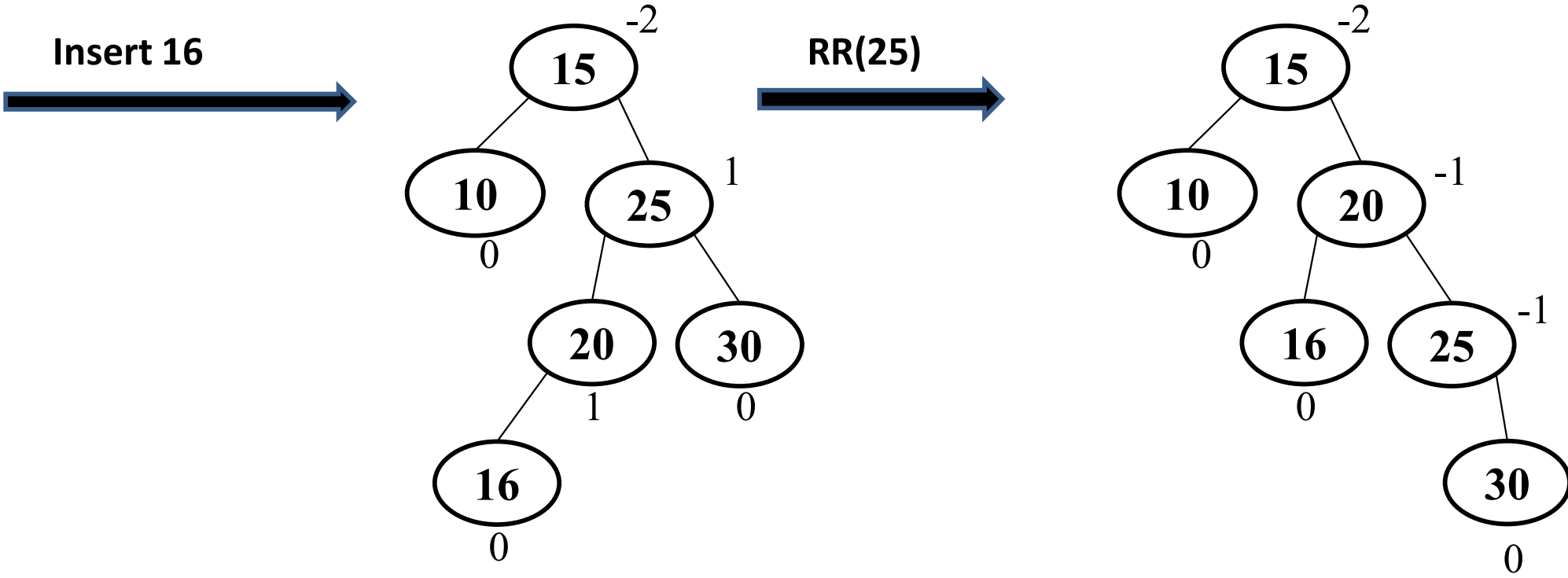


Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL tree

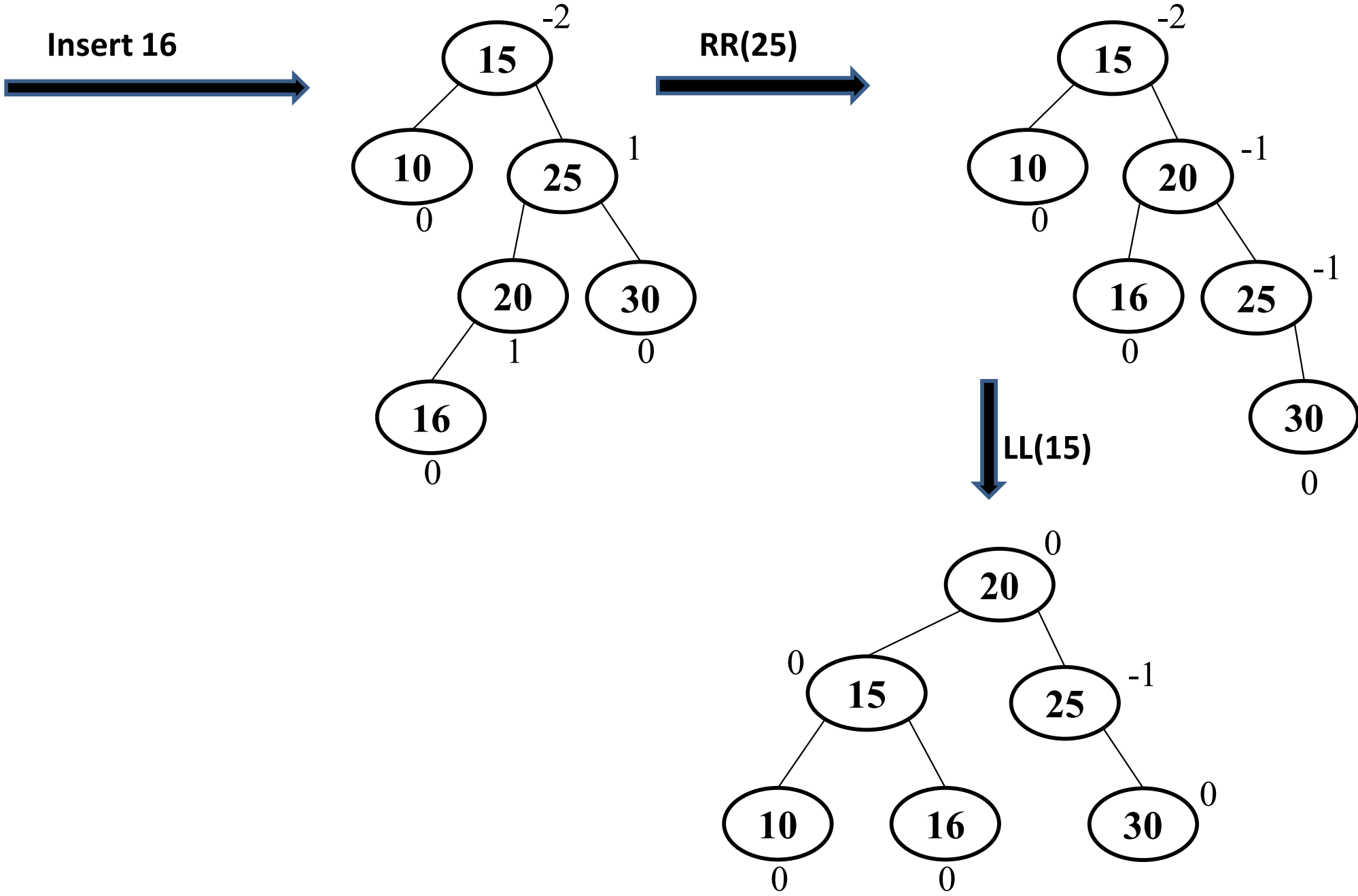
Insert 16



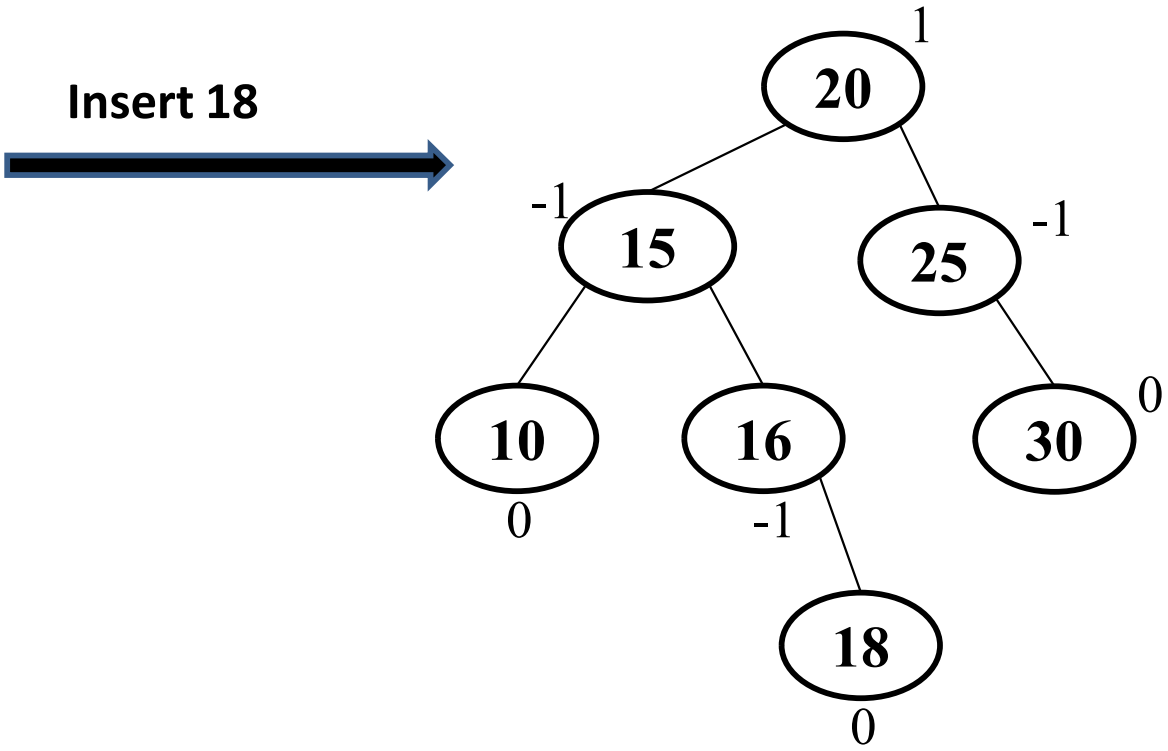
Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL tree



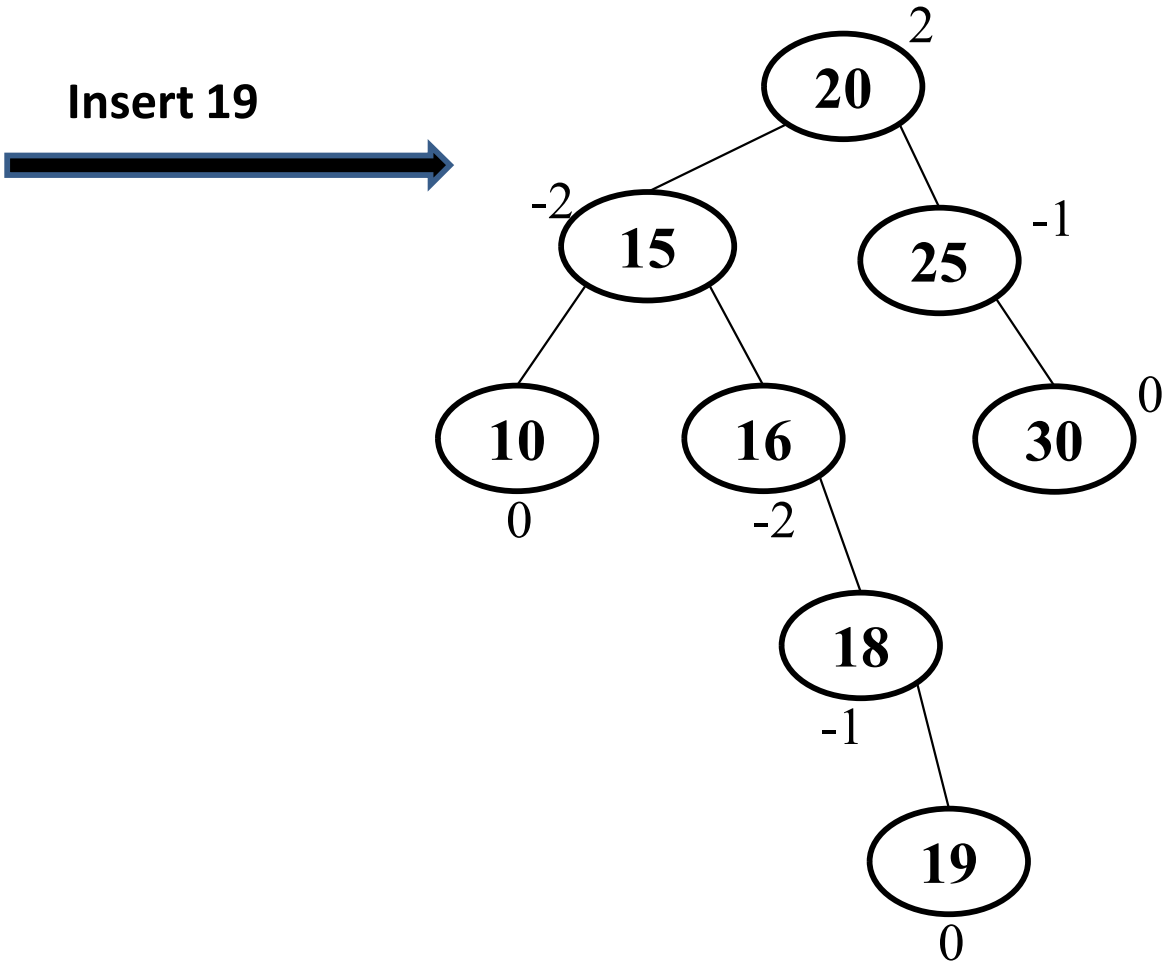
Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL tree



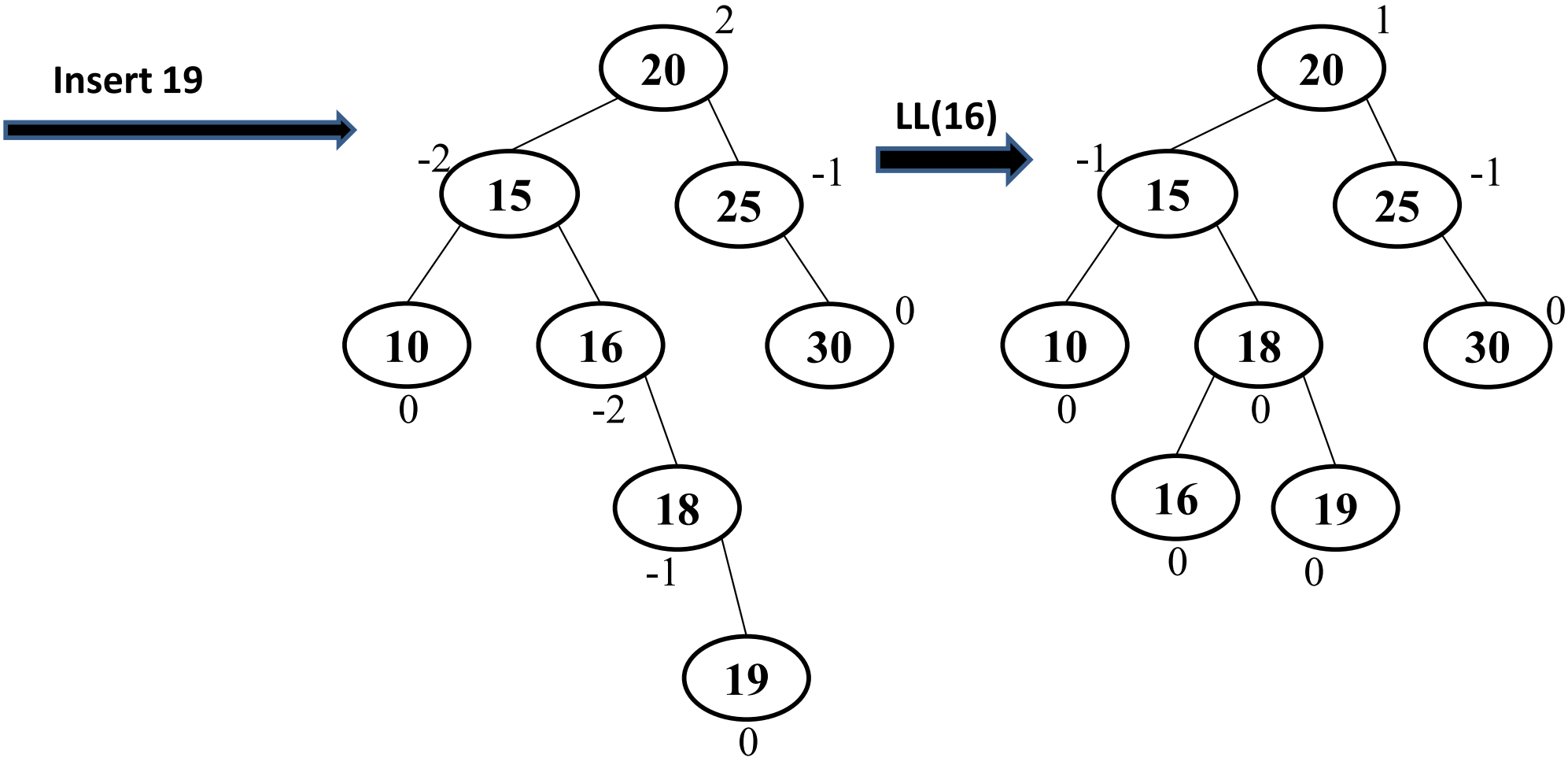
Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL tree



Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL tree



Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL tree



Binary Search Tree Deletion

- Let x be a node to be deleted

Case 1: x is a leaf node

- Delete x

Case 2: x is a node with only one child

- Replace x with the child node of x
- Delete x

Case 3: x having two children

- Find the inorder successor/predessor of x, say y
- Find the right/left child of y, say z
- Replace x by y
- Replace y by z

Binary Search Tree Deletion

- Let x be a node to be deleted

Case 1: x is a leaf node

- **Delete x**

Case 2: x is a node with only one child

- Replace x with the child node of x
- Delete x

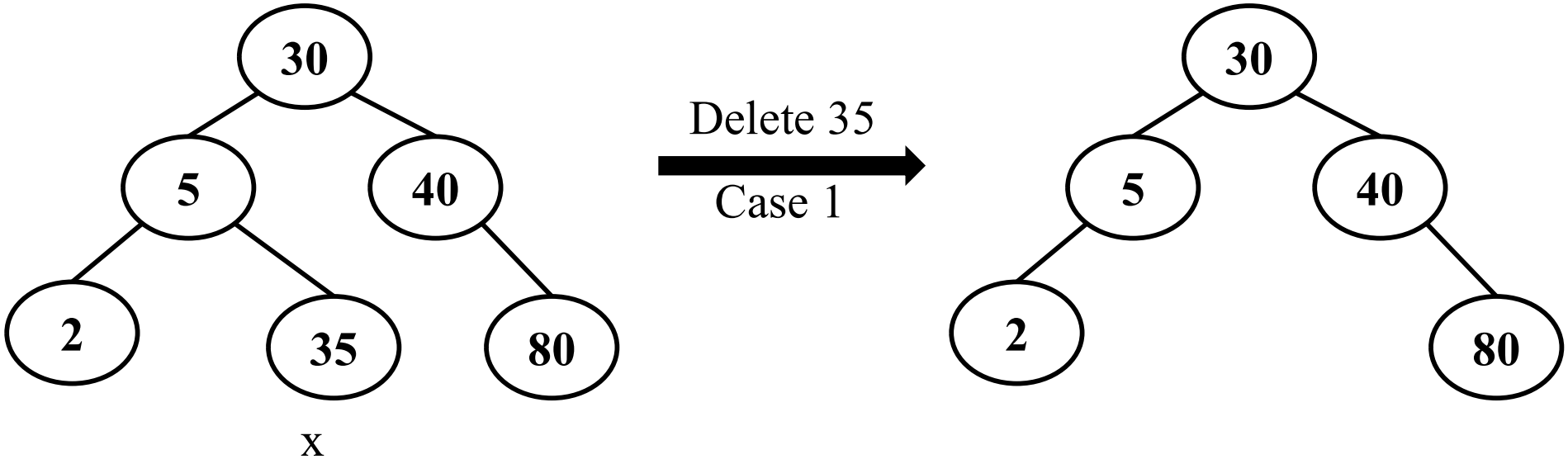
Case 3: x having two children

- Find the inorder successor/predessor of x, say y
- Find the right/left child of y, say z
- Replace x by y
- Replace y by z

