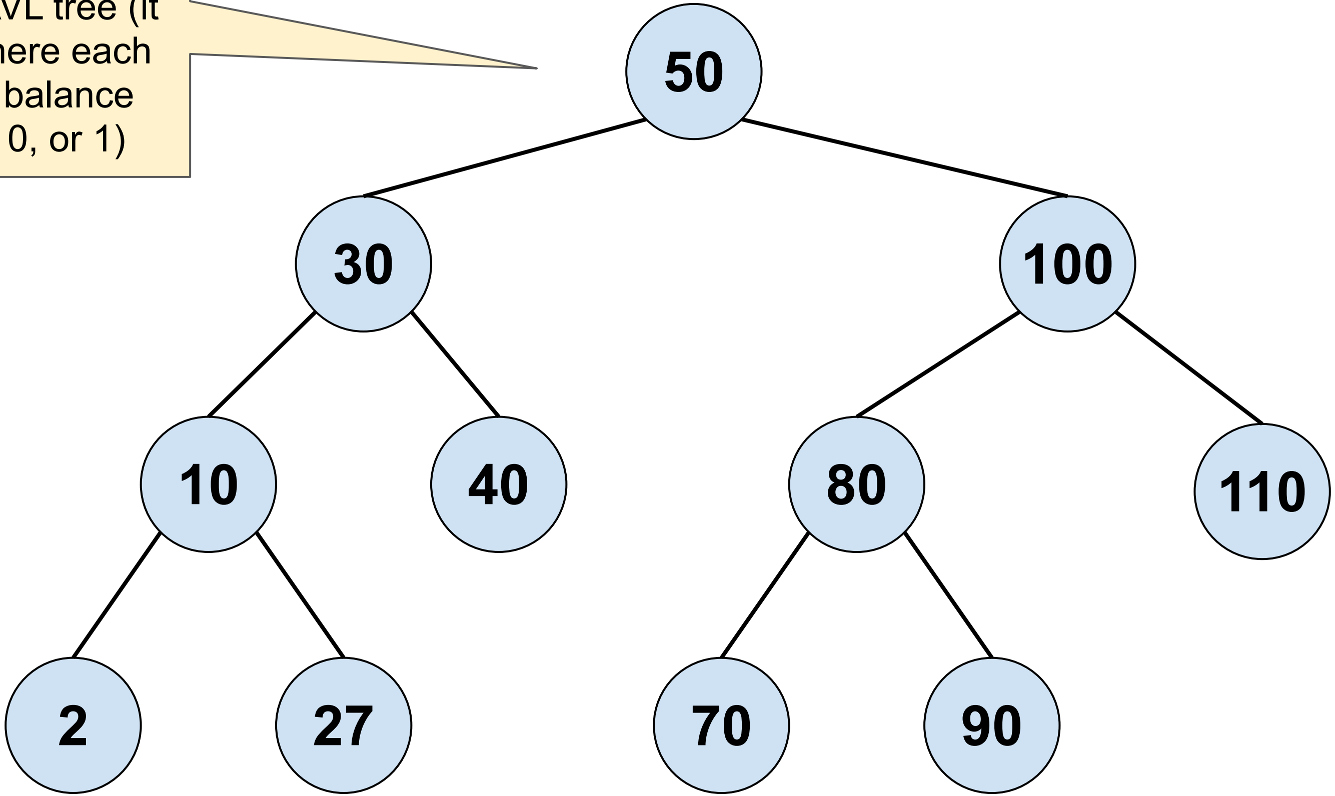
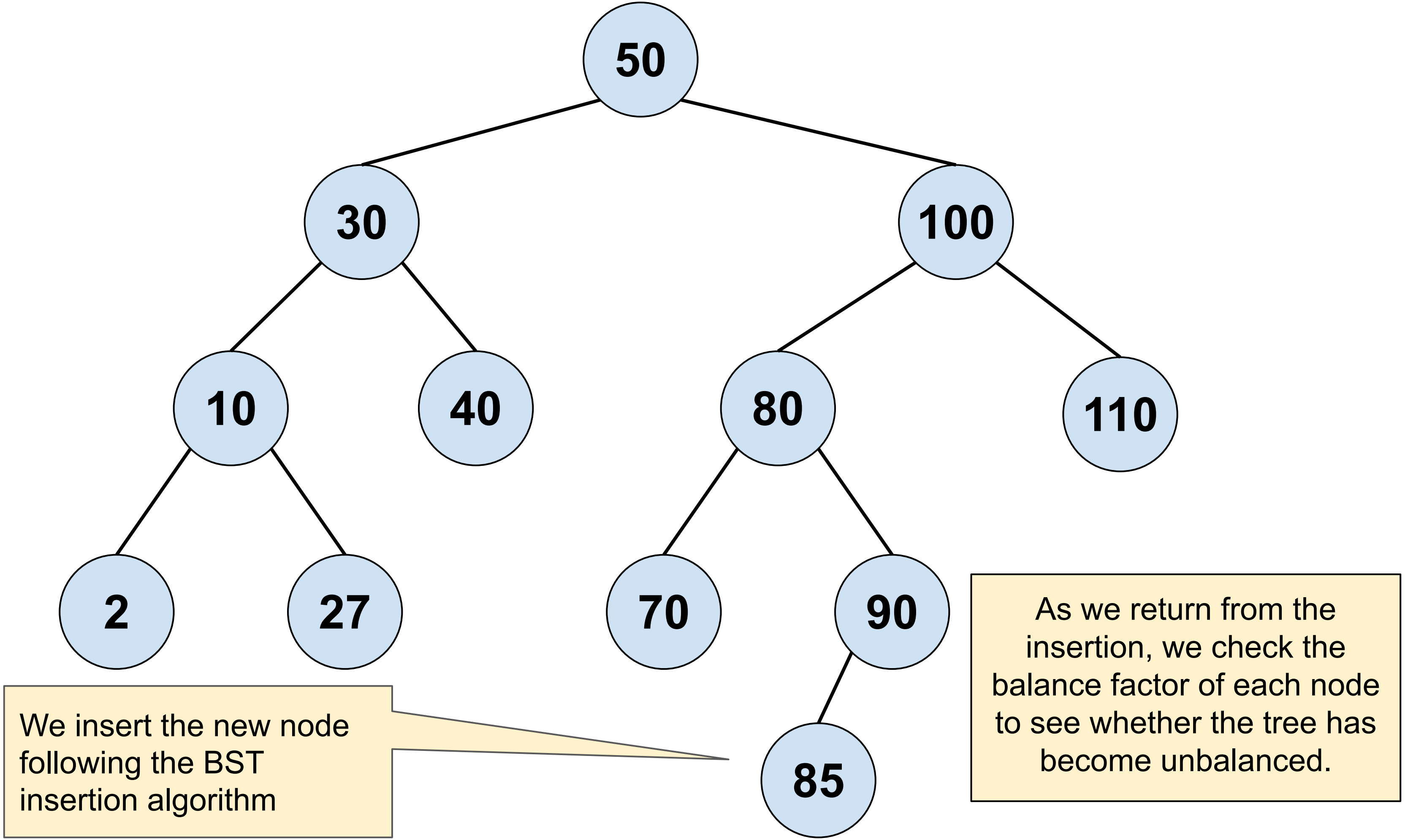
AVL Insertion Example