Binary Search Tree Deletion

Case 1: x is a leaf node

- Delete x



Binary Search Tree Deletion

- Let x be a node to be deleted

Case 1: x is a leaf node

- Delete x

Case 2: x is a node with only one child

- **Replace x with the child node of x**
- **Delete x**

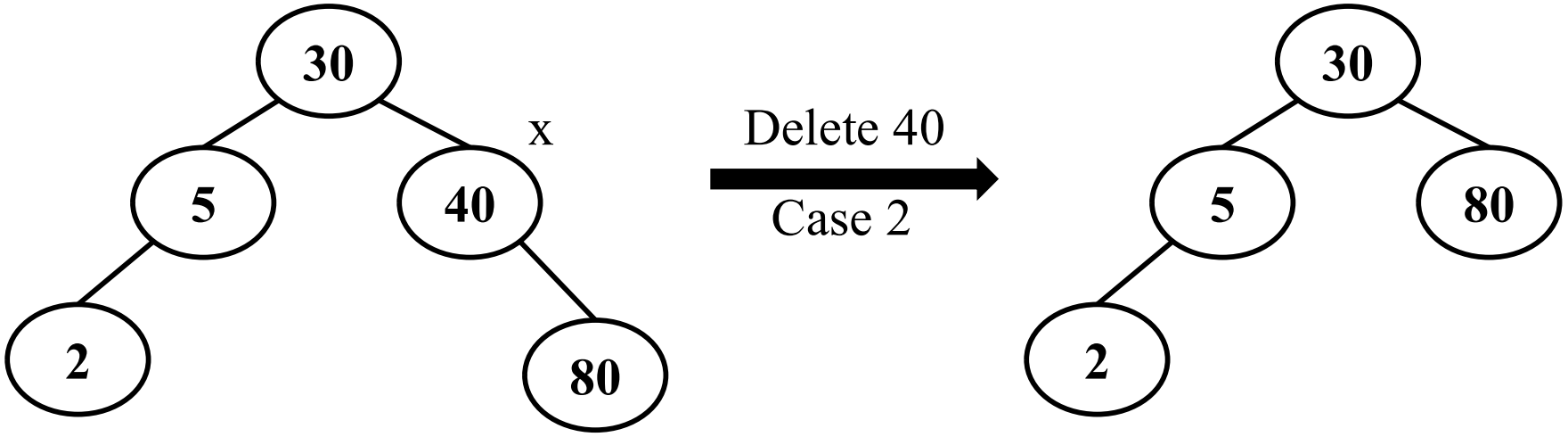
Case 3: x having two children

- Find the inorder successor/predessor of x, say y
- Find the right/left child of y, say z
- Replace x by y
- Replace y by z

Binary Search Tree Deletion

Case 2: x is a node with only one child

- Replace x with the child node of x
- Delete x



Binary Search Tree Deletion

- Let x be a node to be deleted

Case 1: x is a leaf node

- Delete x

Case 2: x is a node with only one child

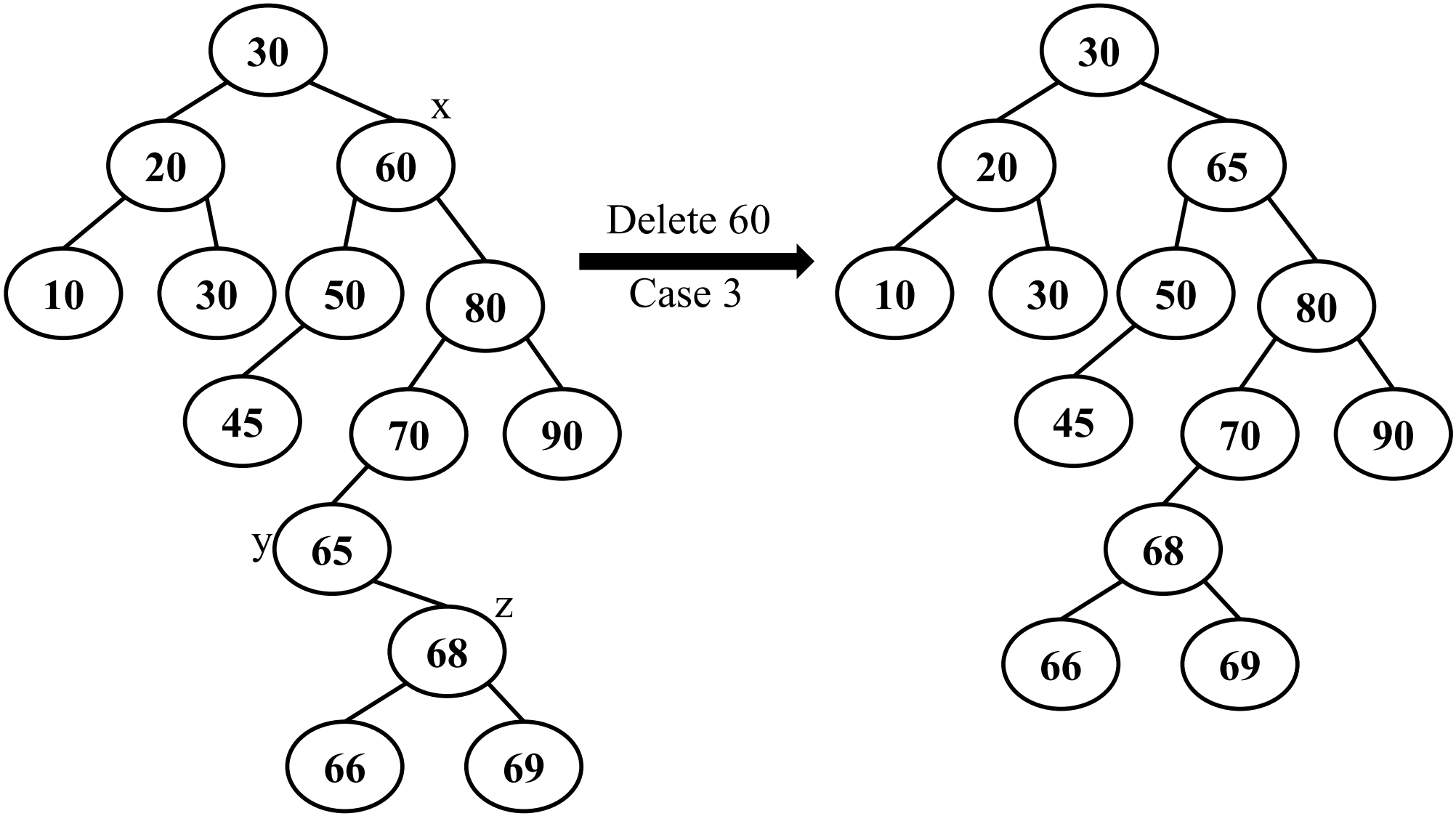
- Replace x with the child node of x
- Delete x

Case 3: x having two children

- **Find the inorder successor/predessor of x, say y**
- **Find the right/left child of y, say z**
- **Replace x by y**
- **Replace y by z**

Case 3: x having two children

- Find the inorder successor/predessor of x, say y
- Find the right/left child of y, say z
- Replace x by y
- Replace y by z



AVL Tree Deletion

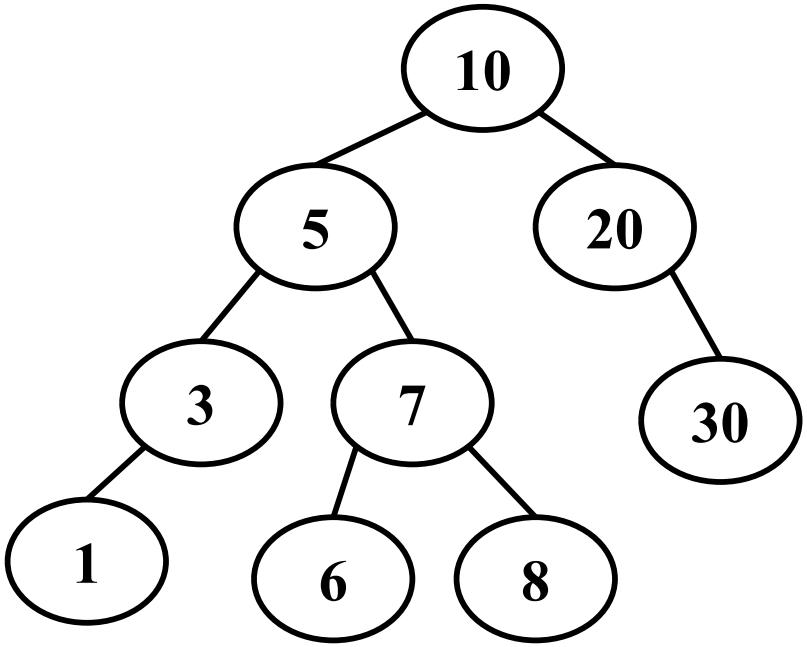
Let w be the node to be deleted

1. Delete w using BST deletion procedure
2. Starting from w, travel up and find the first unbalanced node. Let x be the first unbalanced node.
3. If balance factor(x)>1 then — y=leftchild(x)
4. Else — y=rightchild(x)
5. If y is the leftchild of x
    1. If balance factor(y)≥ 0 then — z=leftchild(y)
    2. Else — z=rightchild(y)
6. Else
    1. If balance factor(y) ≤ 0 then — z=rightchild(y)
    2. Else — z=leftchild(y)

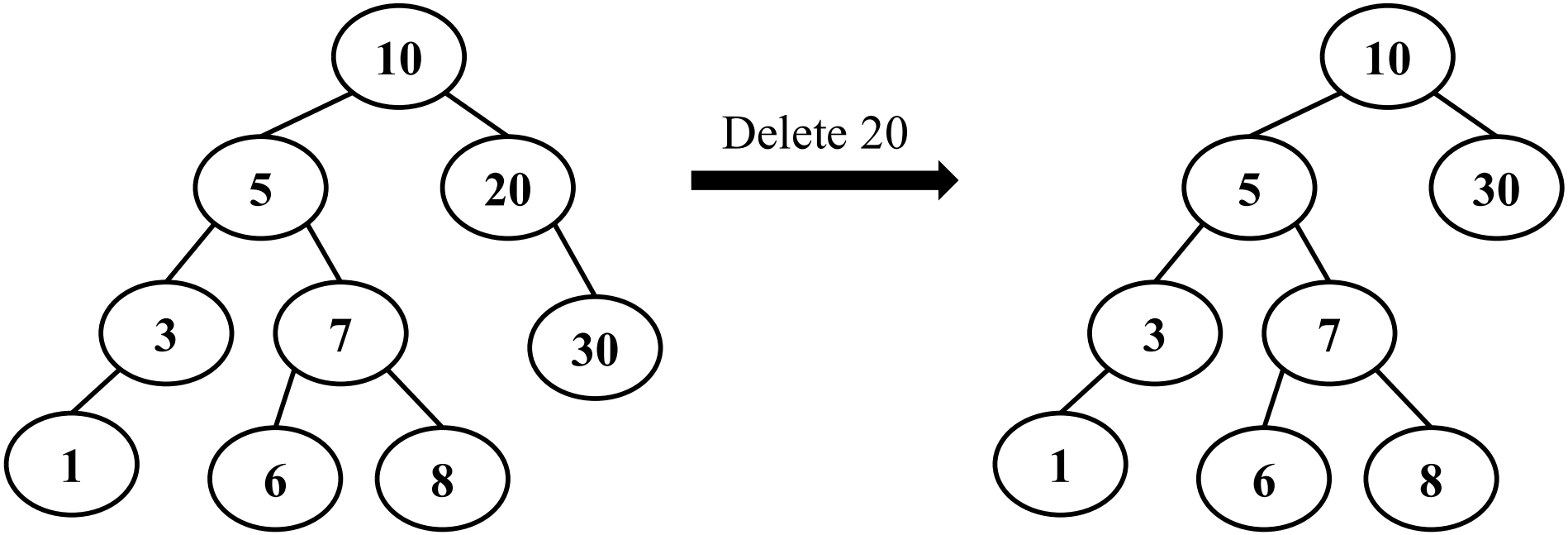
AVL Tree Deletion

7. If y is the left child of x and z is the left child of y, then perform **RR Rotation** with respect to x
8. If y is the right child of x and z is the right child of y, then perform **LL Rotation** with respect to x
9. If y is the left child of x and z is the right child of y, then perform **LR Rotation** with respect to x
10. If y is the right child of x and z is the left child of y, then perform **RL Rotation** with respect to x