This is an AVL tree (it is a BST where each node has a balance factor of -1, 0, or 1)

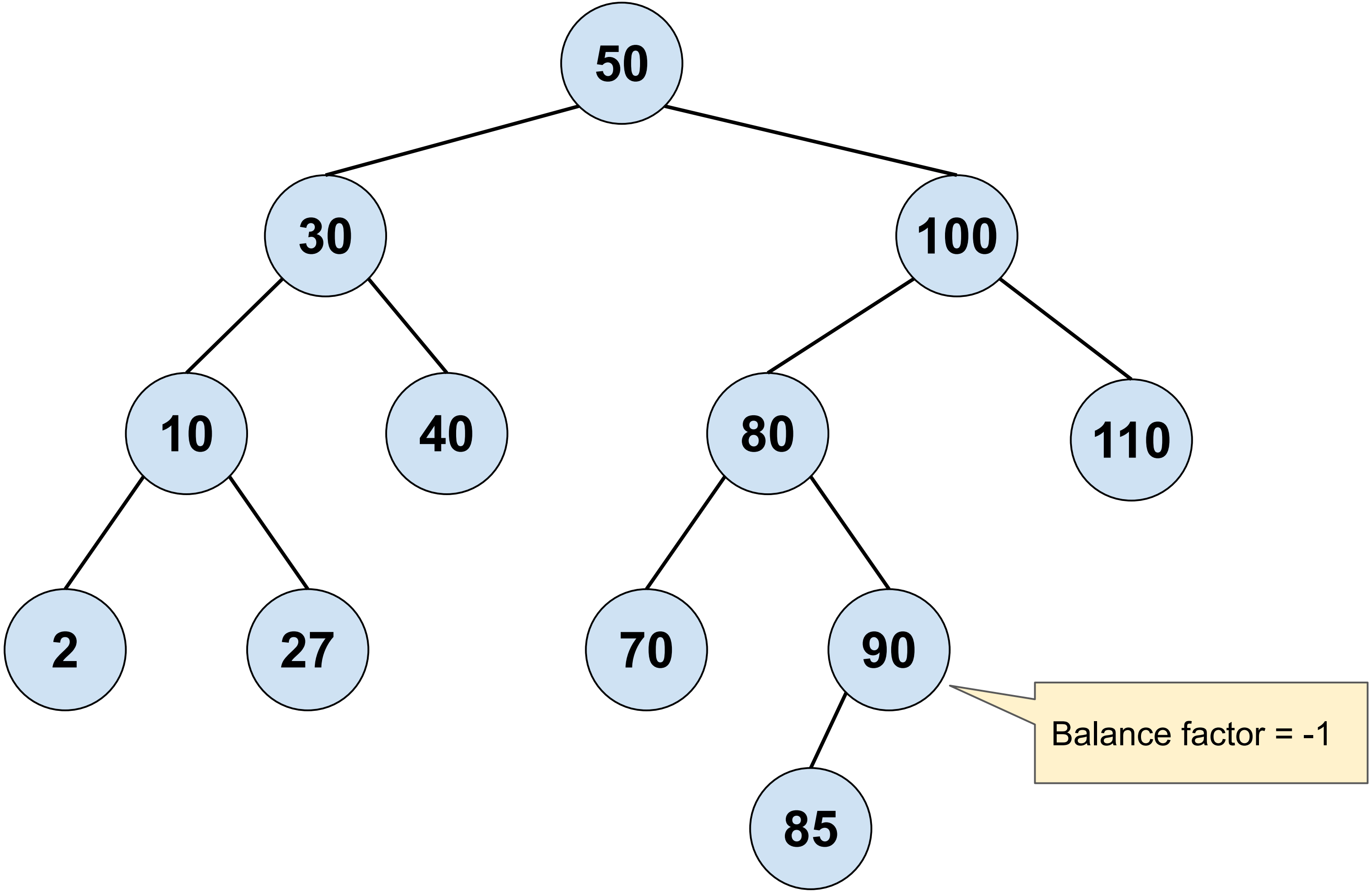


Let’s say we want to insert value **85**

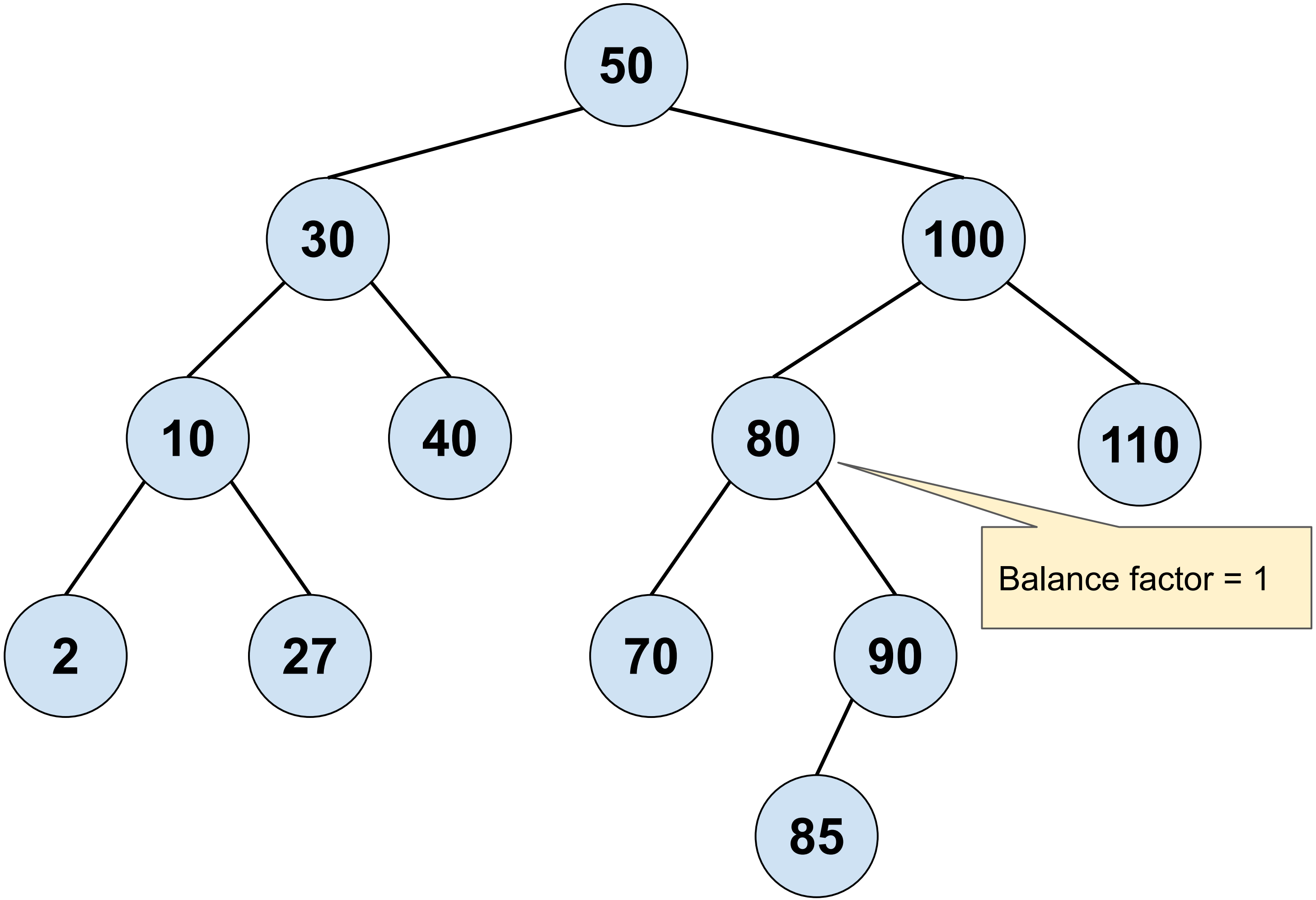


We insert the new node following the BST insertion algorithm

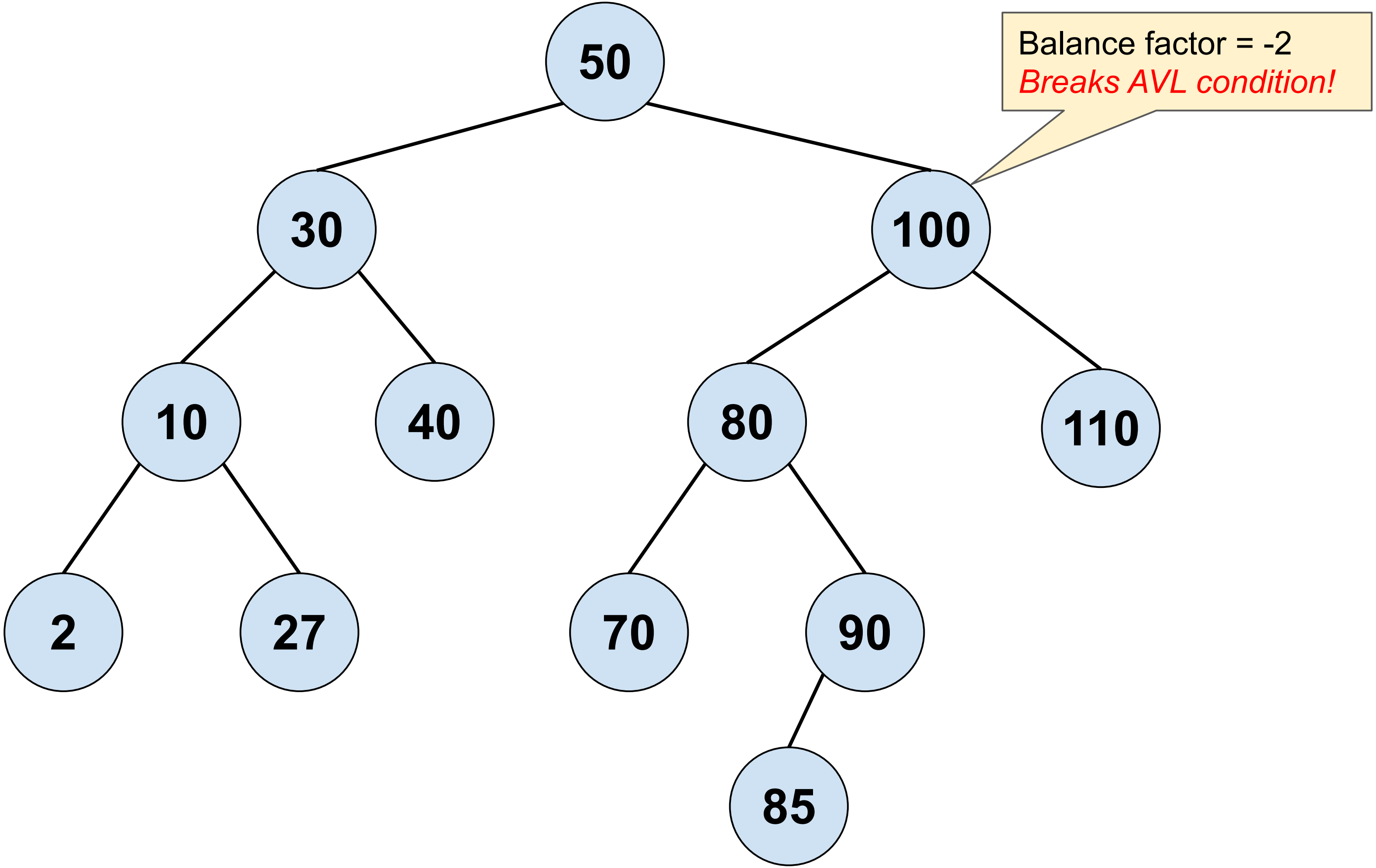
As we return from the insertion, we check the balance factor of each node to see whether the tree has become unbalanced.