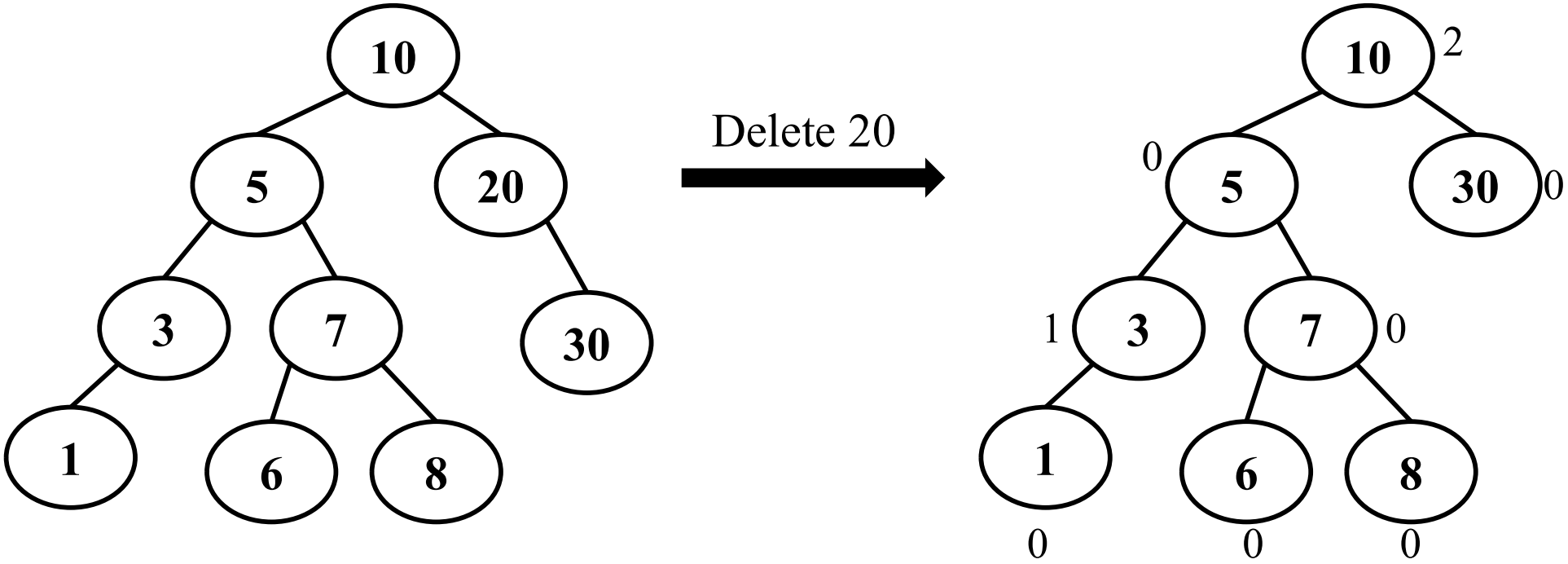
Delete 20 from the given AVL Tree



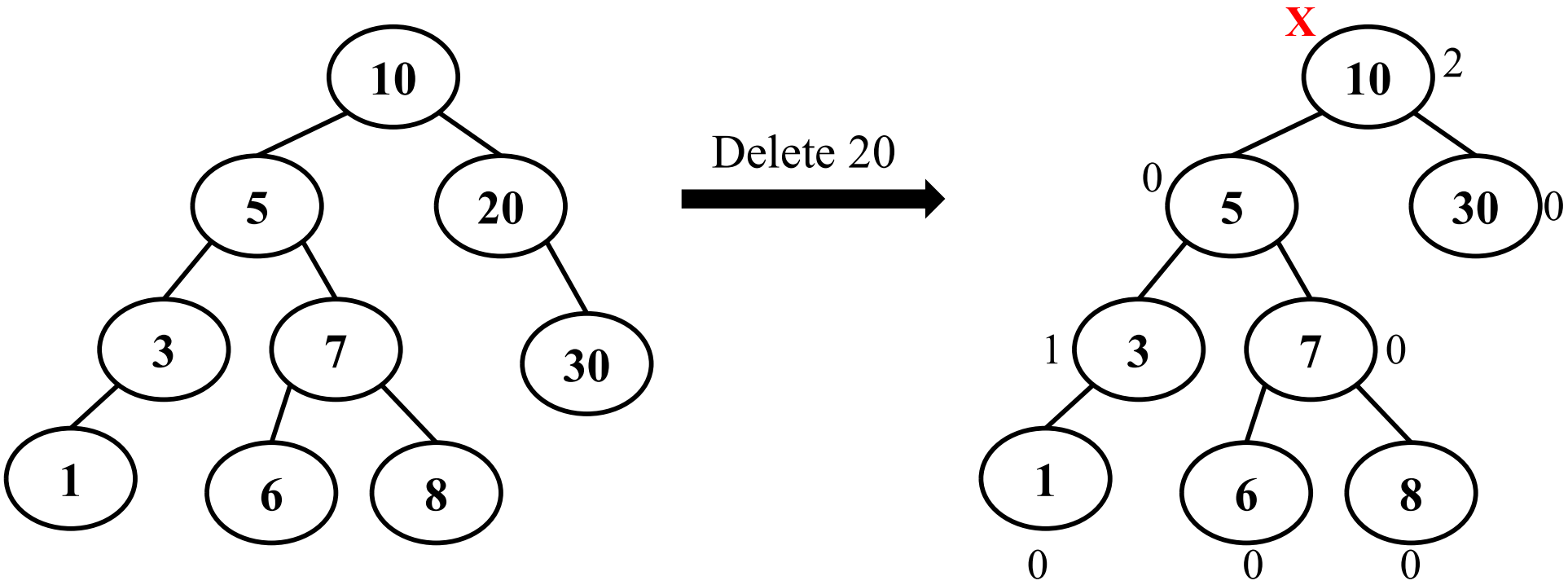
Delete 20 from the given AVL Tree



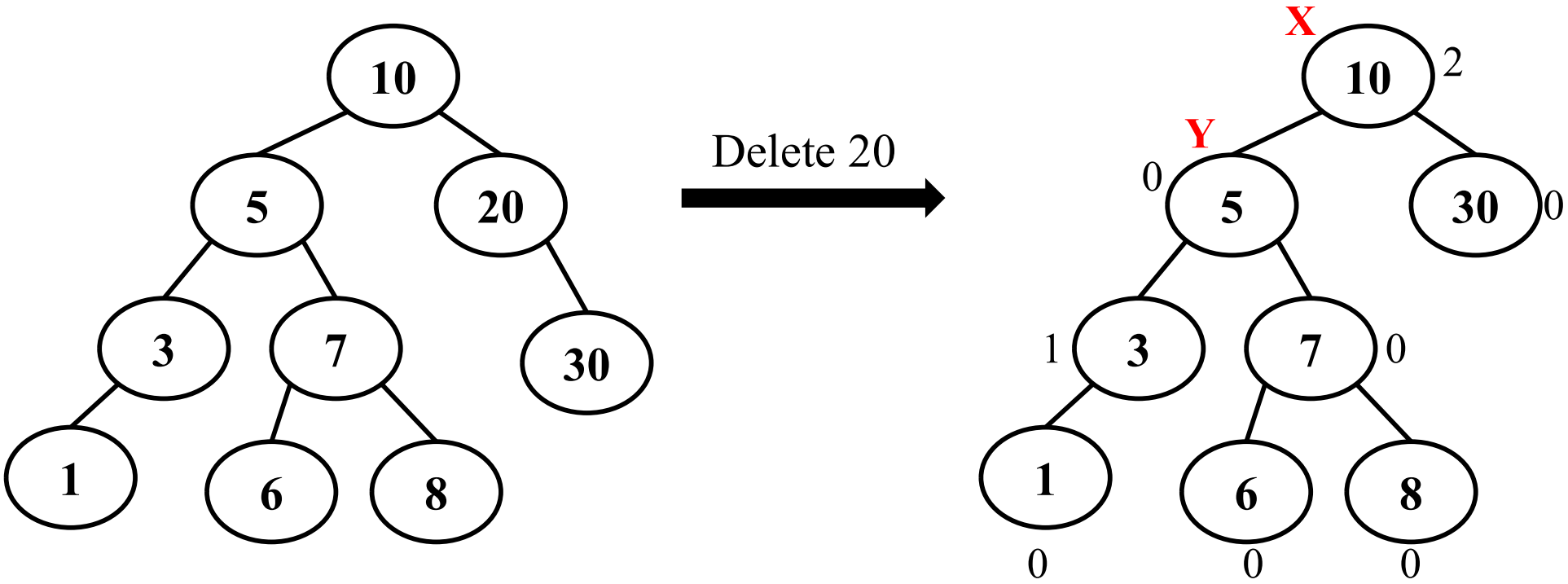
Delete 20 from the given AVL Tree



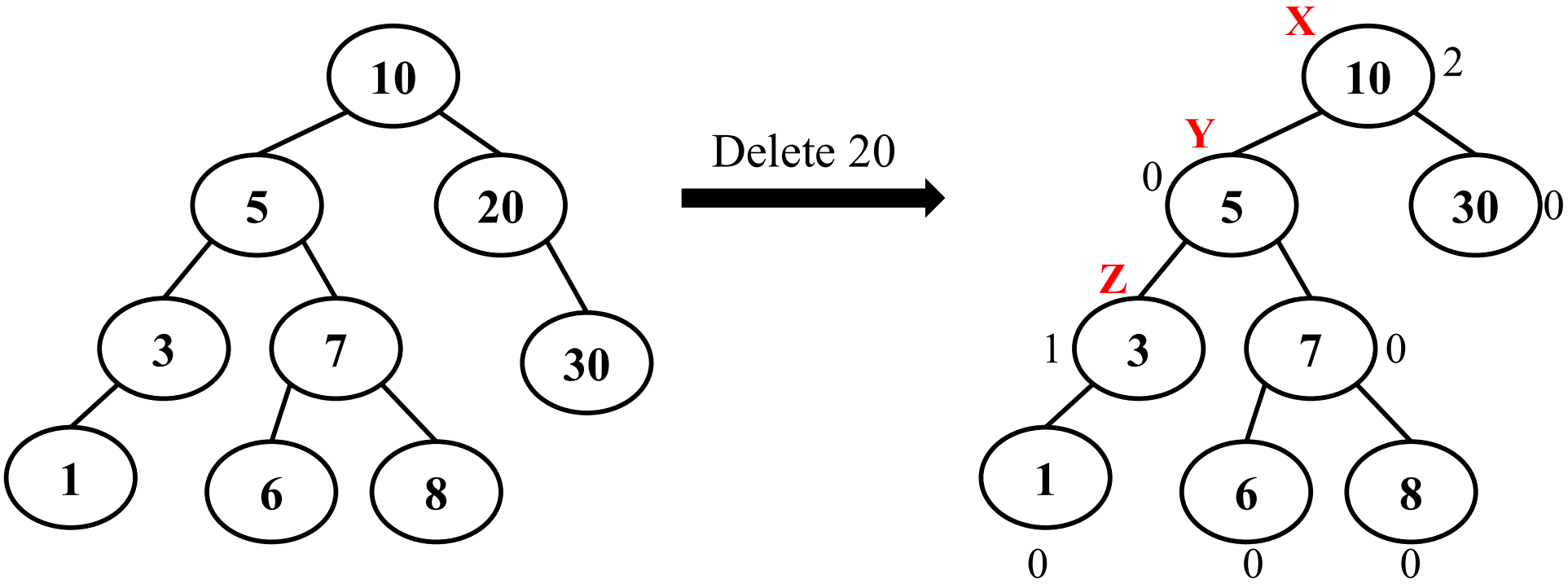
Delete 20 from the given AVL Tree



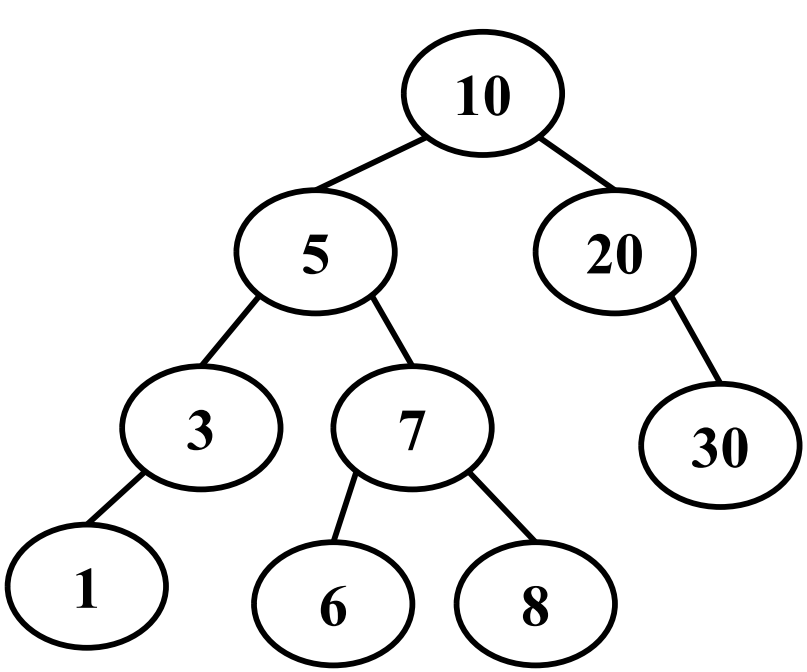
Delete 20 from the given AVL Tree



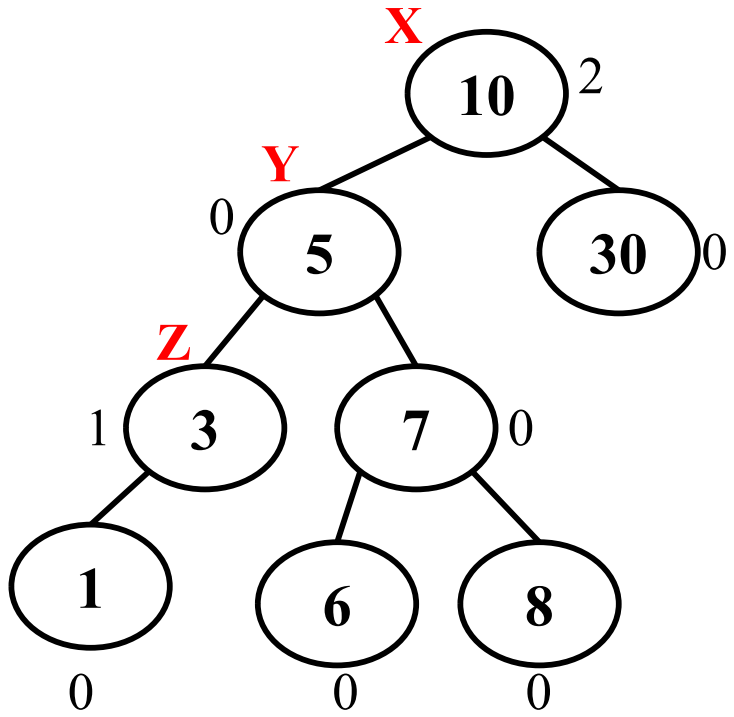
Delete 20 from the given AVL Tree



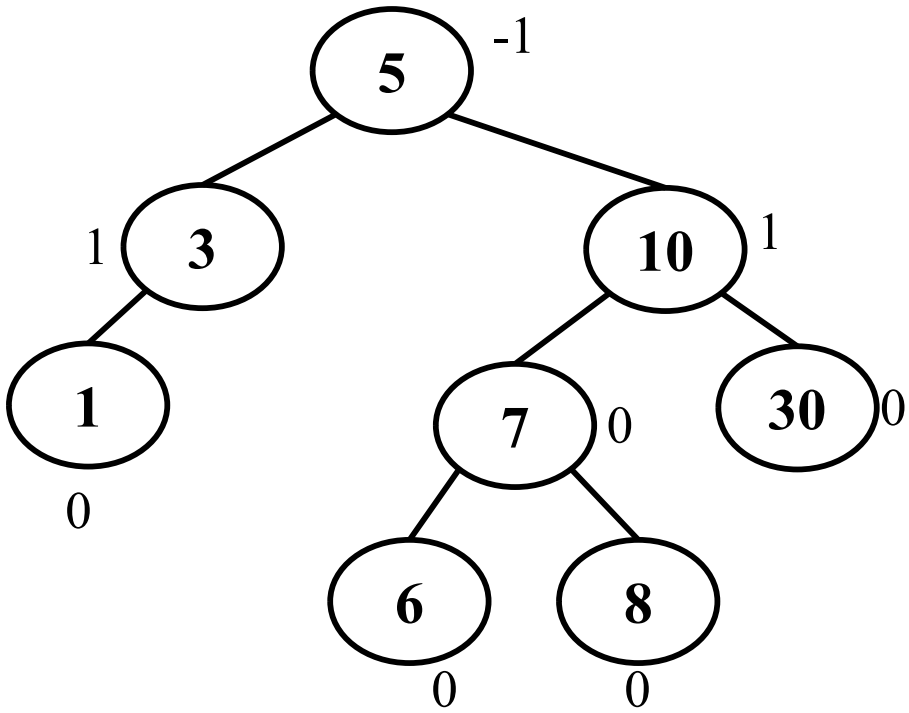
Delete 20 from the given AVL Tree



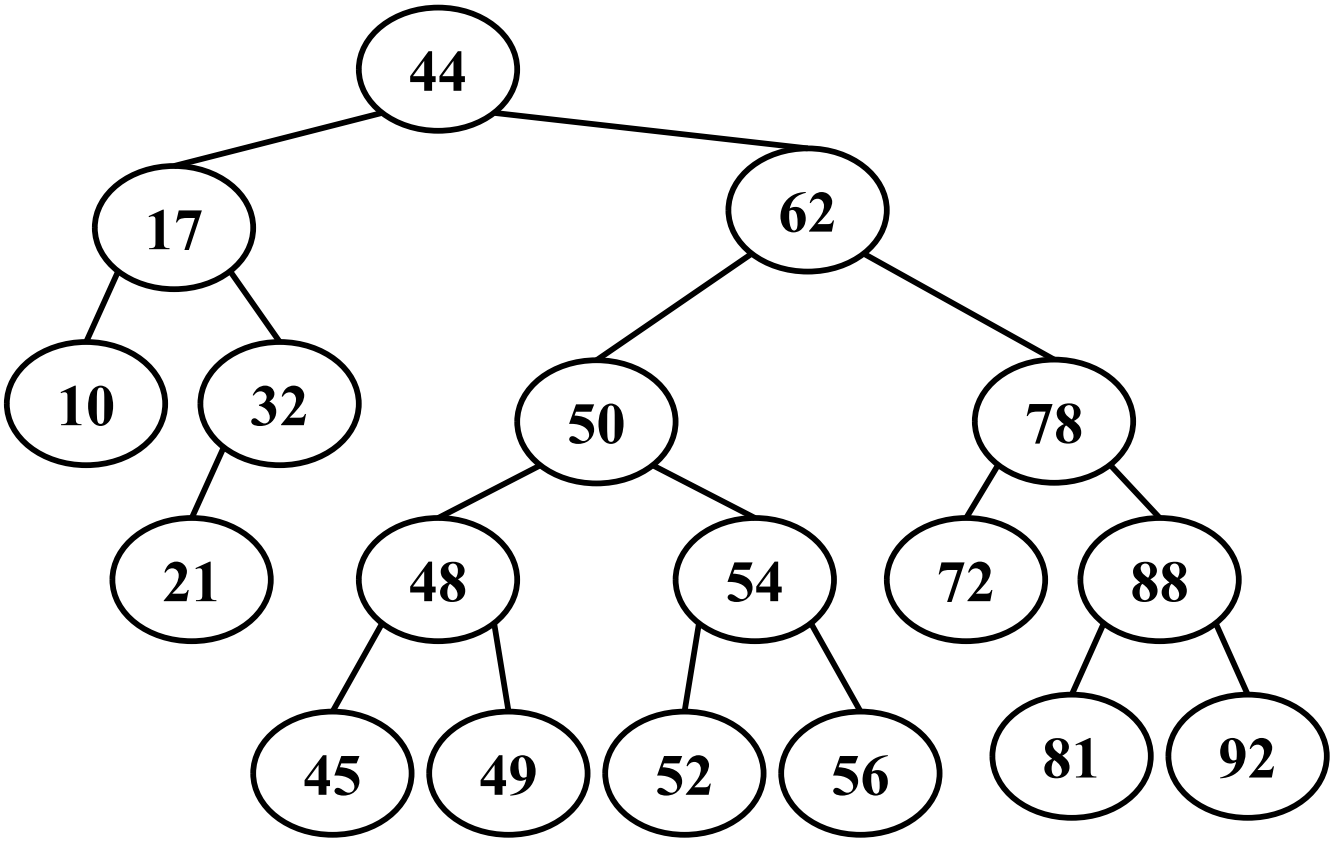
Delete 20



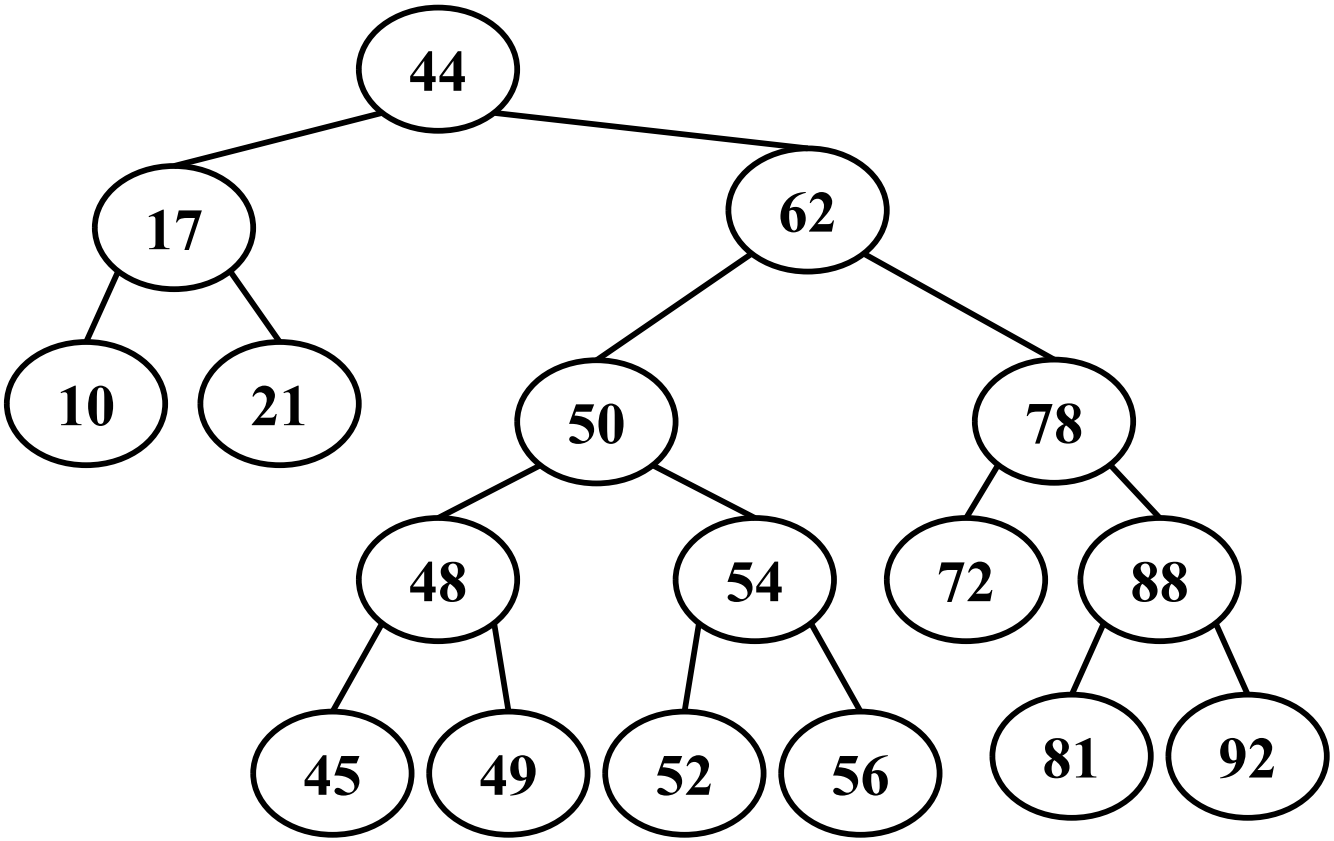
RR(X)