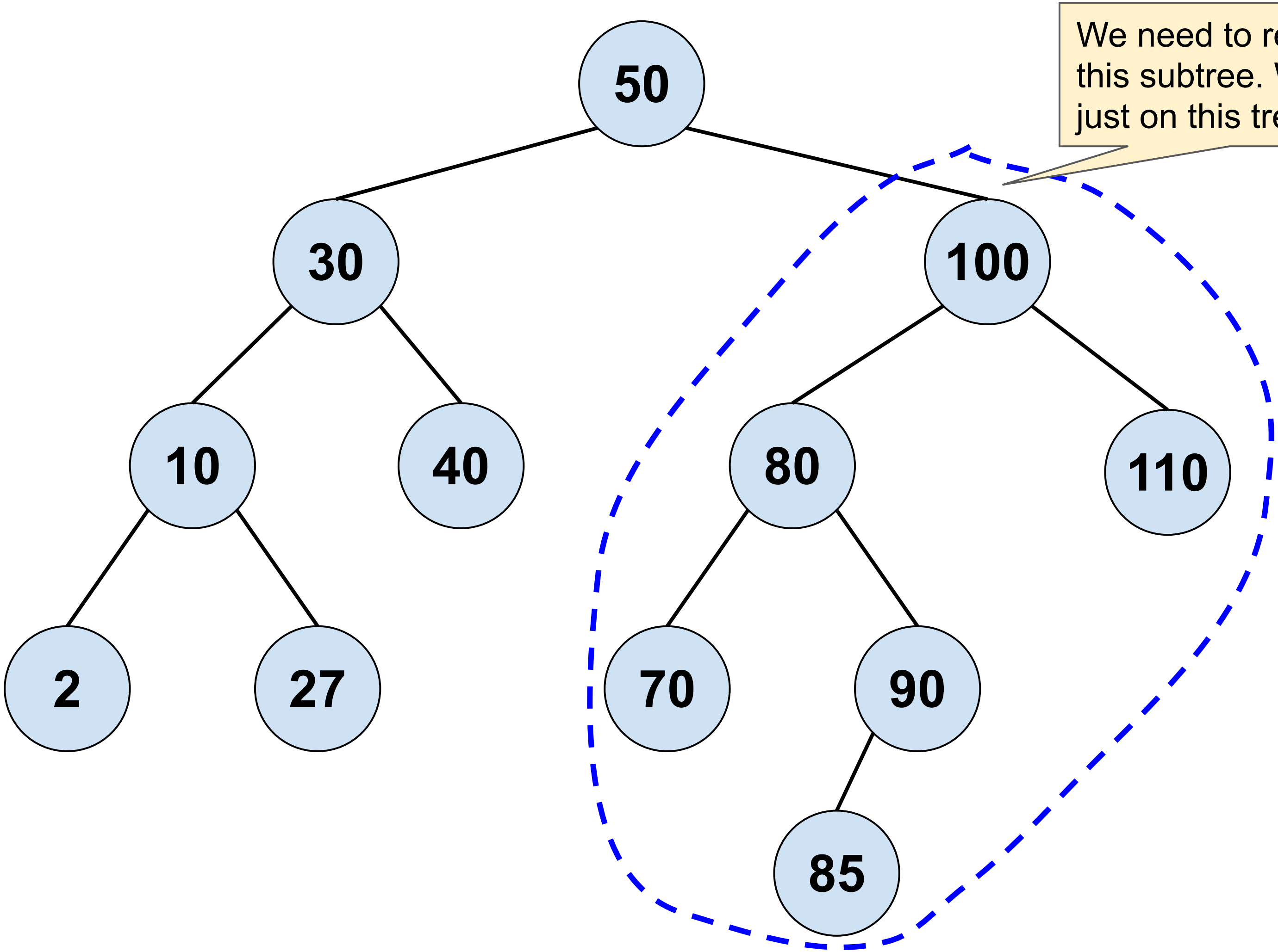
Balance factor = -1



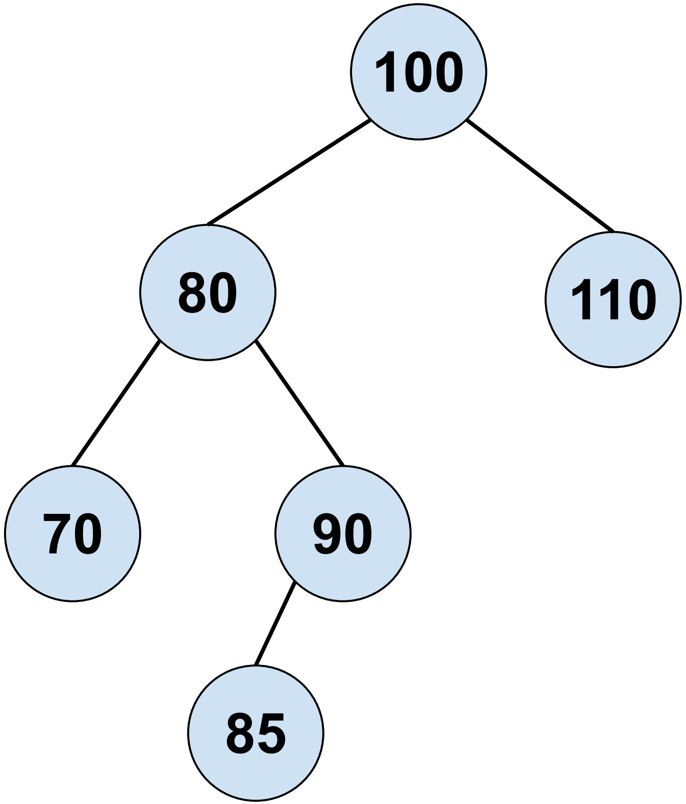
Balance factor = 1



Balance factor = -2
Breaks AVL condition!

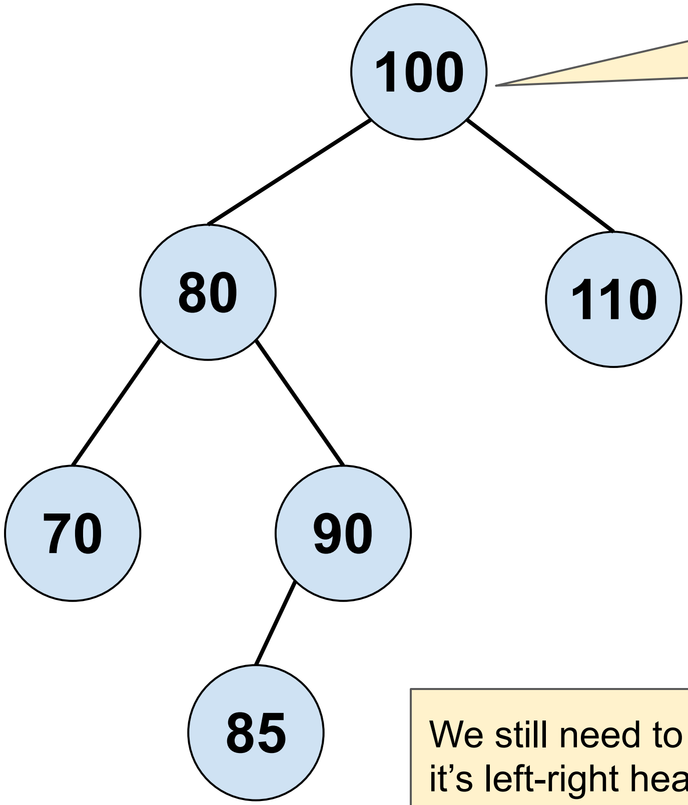


We need to rebalance this subtree. We'll focus just on this tree



We need to check what case we're in, based on the heaviness of the left and right subtrees:

- Left-left heavy ("Left heavy")
- Right-right heavy ("Right heavy")
- Left-Right heavy
- Right-Left heavy

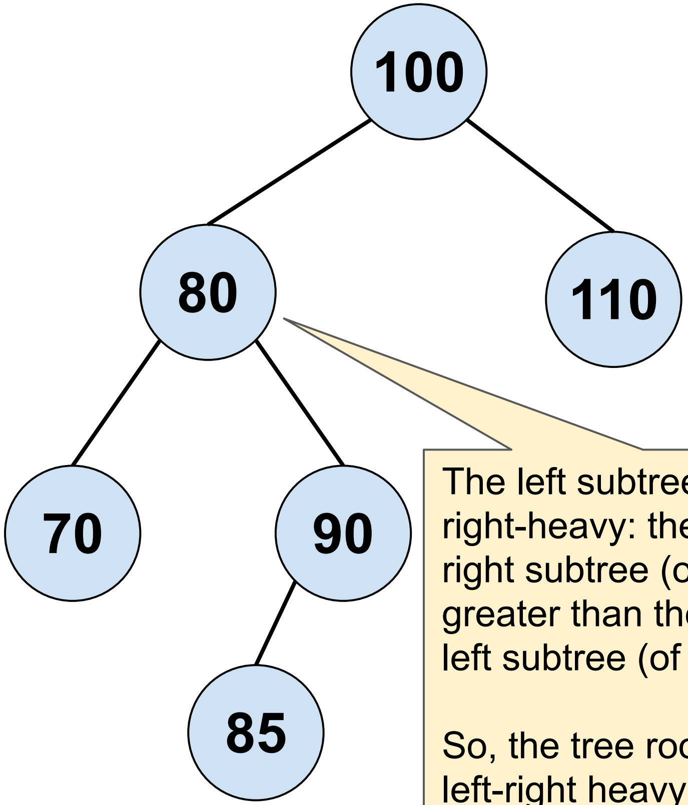


The tree is left heavy: the height of the left subtree is greater than the right subtree.

We need to check what case we're in, based on the heaviness of the left and right subtrees:

- Left-left heavy ("Left heavy")
- ~~Right-right heavy ("Right heavy")~~
- Left-Right heavy
- ~~Right-Left heavy~~

We still need to check whether it's left-right heavy, based on the heaviness of the left subtree.

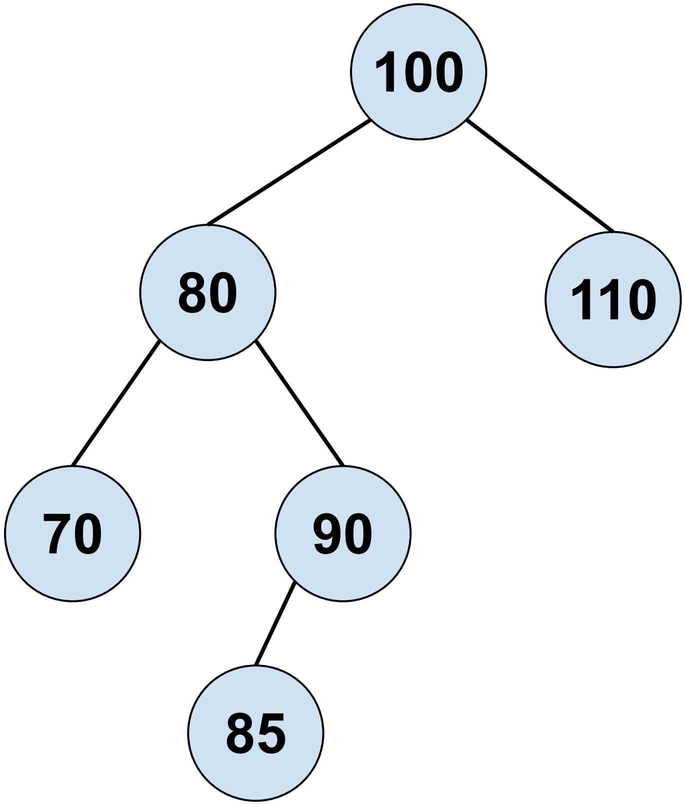


The left subtree (of 100) is right-heavy: the height of the right subtree (of 80) is greater than the height of the left subtree (of 80).

So, the tree rooted at 100 is left-right heavy (it is left-heavy, and its left subtree is right-heavy)

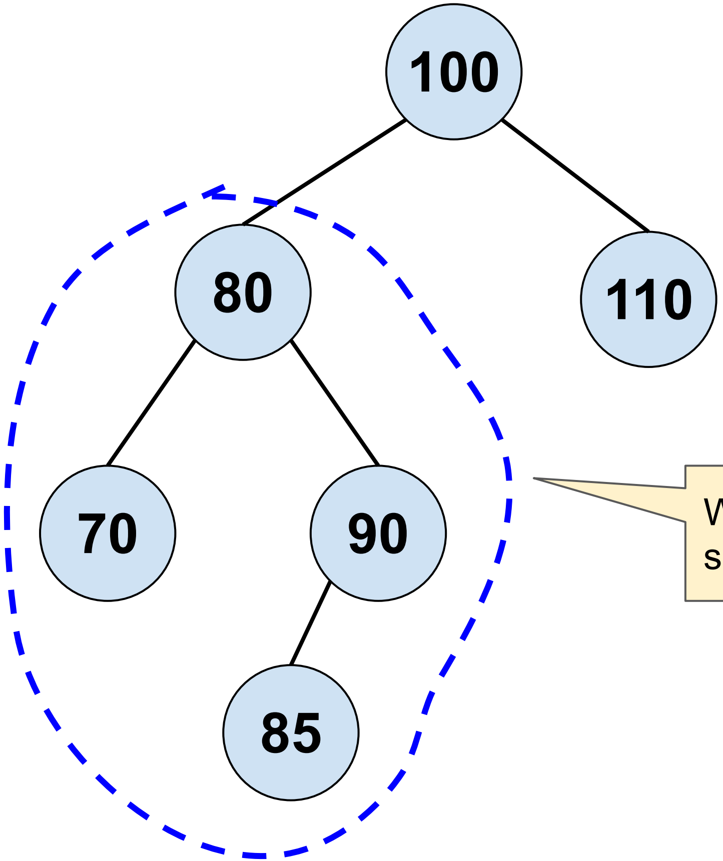
We need to check what case we're in, based on the heaviness of the left and right subtrees:

- ~~Left-left heavy ("Left heavy")~~
- ~~Right-right heavy ("Right heavy")~~
- Left-Right heavy
- ~~Right-Left heavy~~