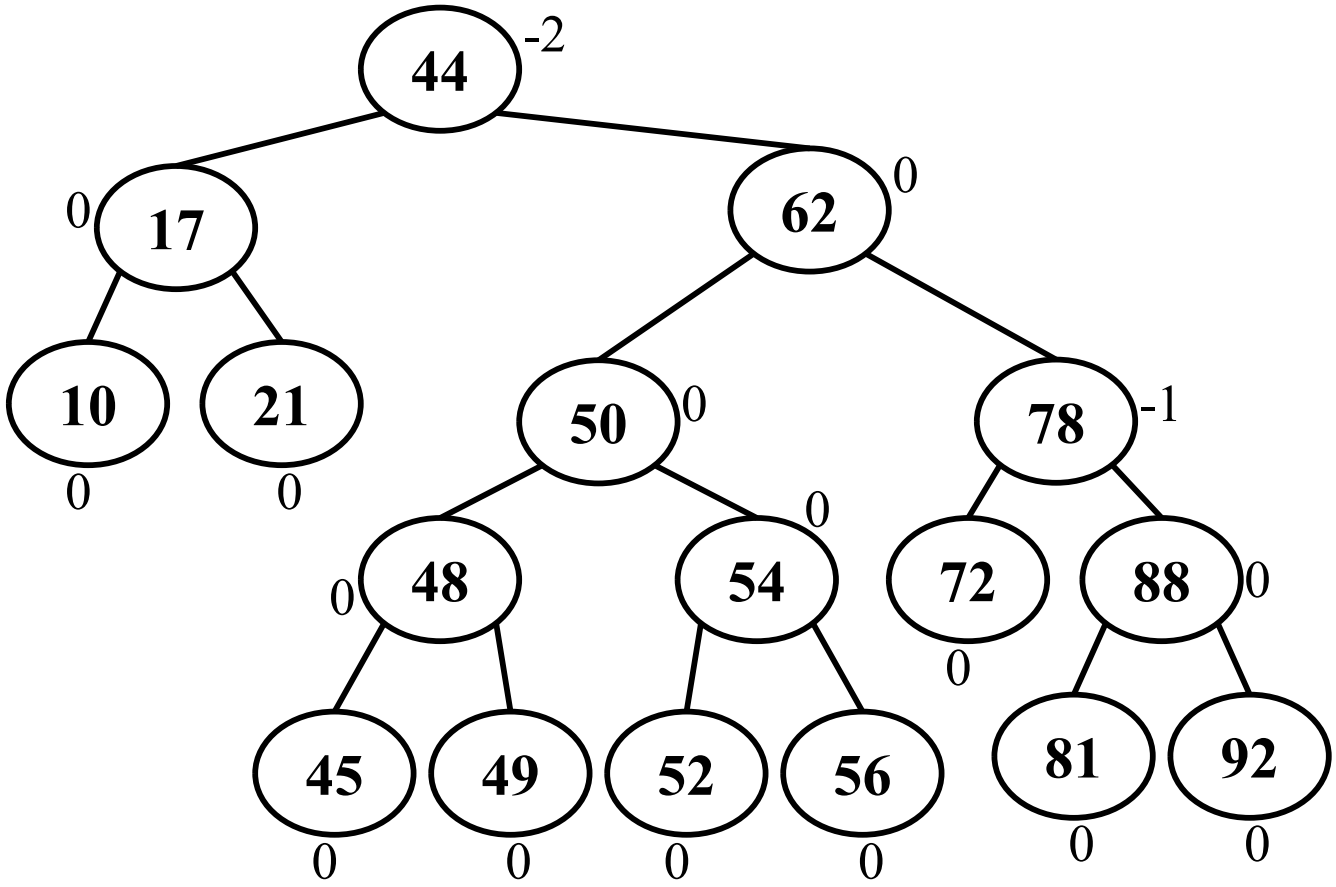
Delete 32 from the given AVL Tree



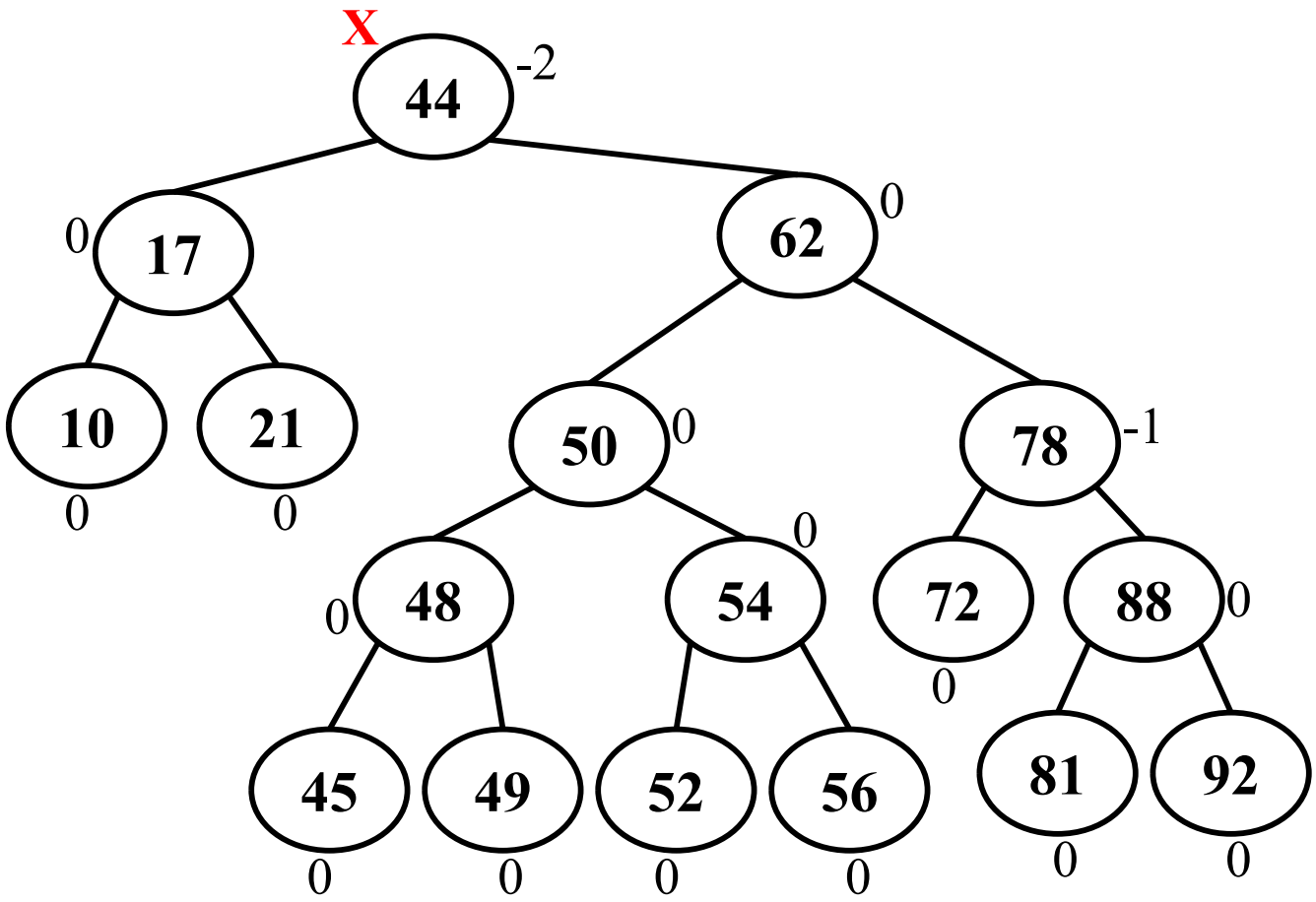
Delete 32 from the given AVL Tree



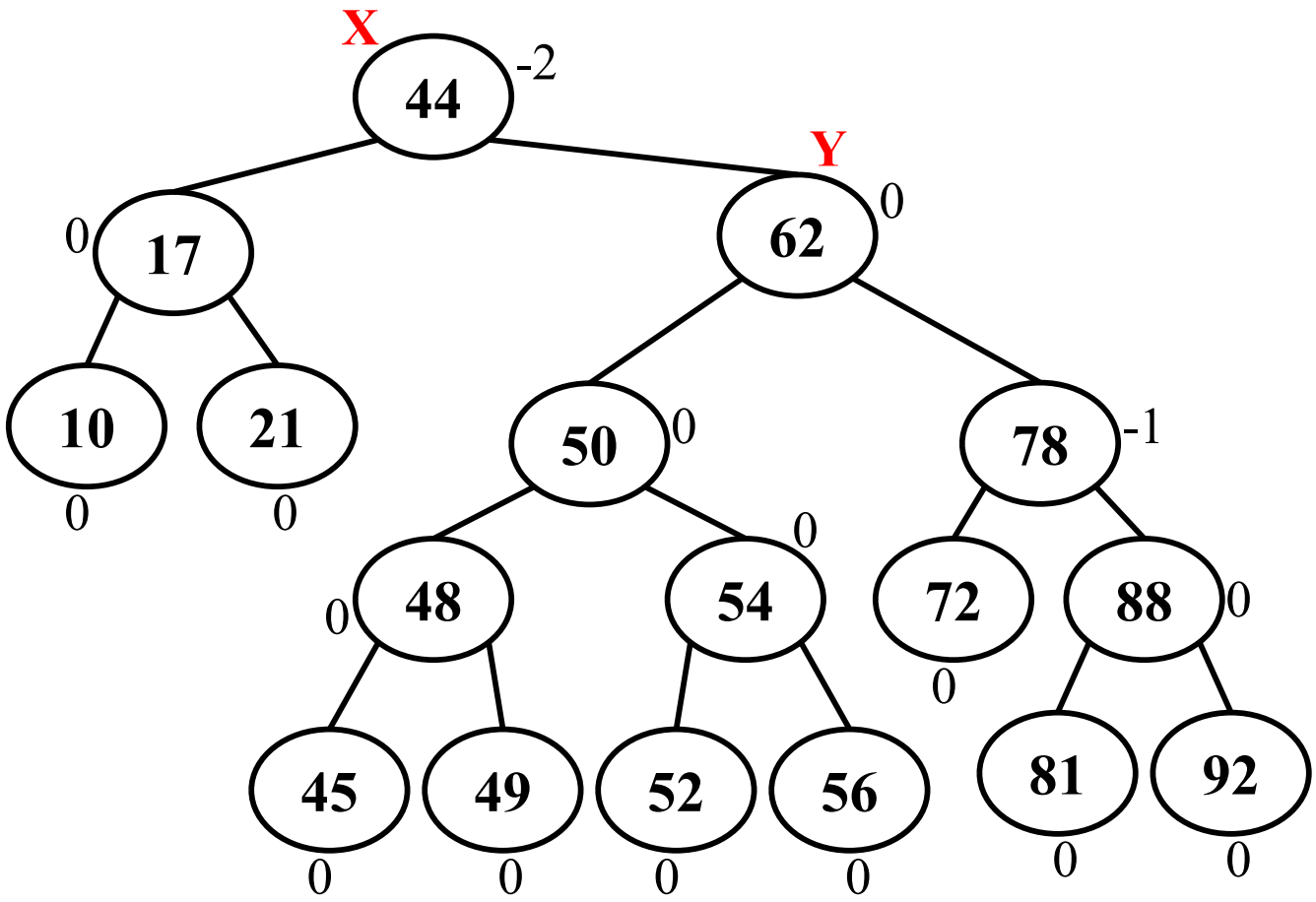
Delete 32 from the given AVL Tree



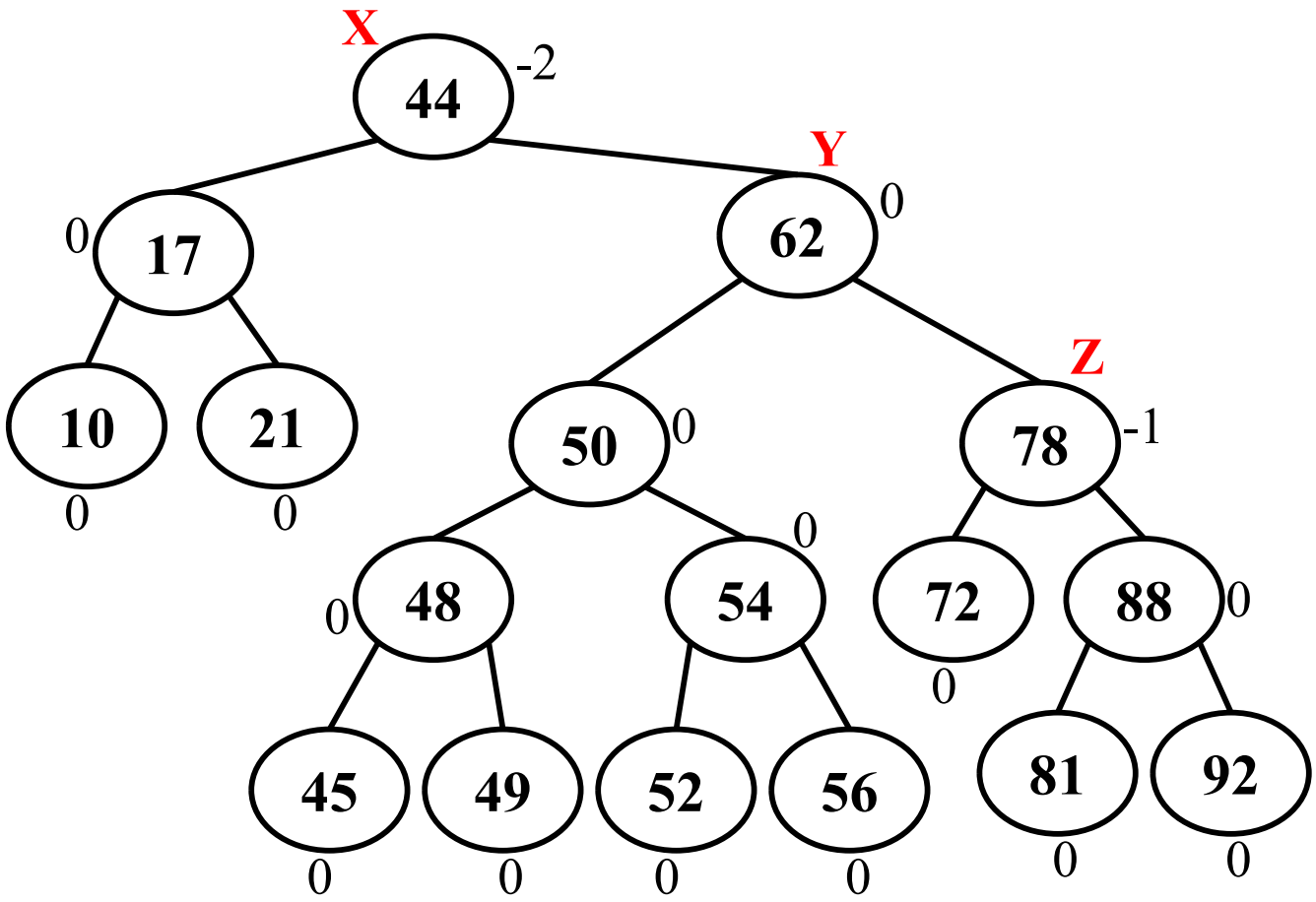
Delete 32 from the given AVL Tree



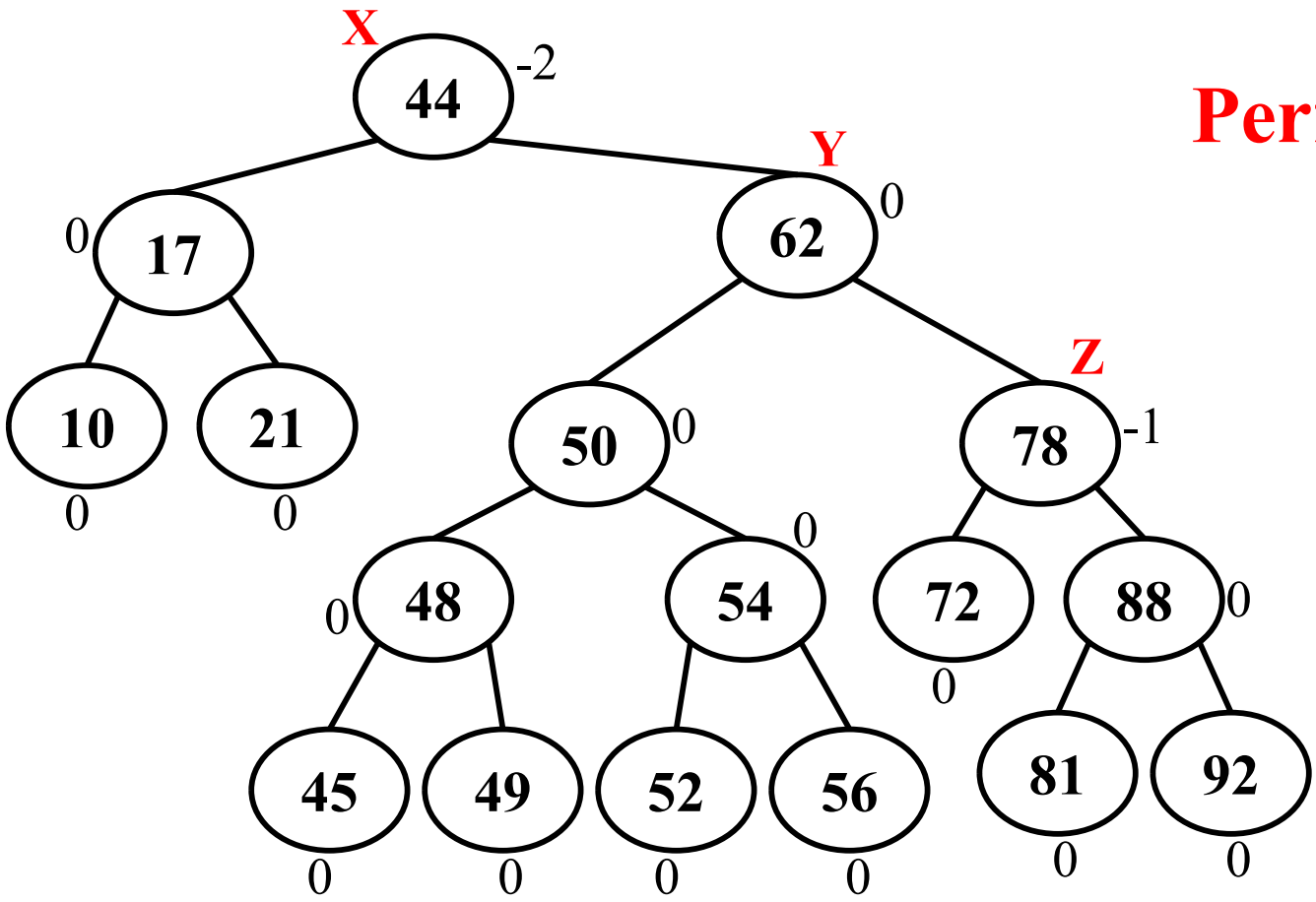
Delete 32 from the given AVL Tree



Delete 32 from the given AVL Tree

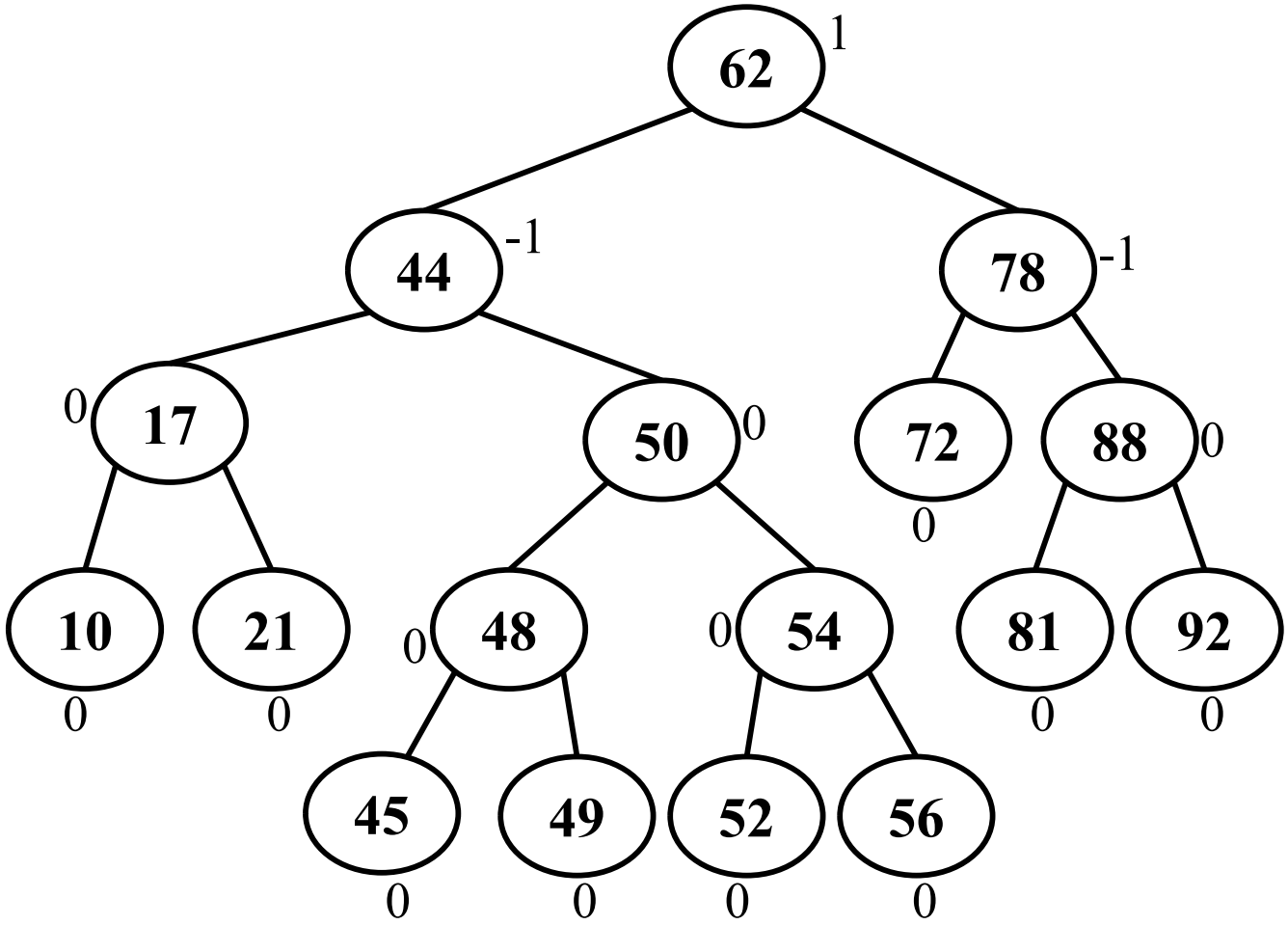


Delete 32 from the given AVL Tree

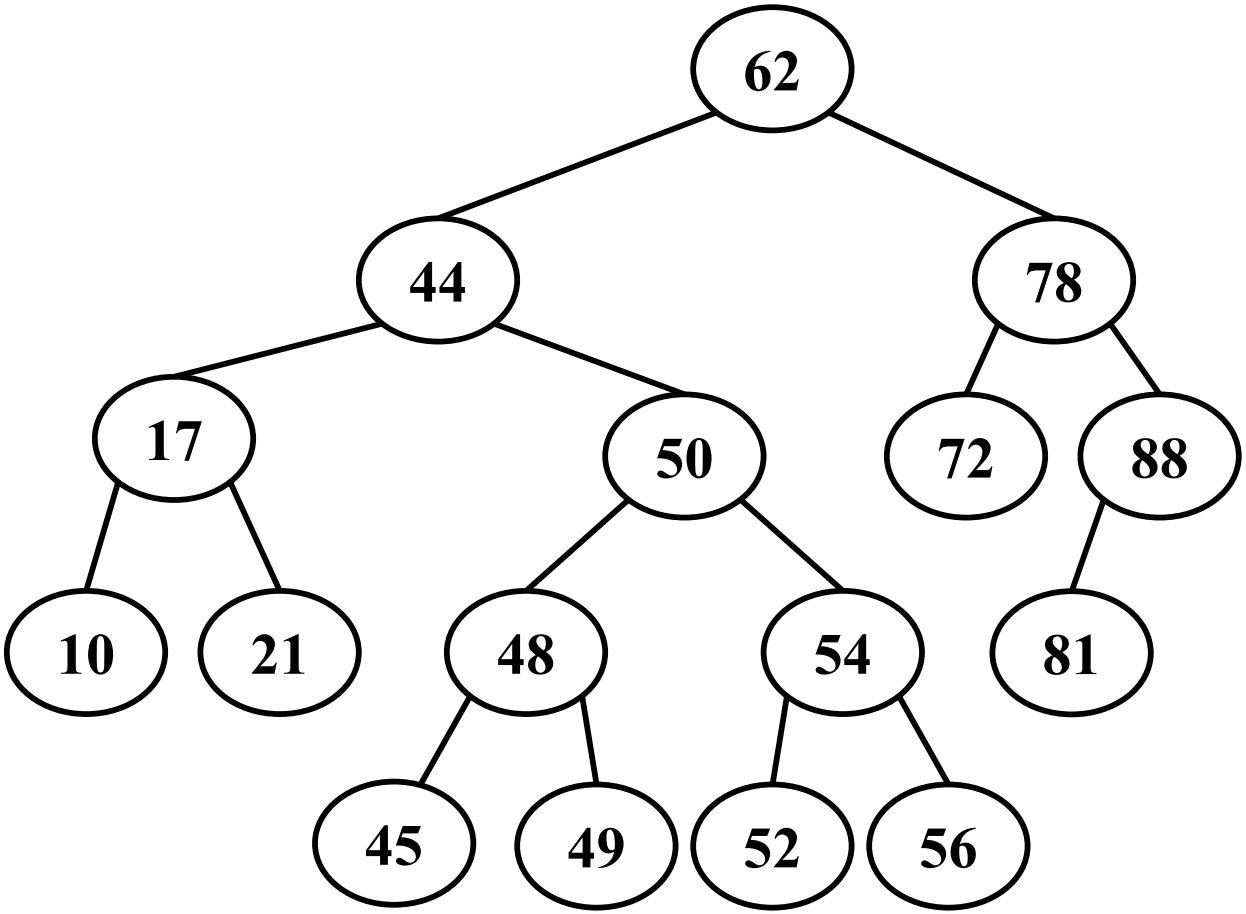


Perform LL(X)