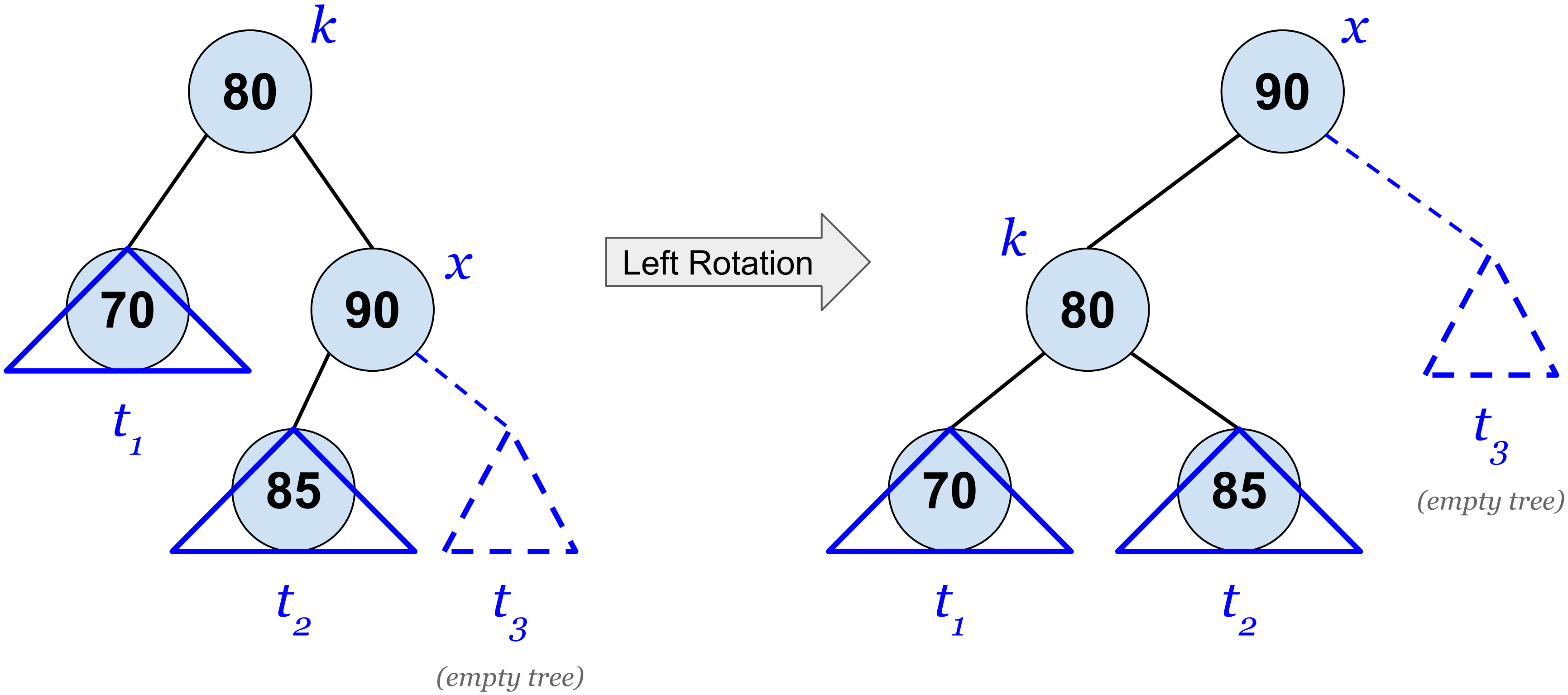


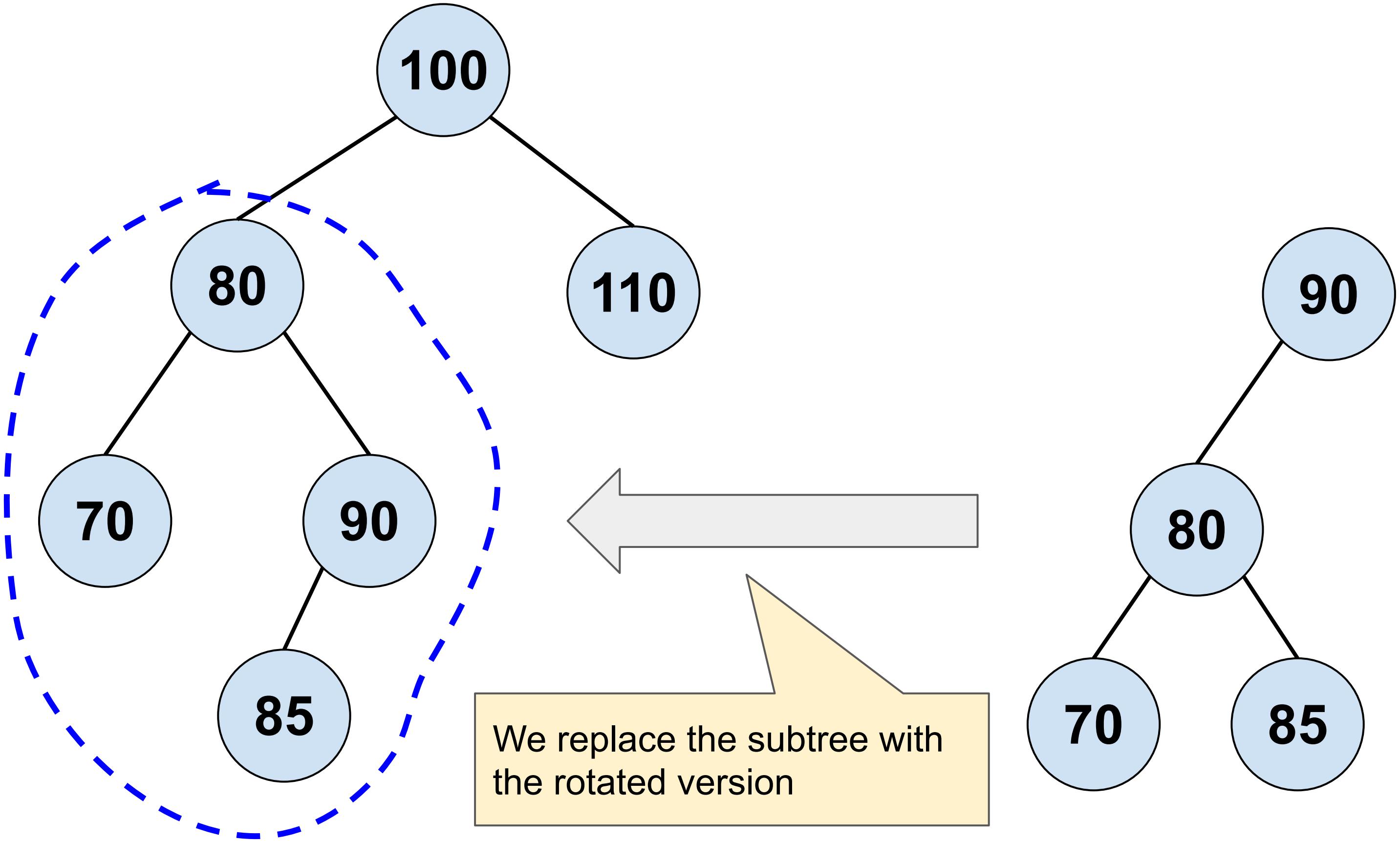
To re-balance a left-right heavy tree, we need to:

1. Left-rotate its left subtree
2. Right-rotate the entire tree.

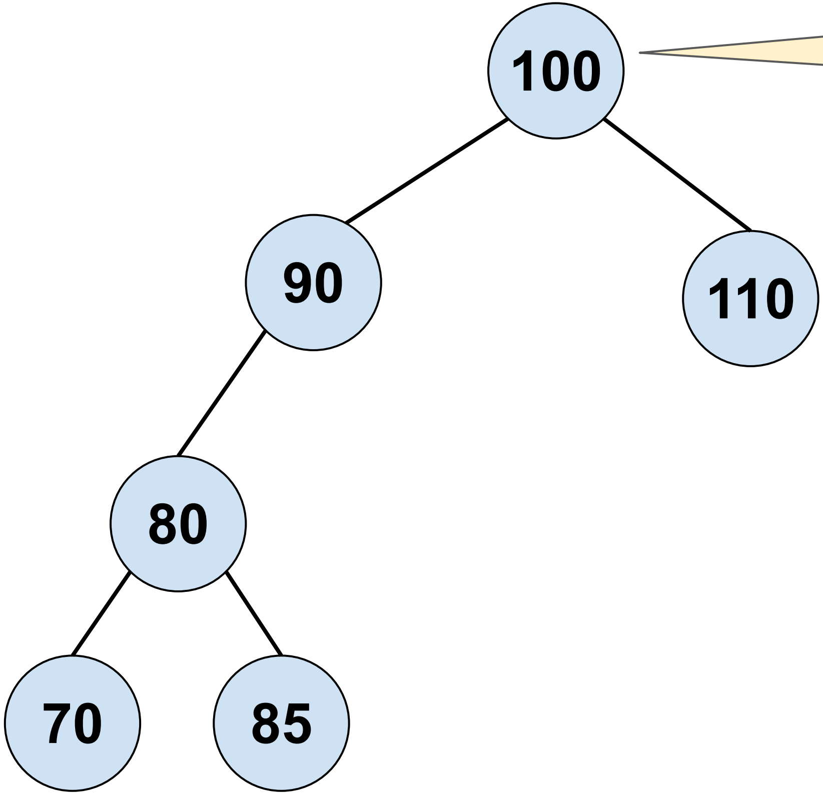


We start by left-rotating this subtree

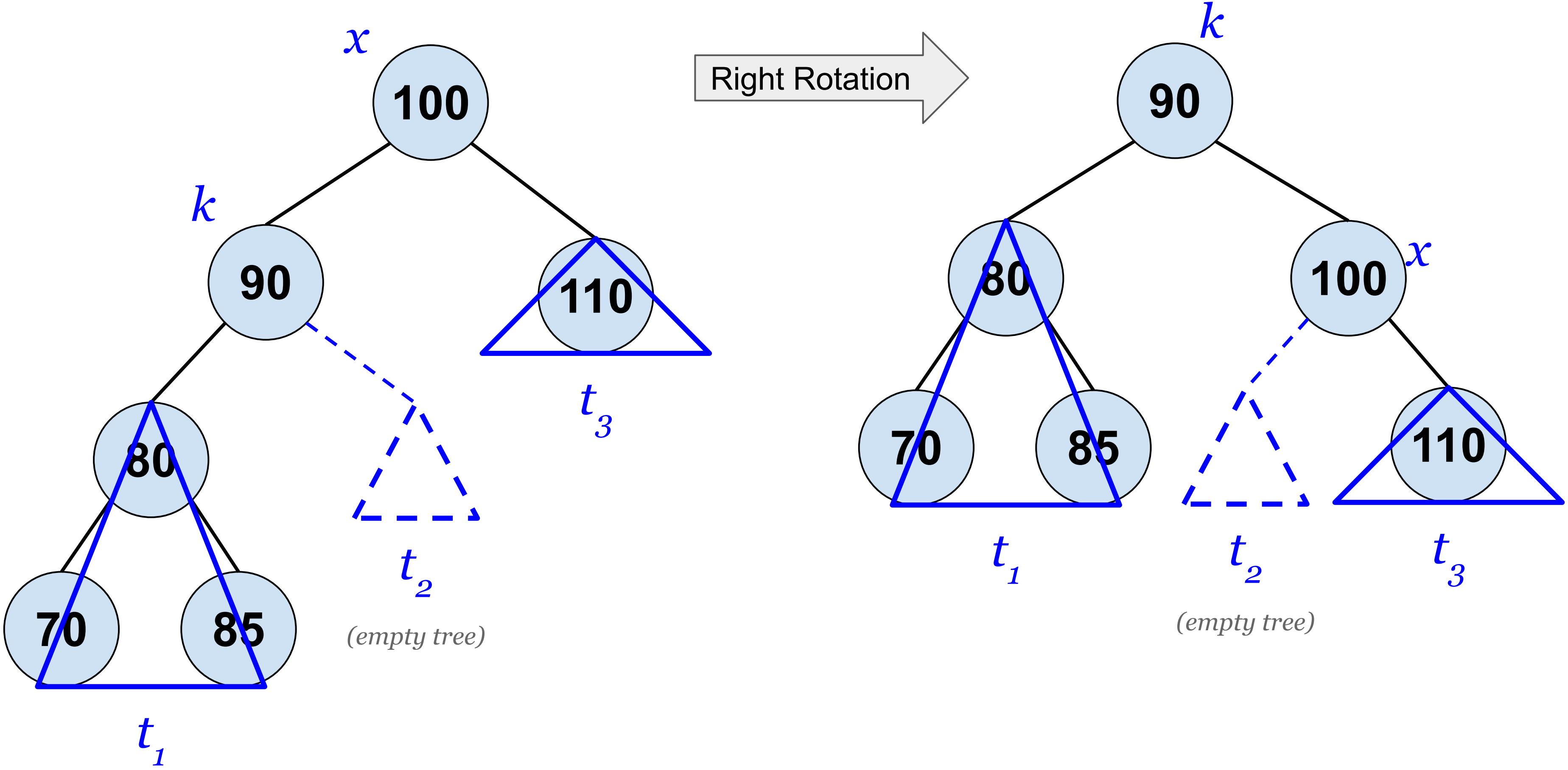

k
80
70
90
x
85
t_1
t_2
t_3
(empty tree)
Left Rotation
x
90
k
80
70
85
t_3
(empty tree)
t_1
t_2

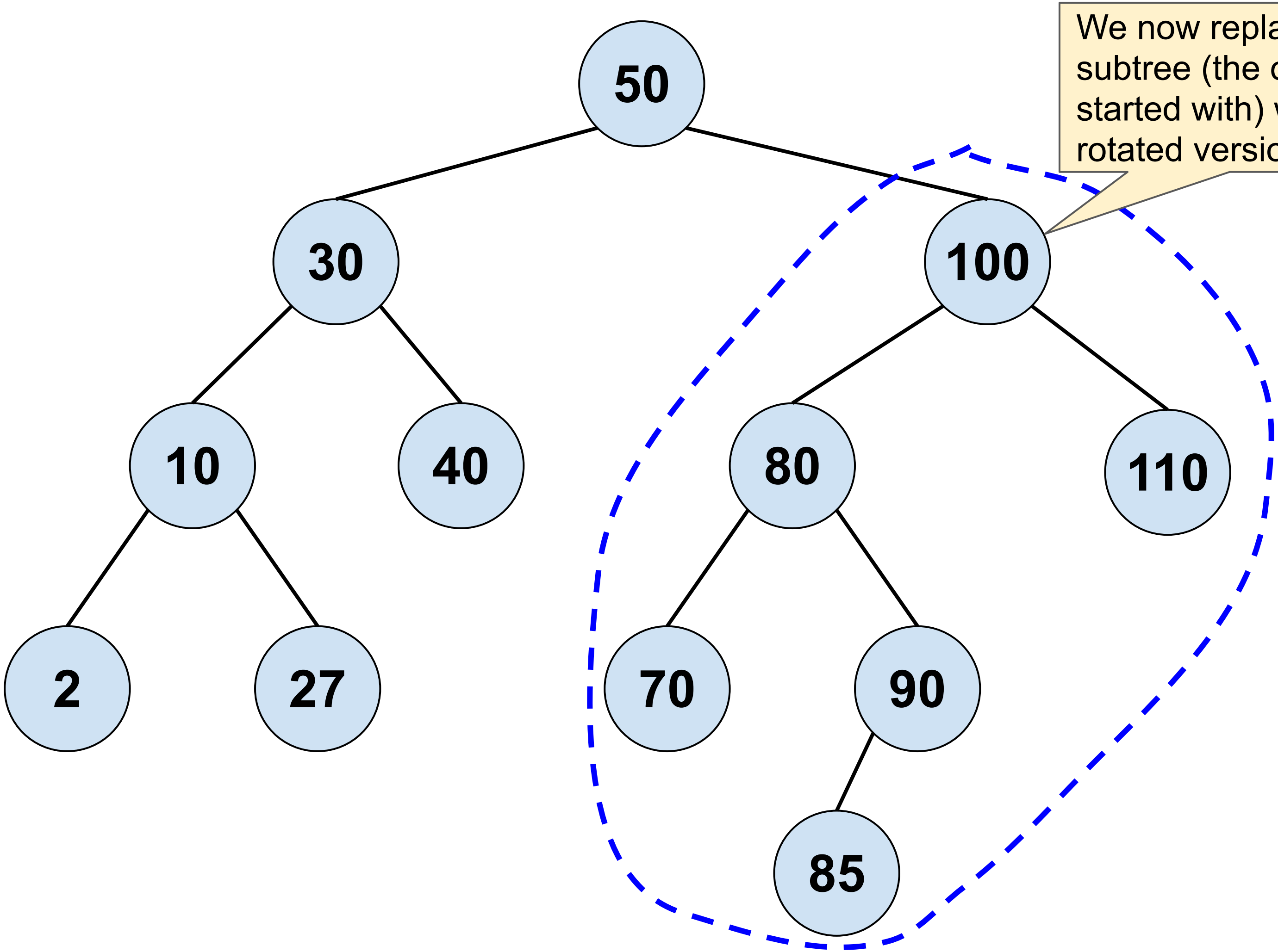


We replace the subtree with the rotated version

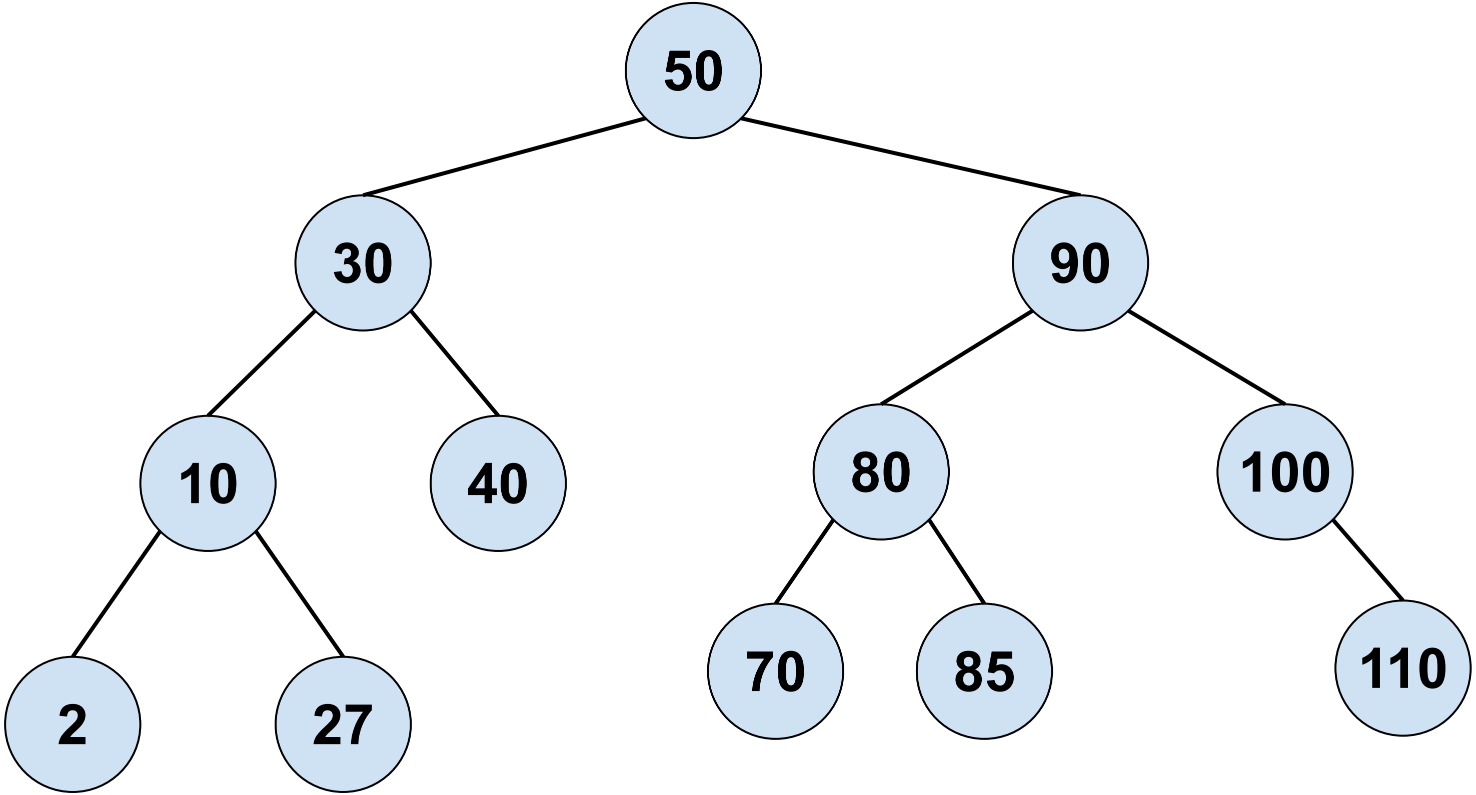


The next step is to right-rotate this entire tree. Notice how it's now left-left heavy: the tree is left-heavy, and its left subtree is itself left-heavy


x
100
k
90
110
80
70
85
t_1
t_2
(empty tree)
t_3
Right Rotation
k
90
80
100
x
70
85
110
t_1
t_2
t_3
(empty tree)



We now replace this subtree (the one we started with) with the rotated version



The tree now meets the AVL condition again
(all nodes have a balance factor of -1, 0, or 1)