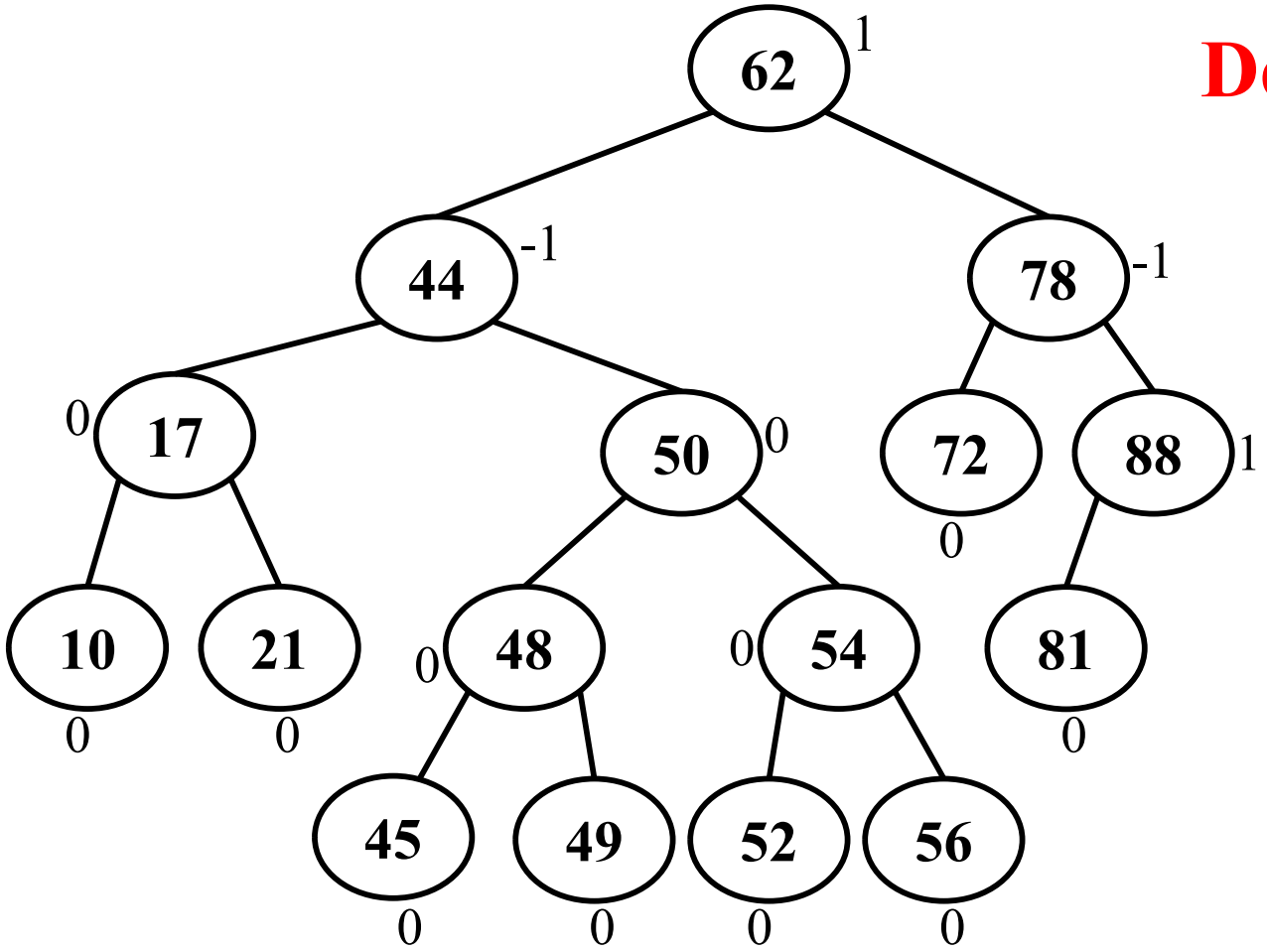
Delete 32 from the given AVL Tree



Delete 78 from the given AVL Tree

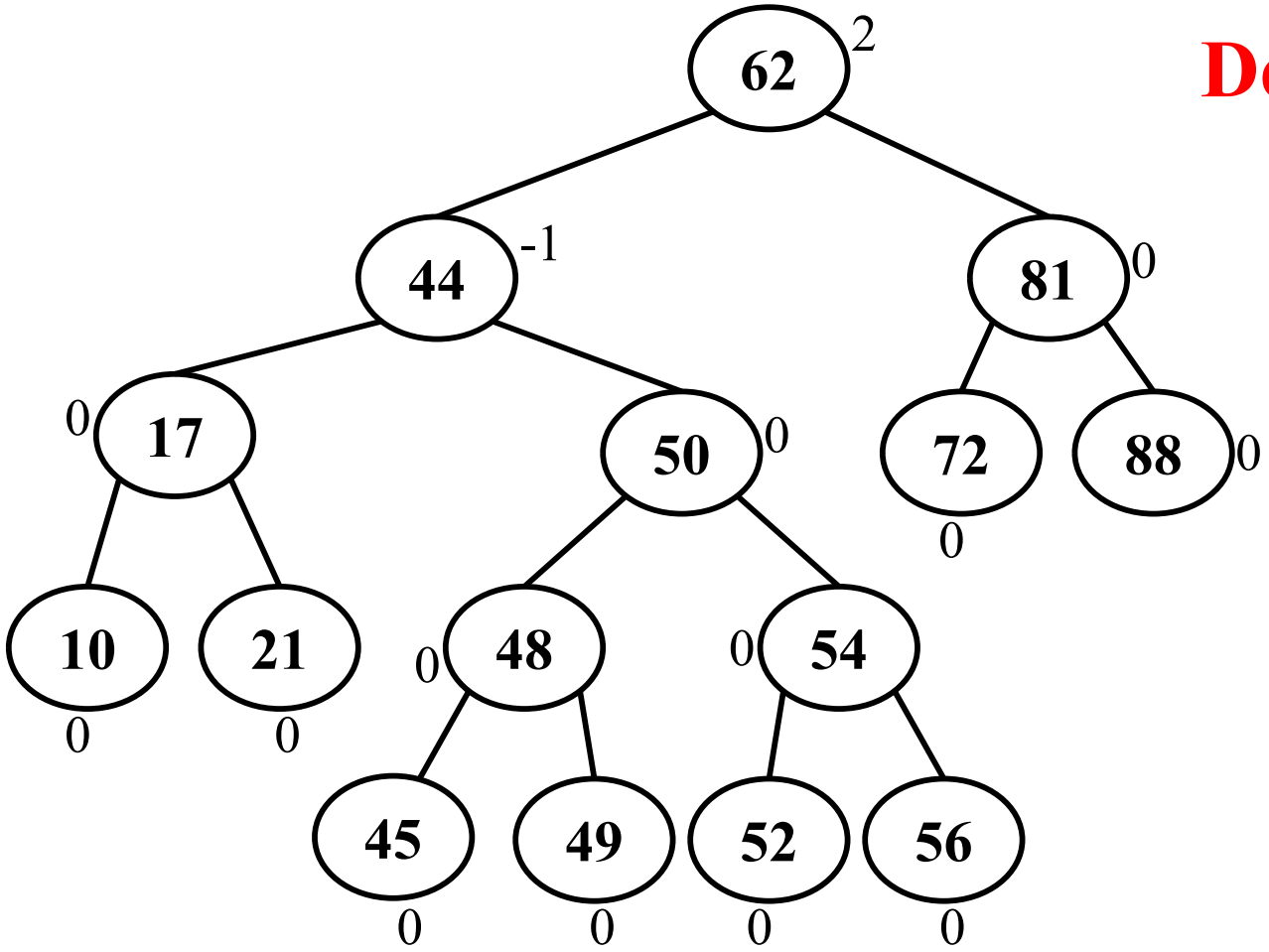


Delete 78 from the given AVL Tree



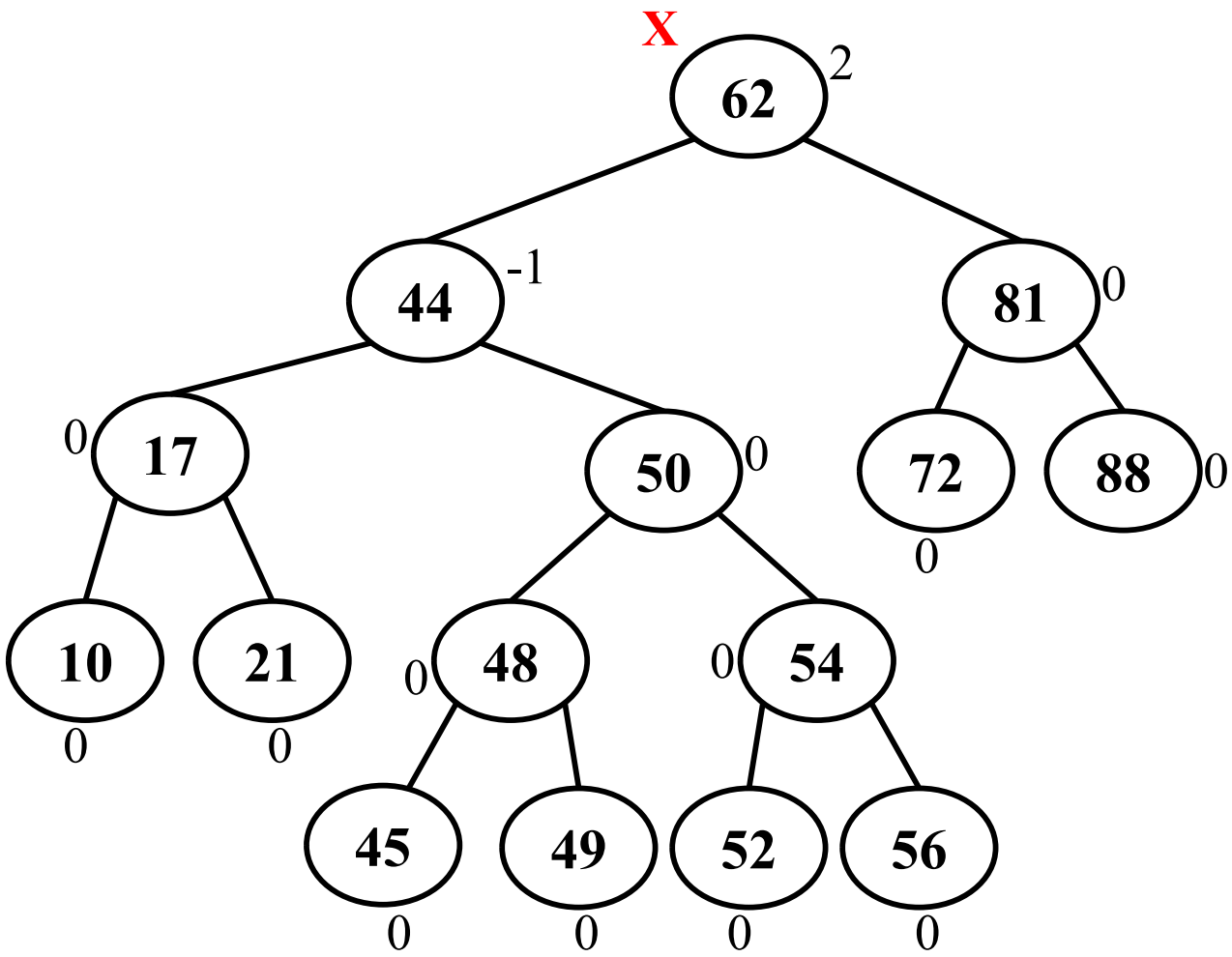
Delete 78

Delete 78 from the given AVL Tree

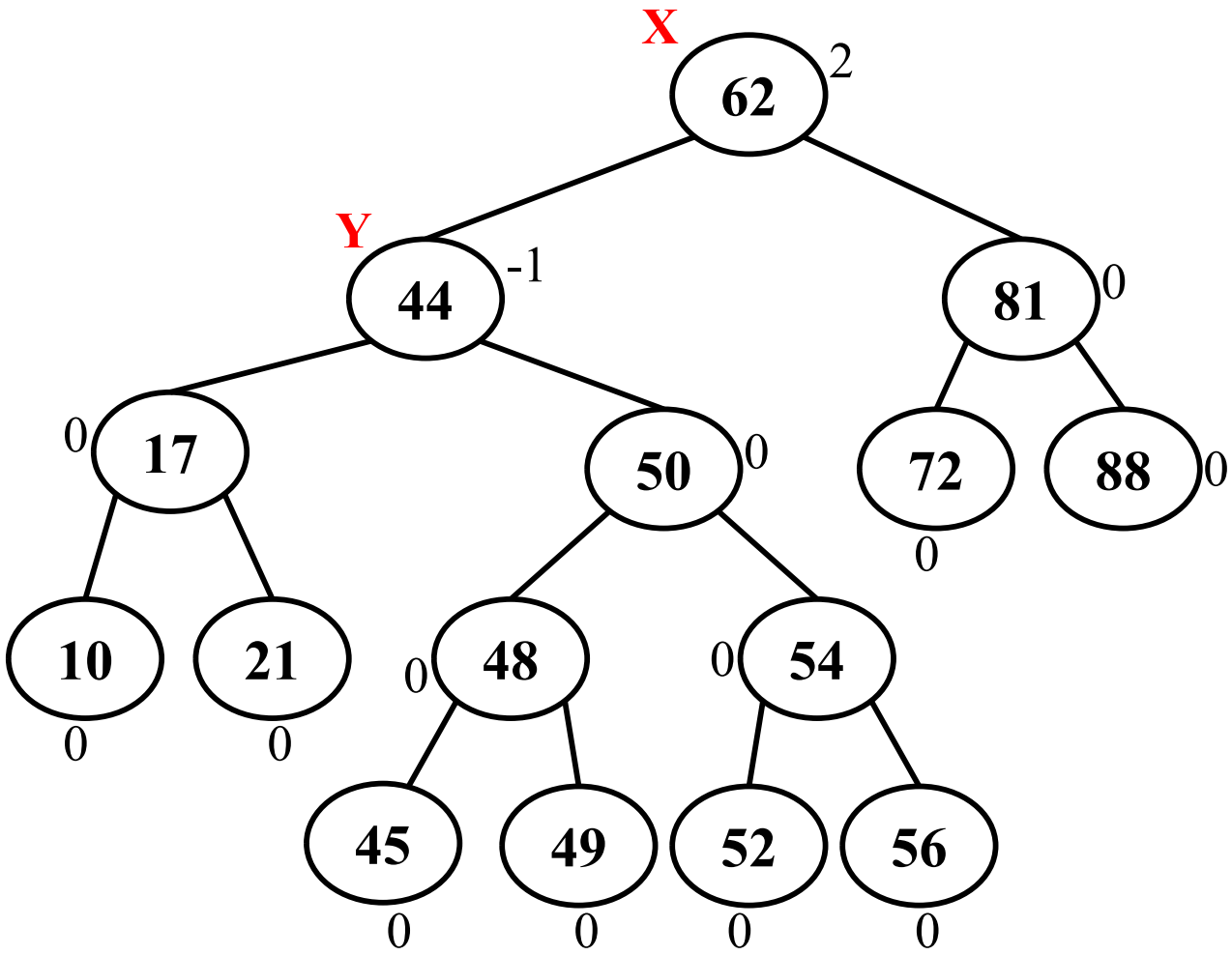


Delete 78

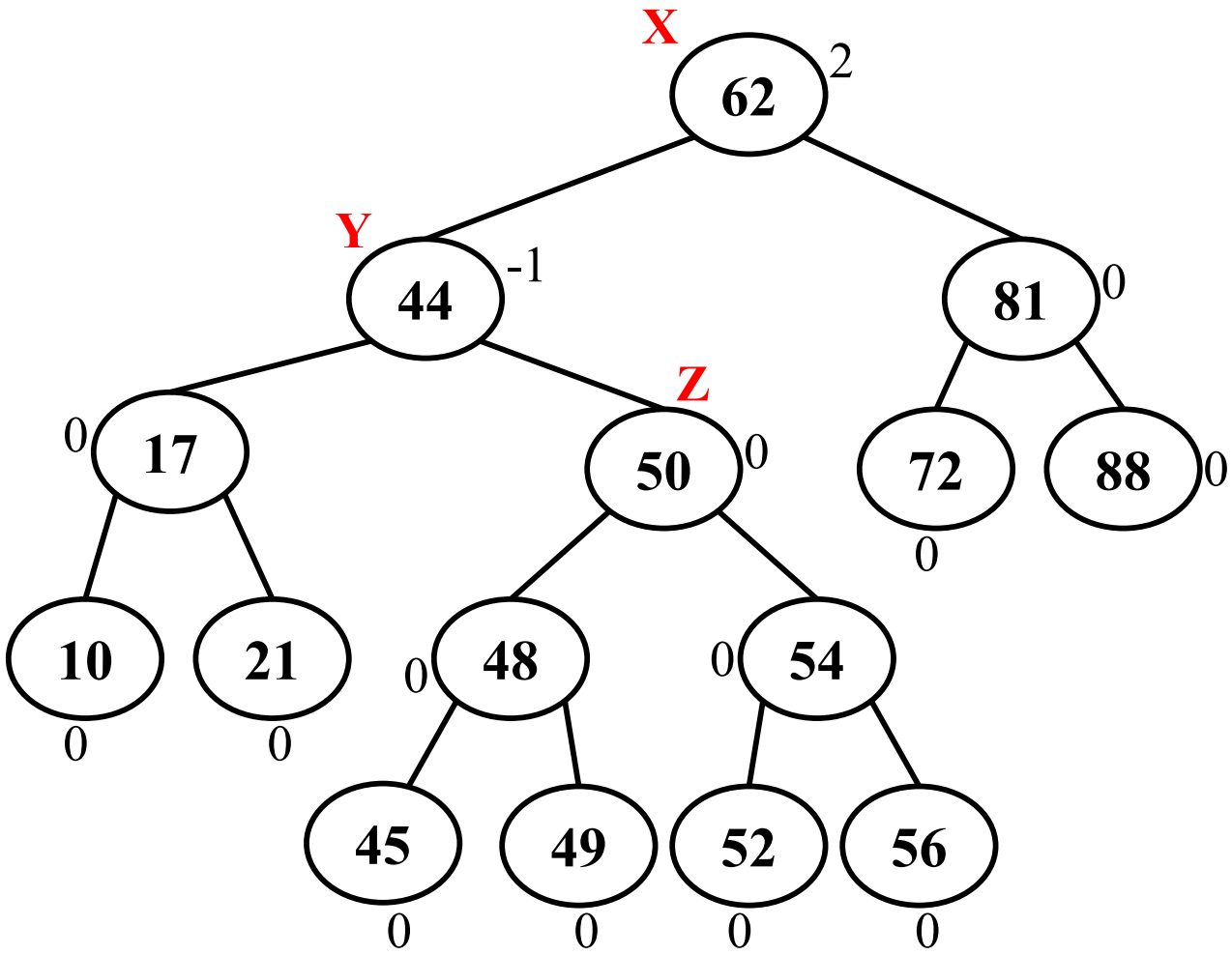
Delete 78 from the given AVL Tree



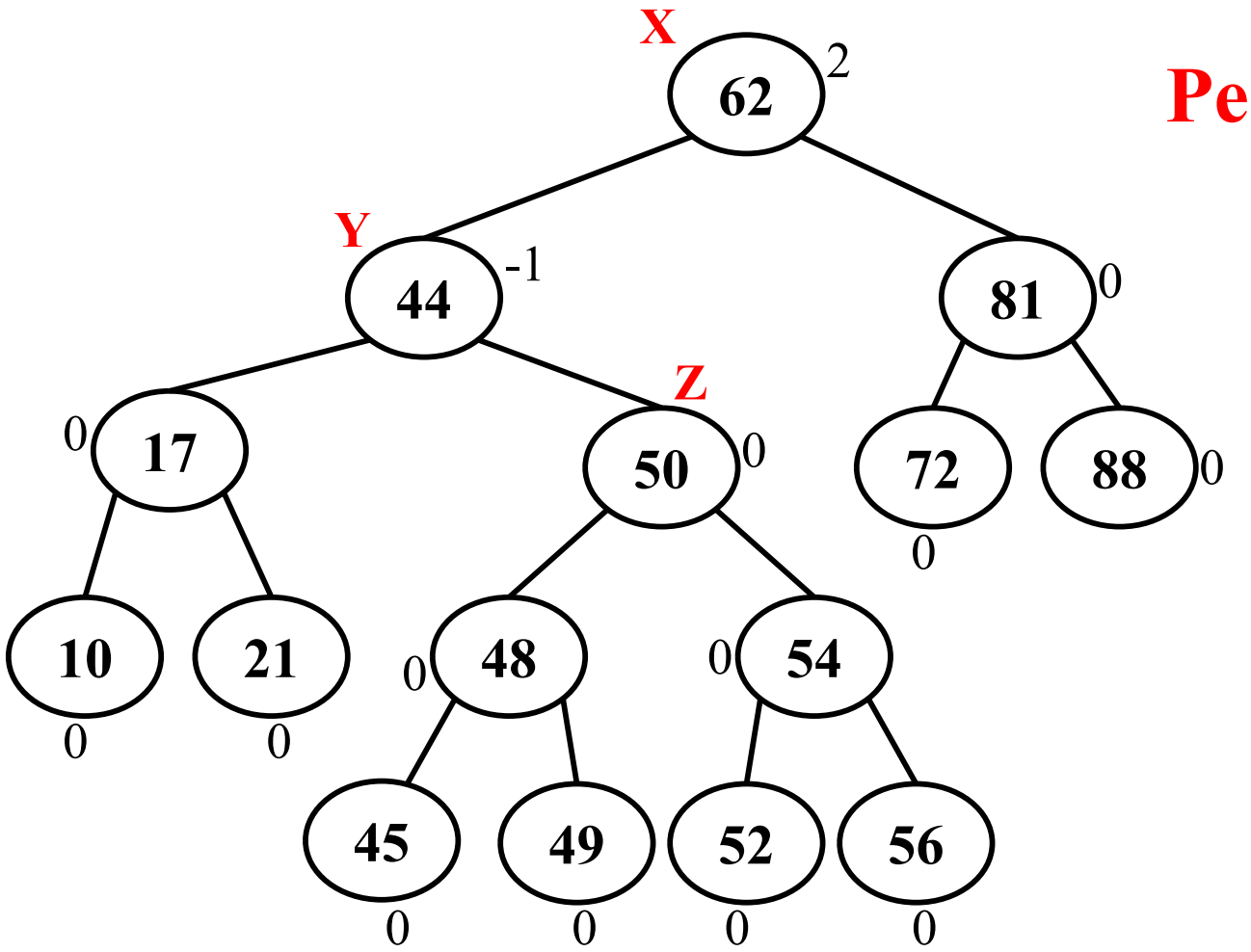
Delete 78 from the given AVL Tree



Delete 78 from the given AVL Tree

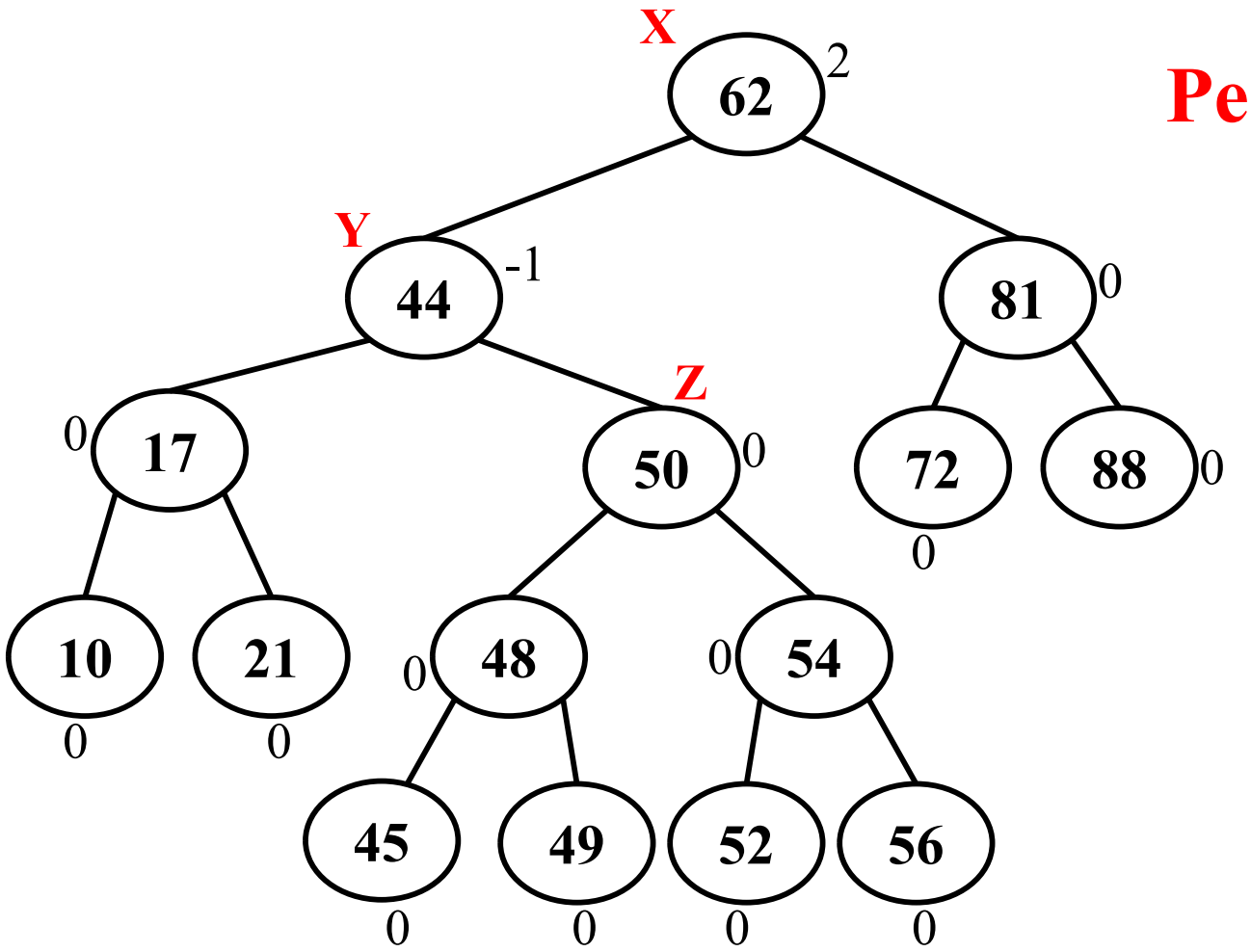


Delete 78 from the given AVL Tree



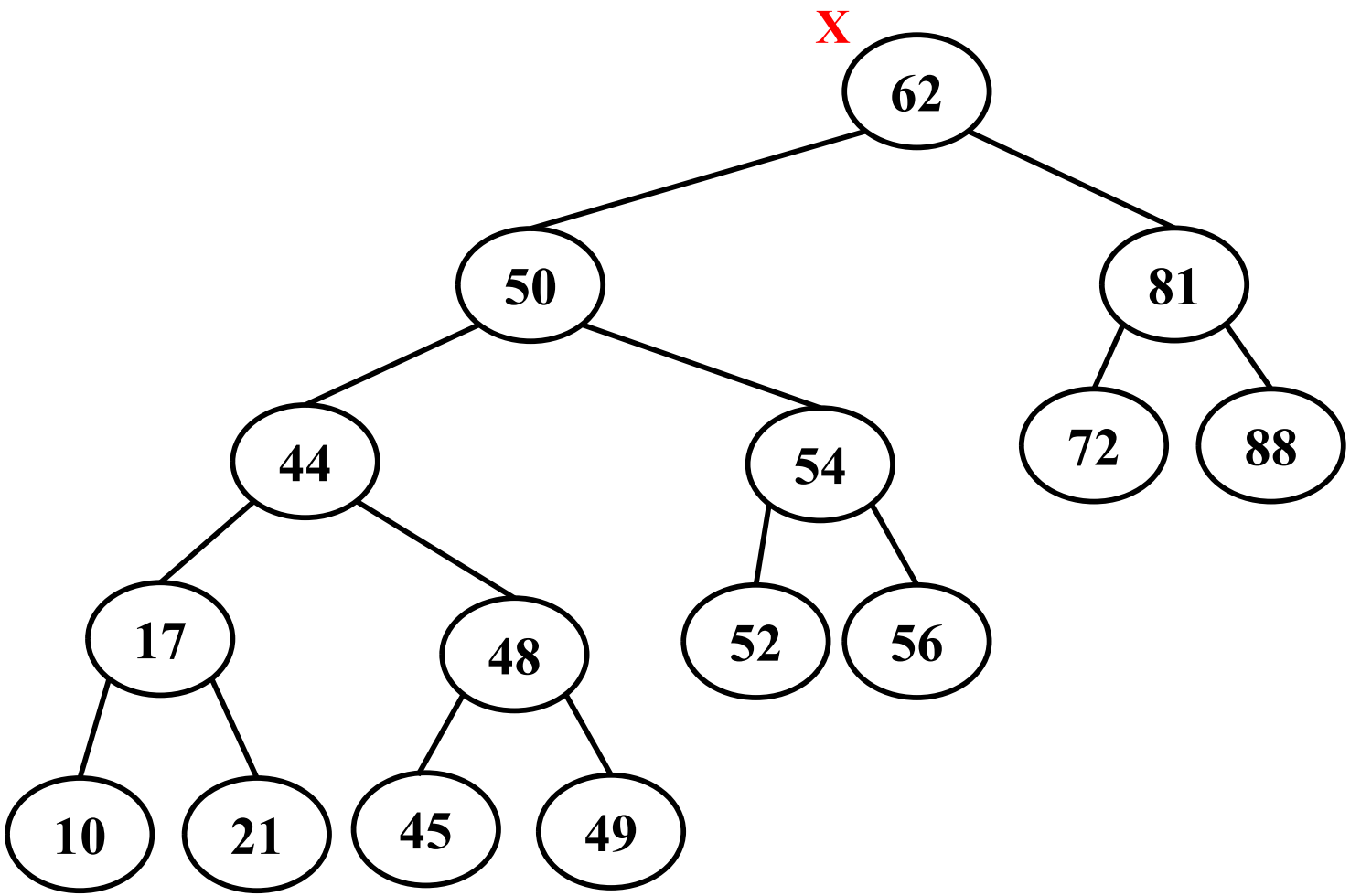
Perform LR(X)

Delete 78 from the given AVL Tree

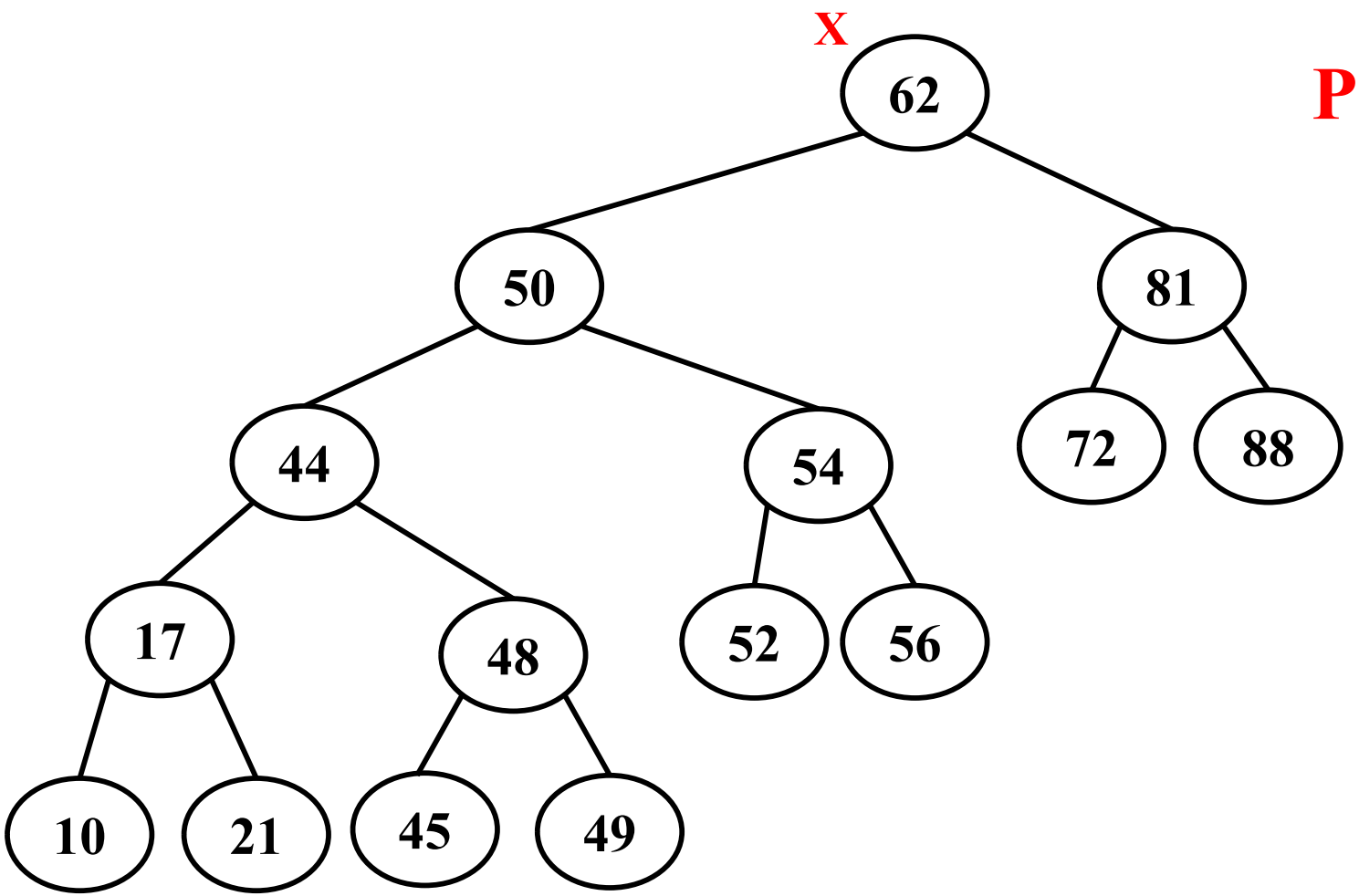


Perform LL(Y)

Delete 78 from the given AVL Tree

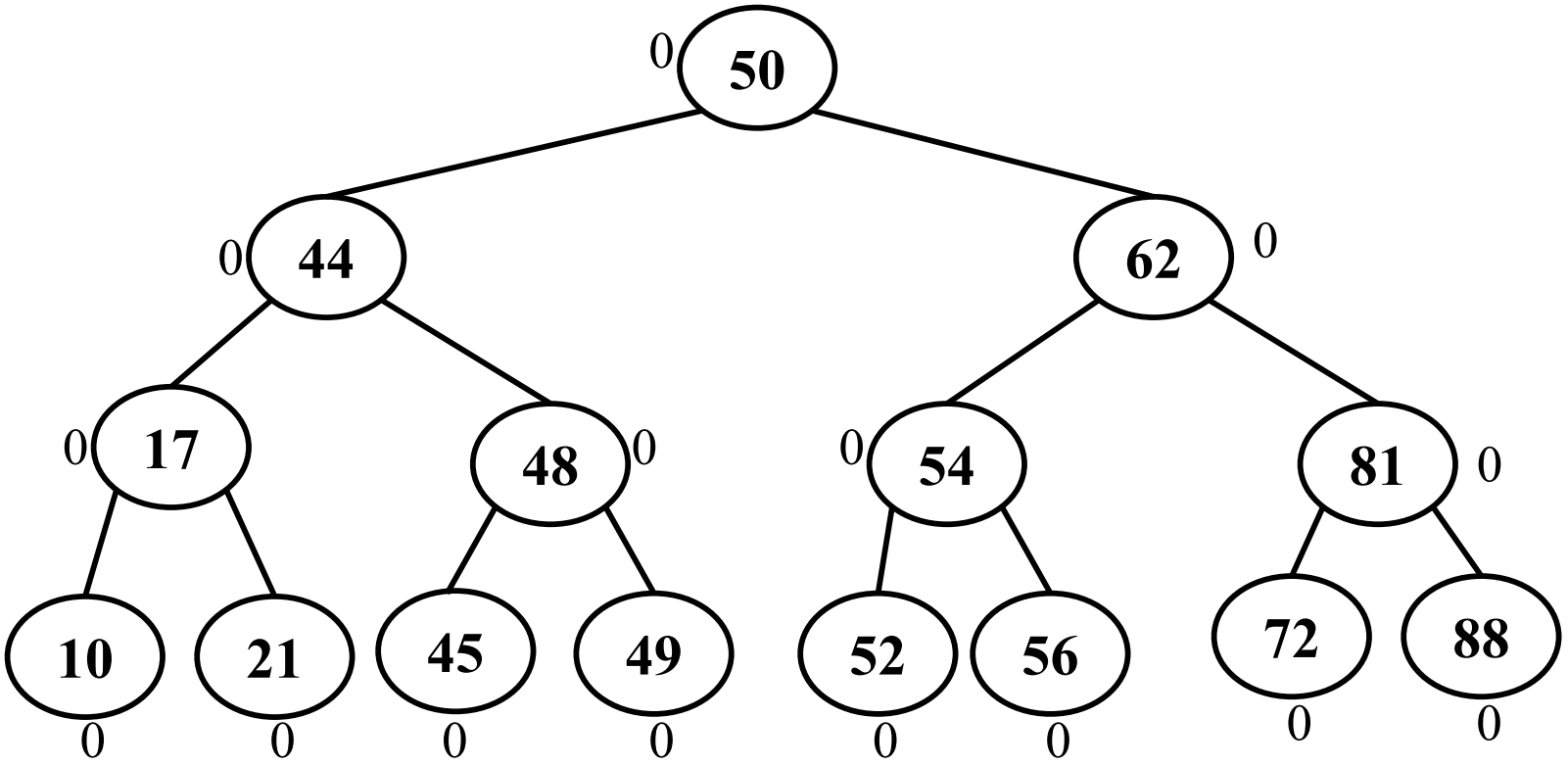


Delete 78 from the given AVL Tree

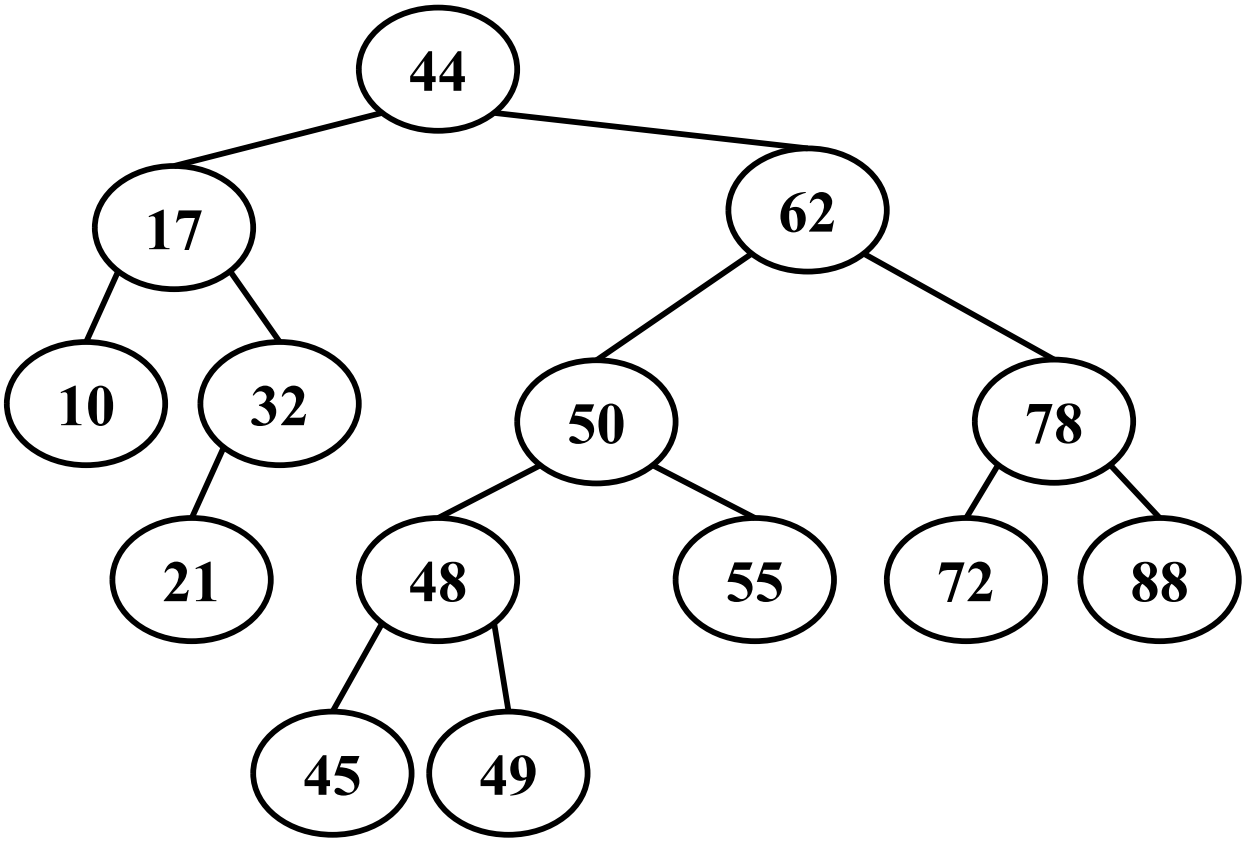


Perform RR(X)

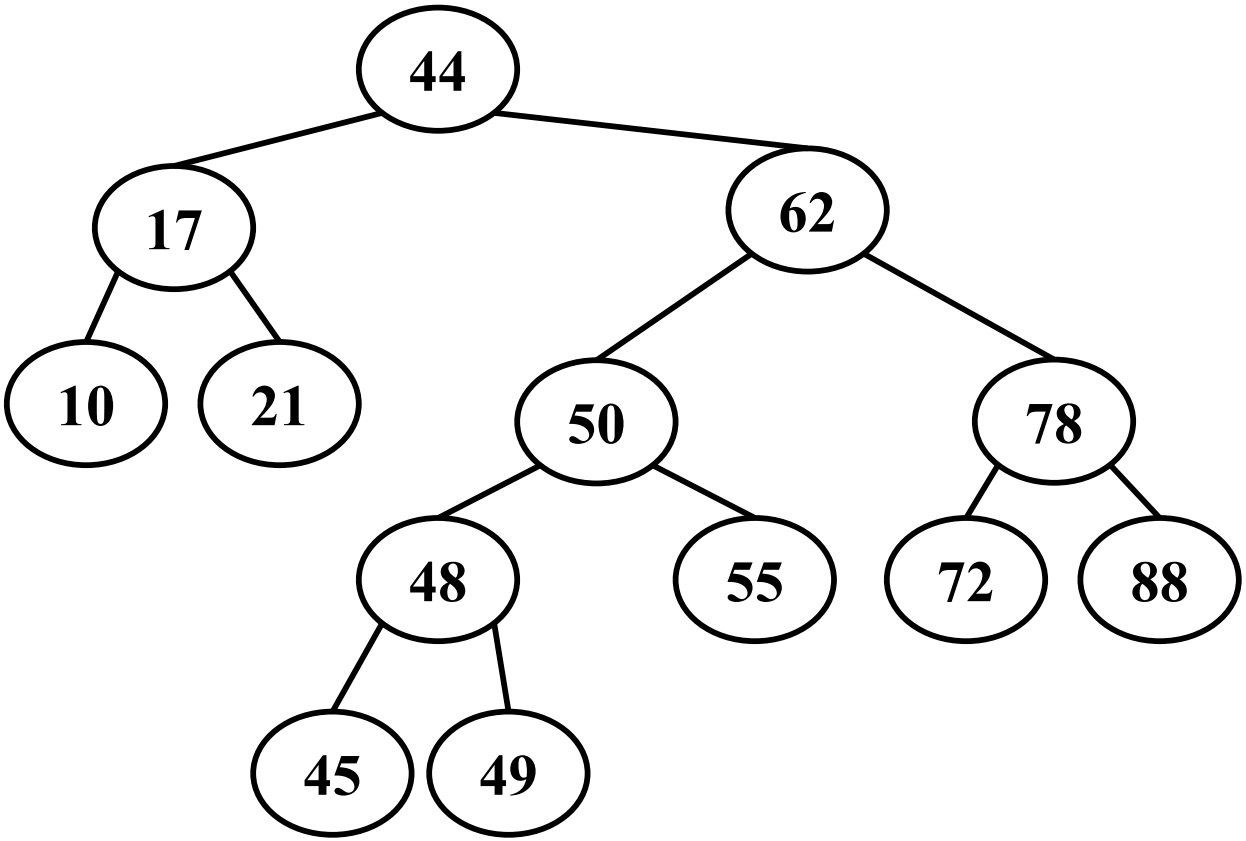
Delete 78 from the given AVL Tree



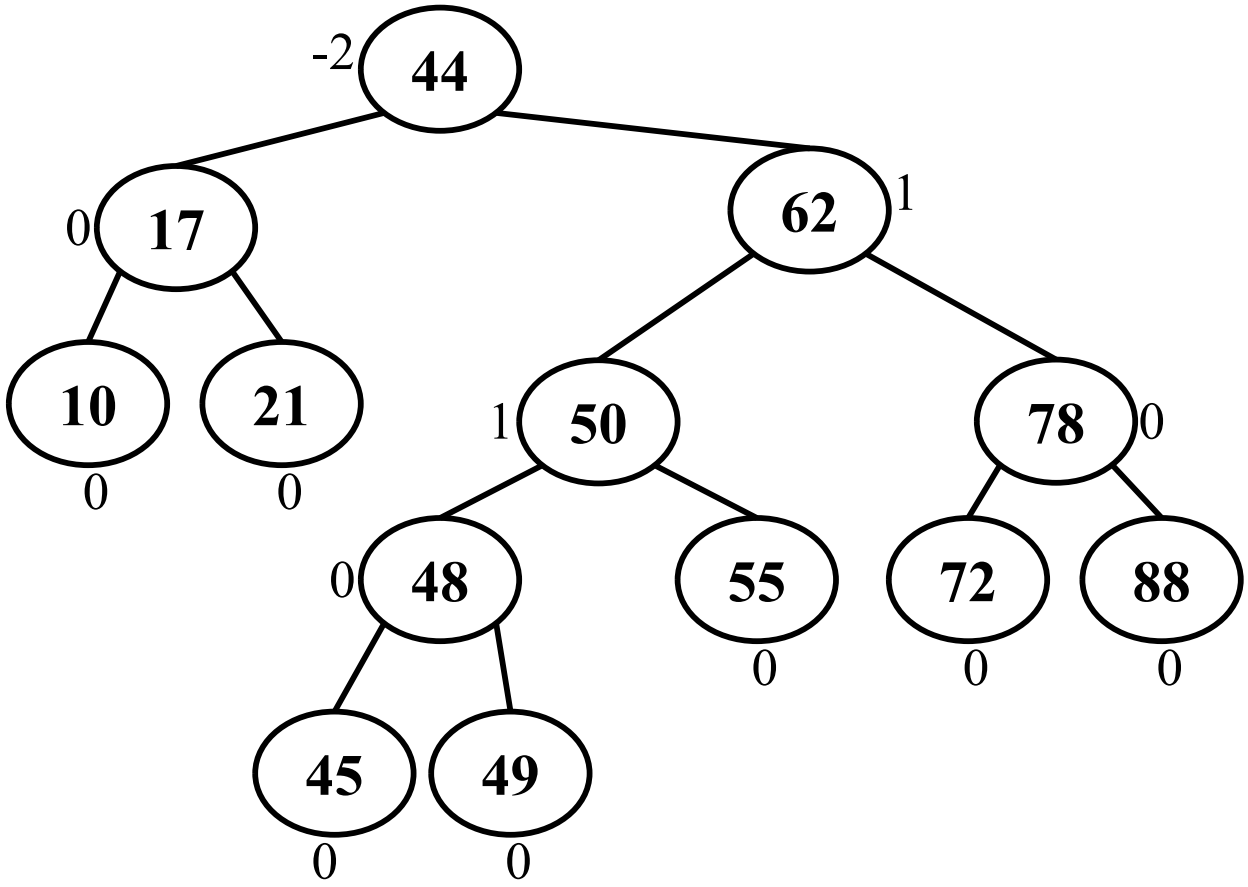
Delete 32 from the given AVL Tree



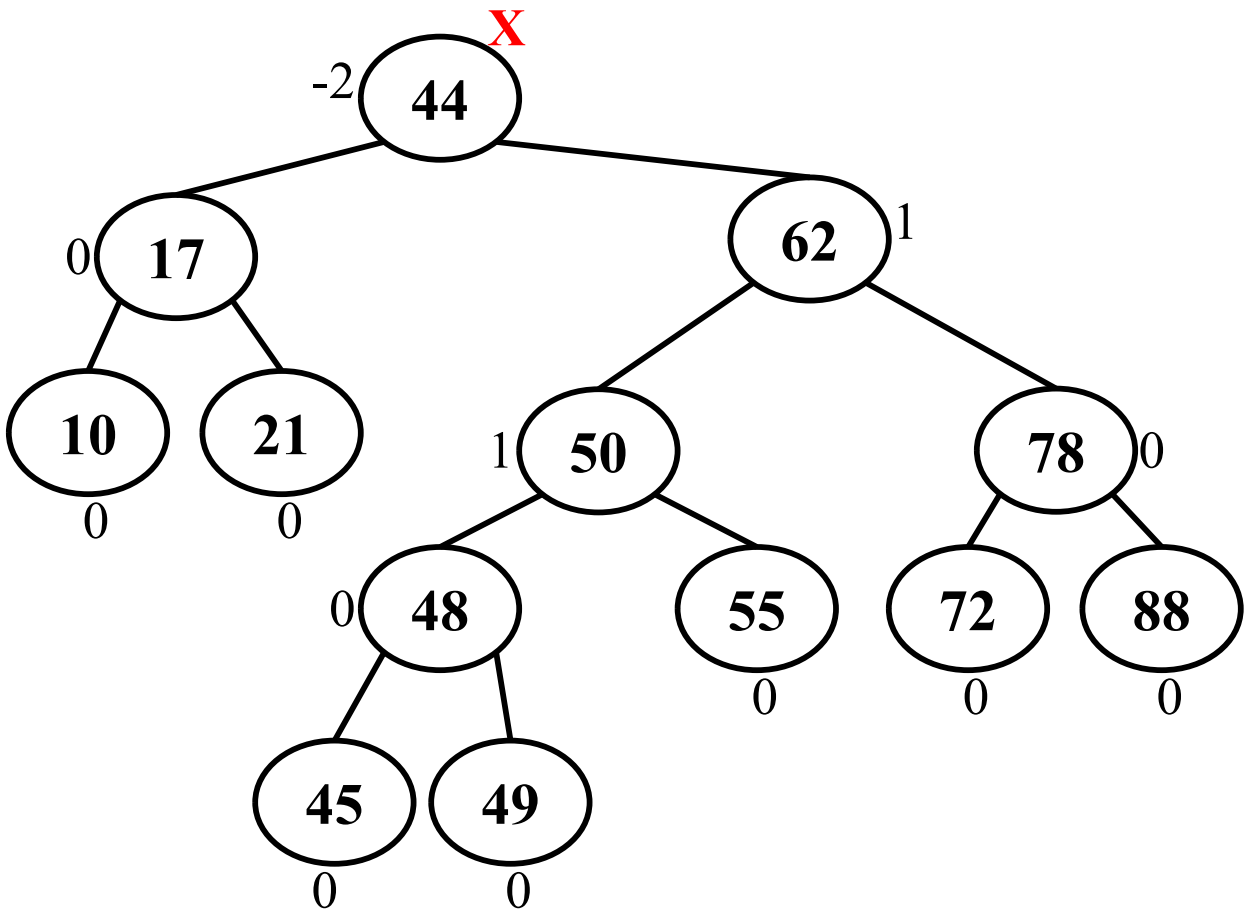
Delete 32 from the given AVL Tree



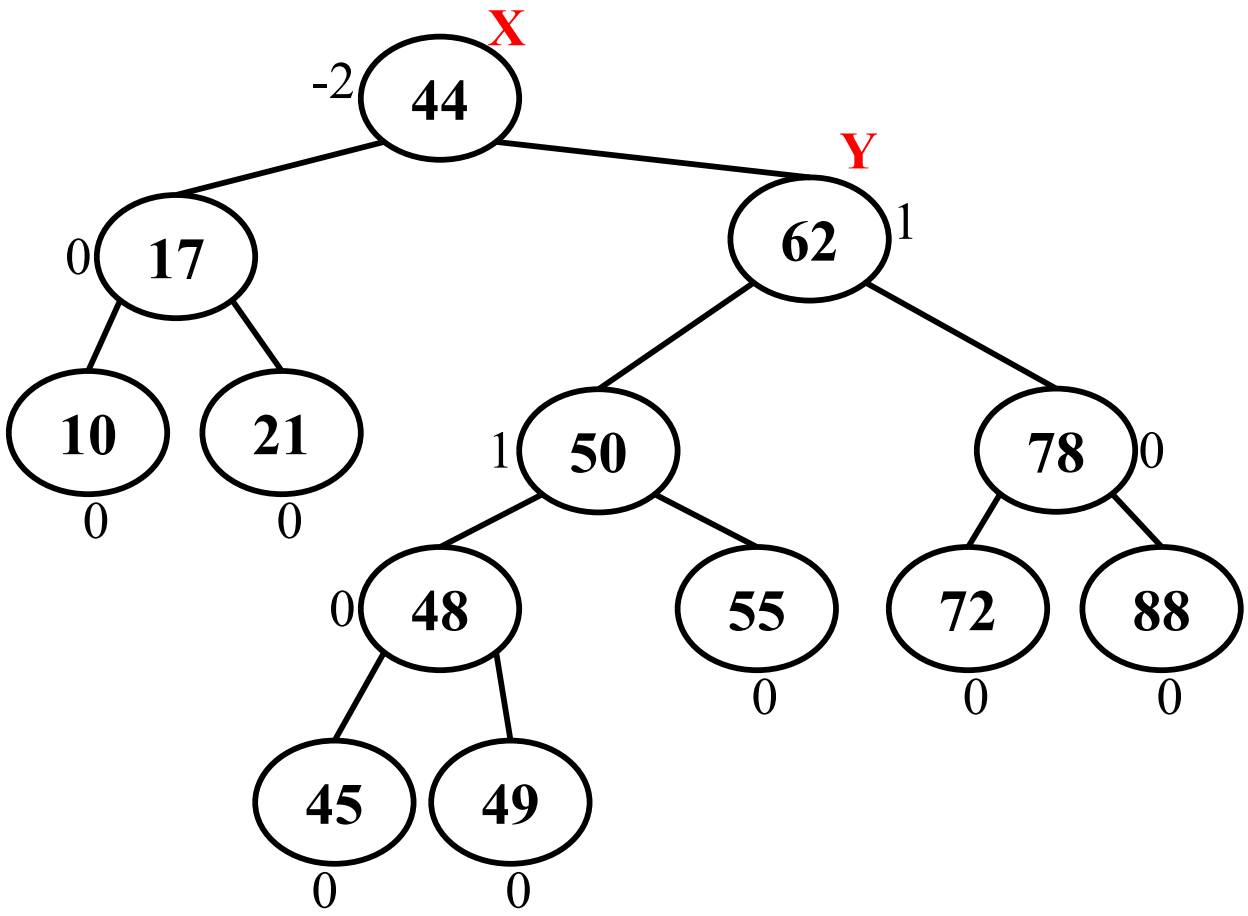
Delete 32 from the given AVL Tree



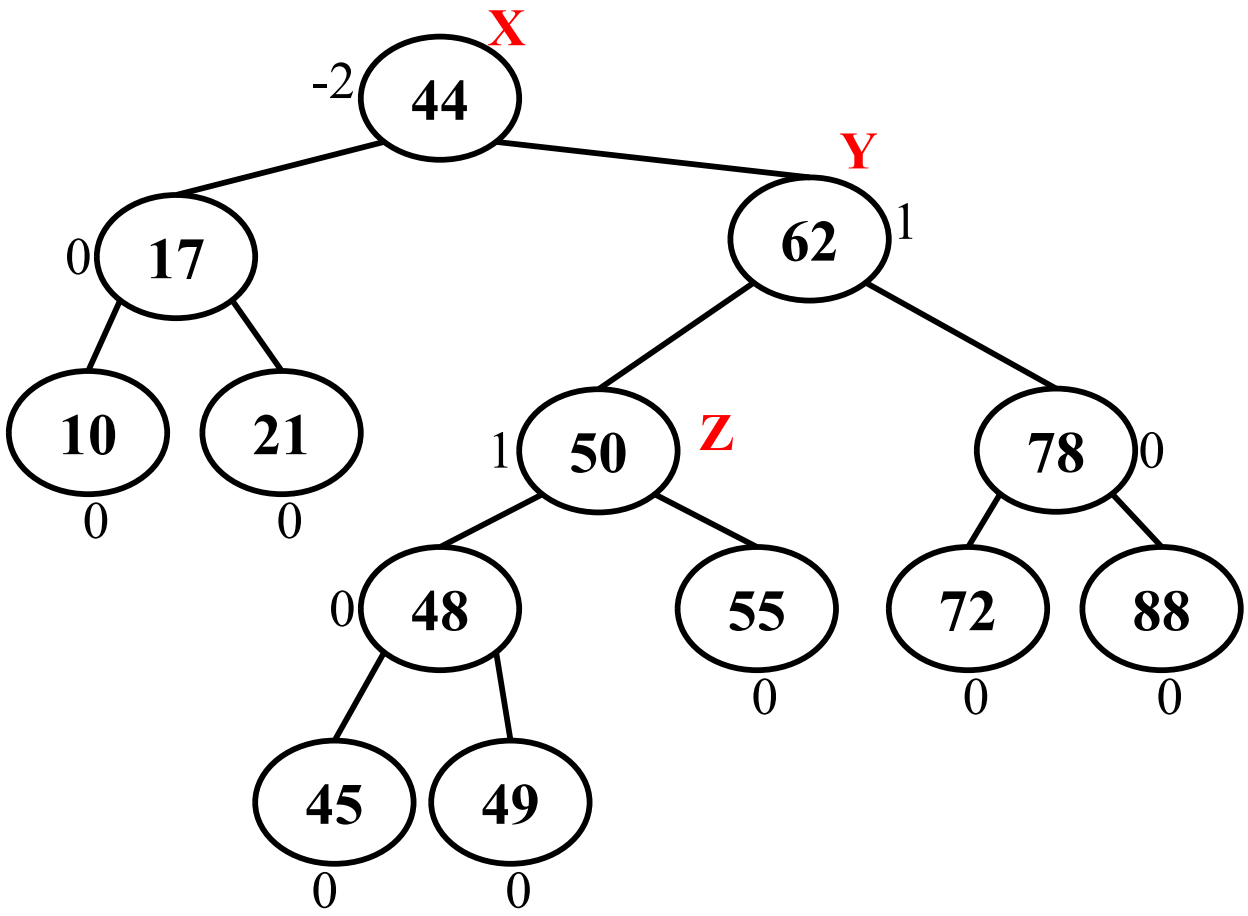
Delete 32 from the given AVL Tree



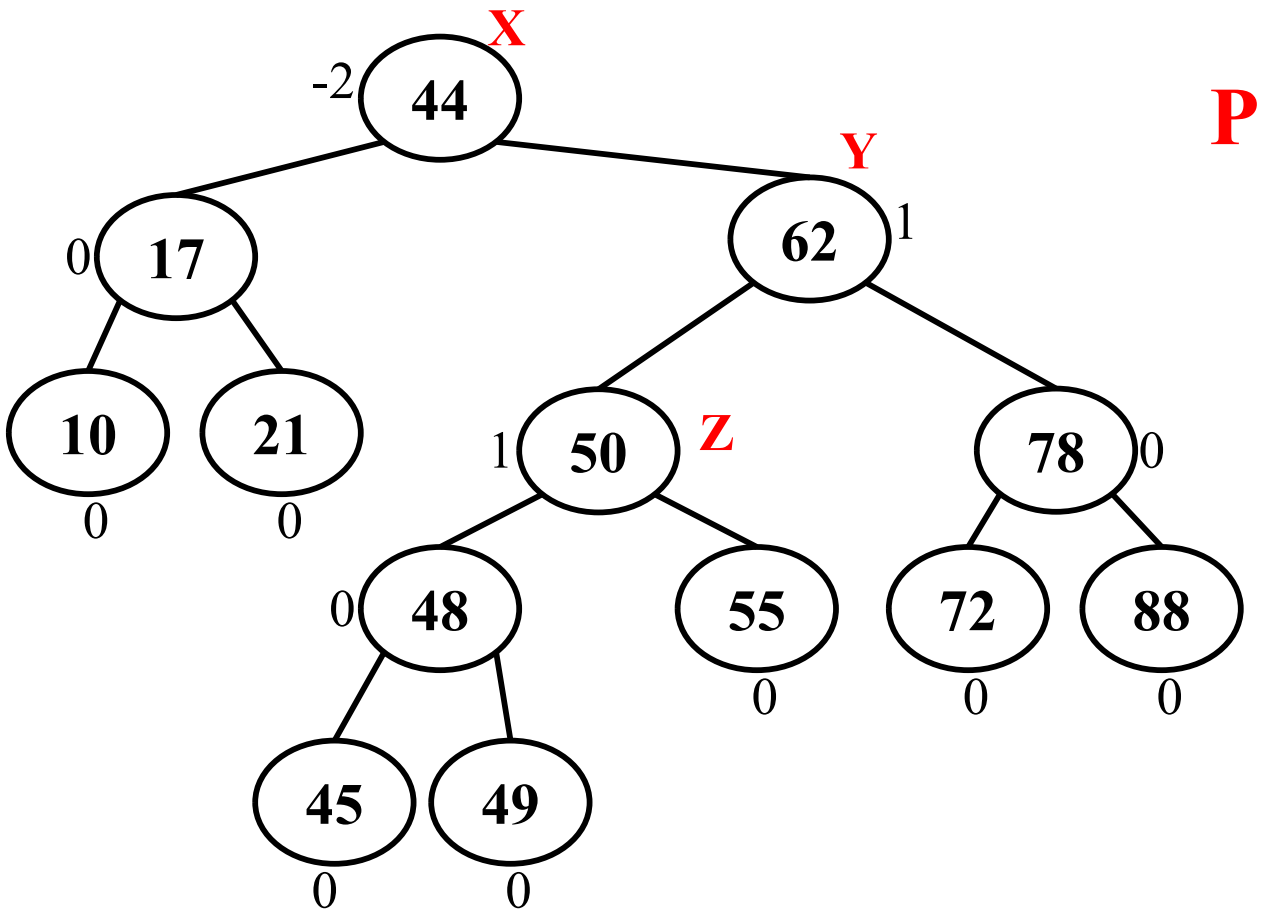
Delete 32 from the given AVL Tree



Delete 32 from the given AVL Tree

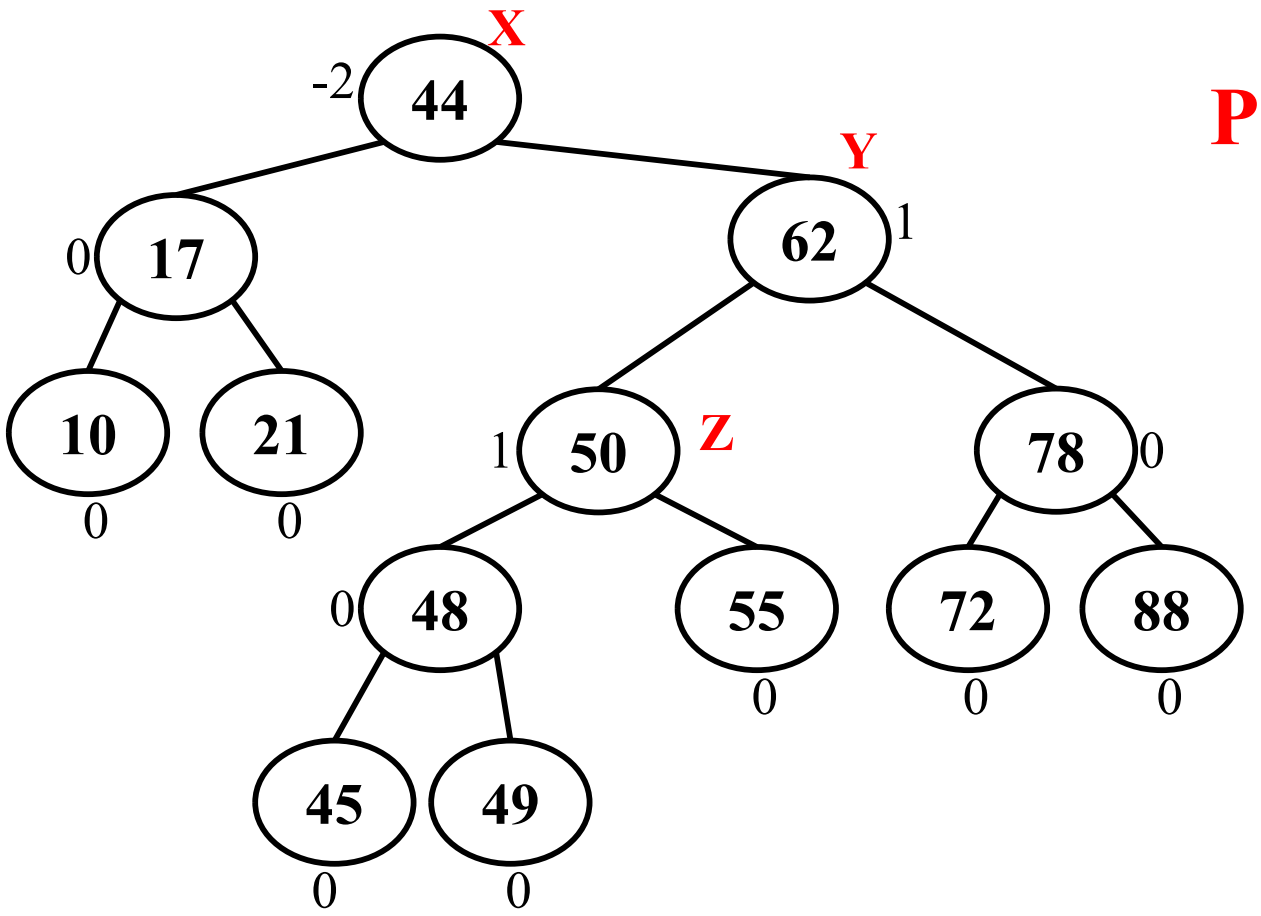


Delete 32 from the given AVL Tree



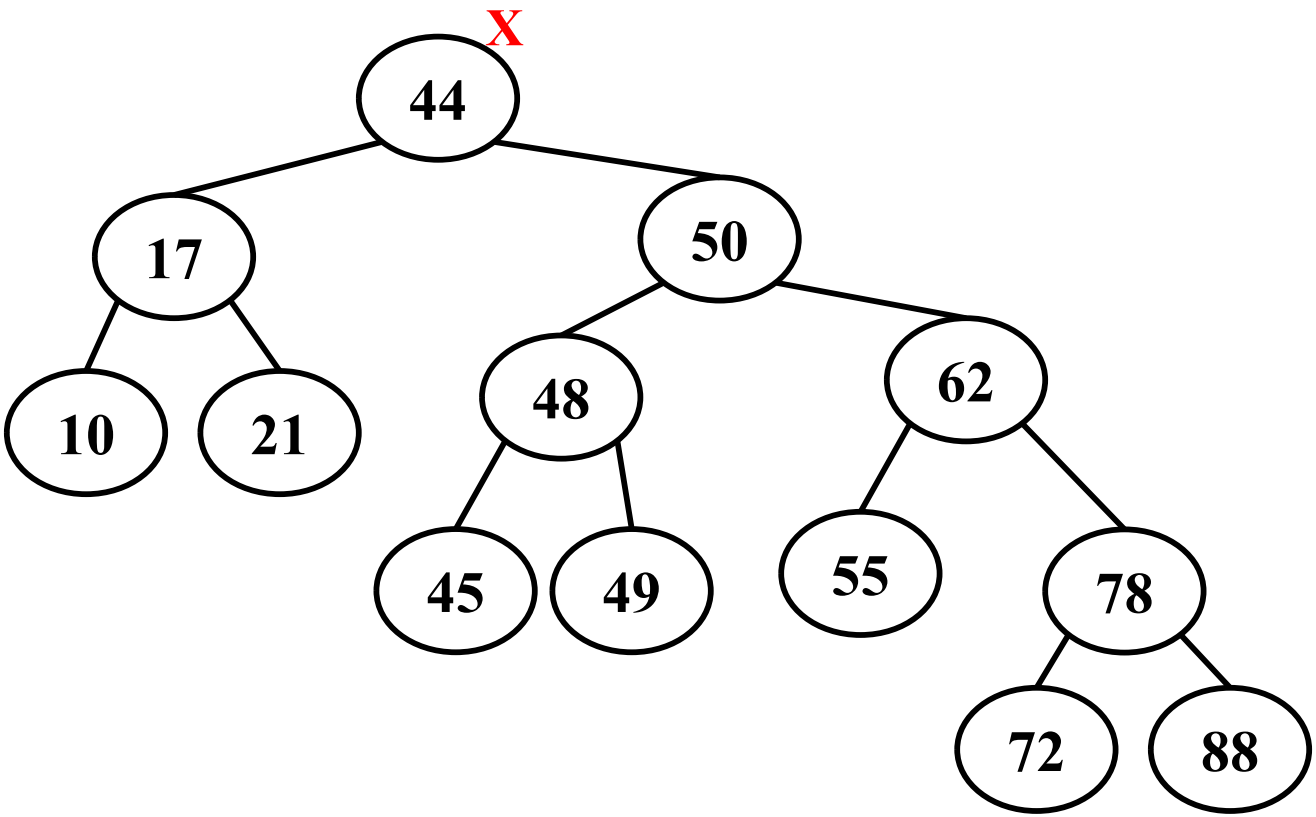
Perform RL(X)

Delete 32 from the given AVL Tree

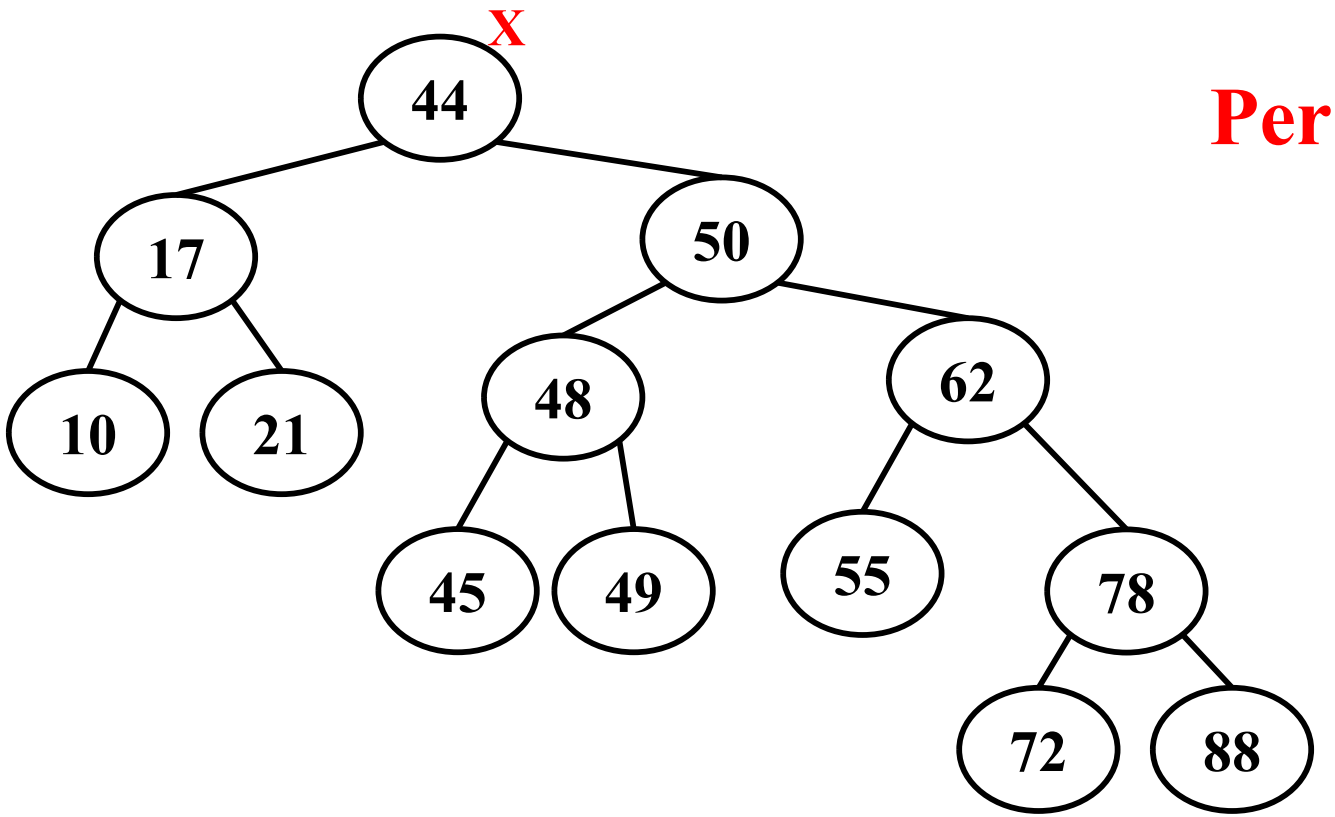


Perform RR(Y)

Delete 32 from the given AVL Tree



Delete 32 from the given AVL Tree



Perform LL(X)

Delete 32 from the given AVL Tree