CSCI 241

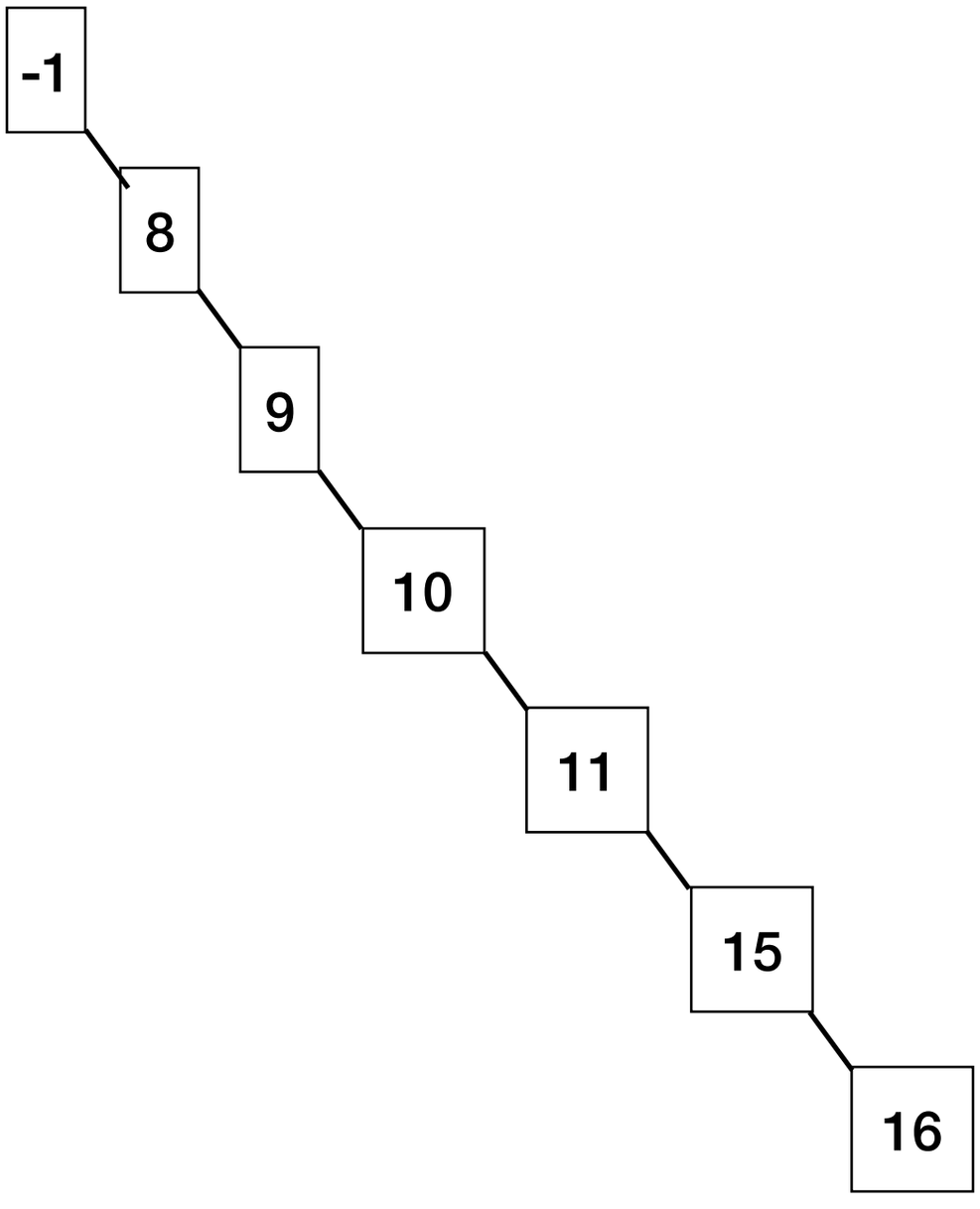
Lecture 13c:
Tree Rotations

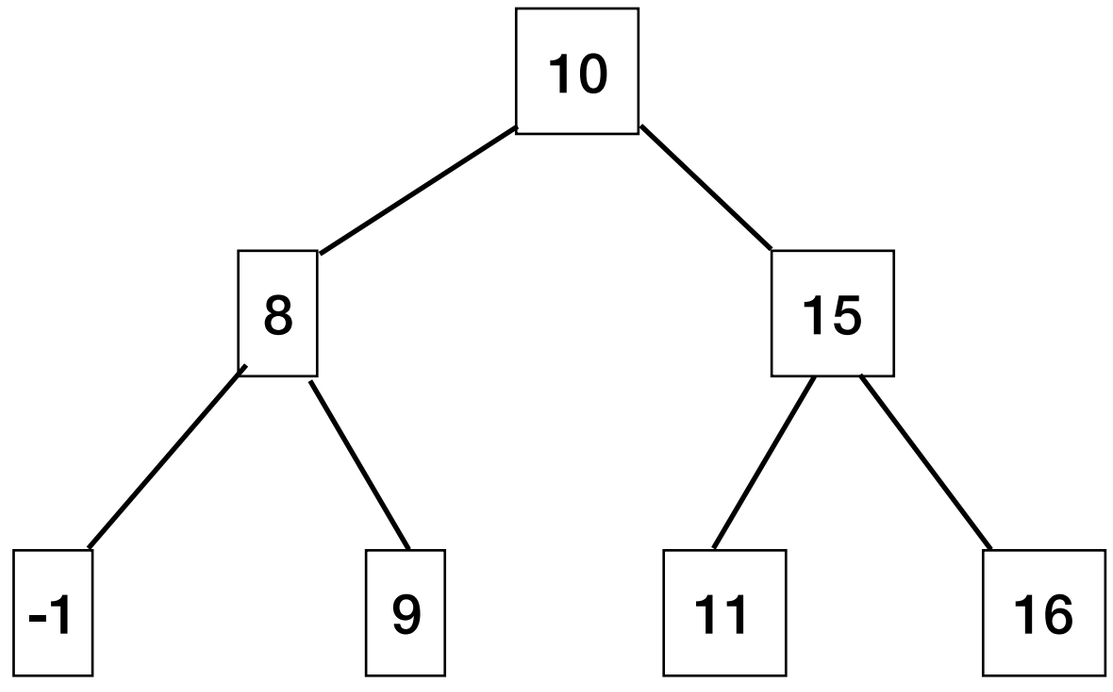
Goals

- Be prepared implement rotations in BSTs

Tree Badness: Balance Factor

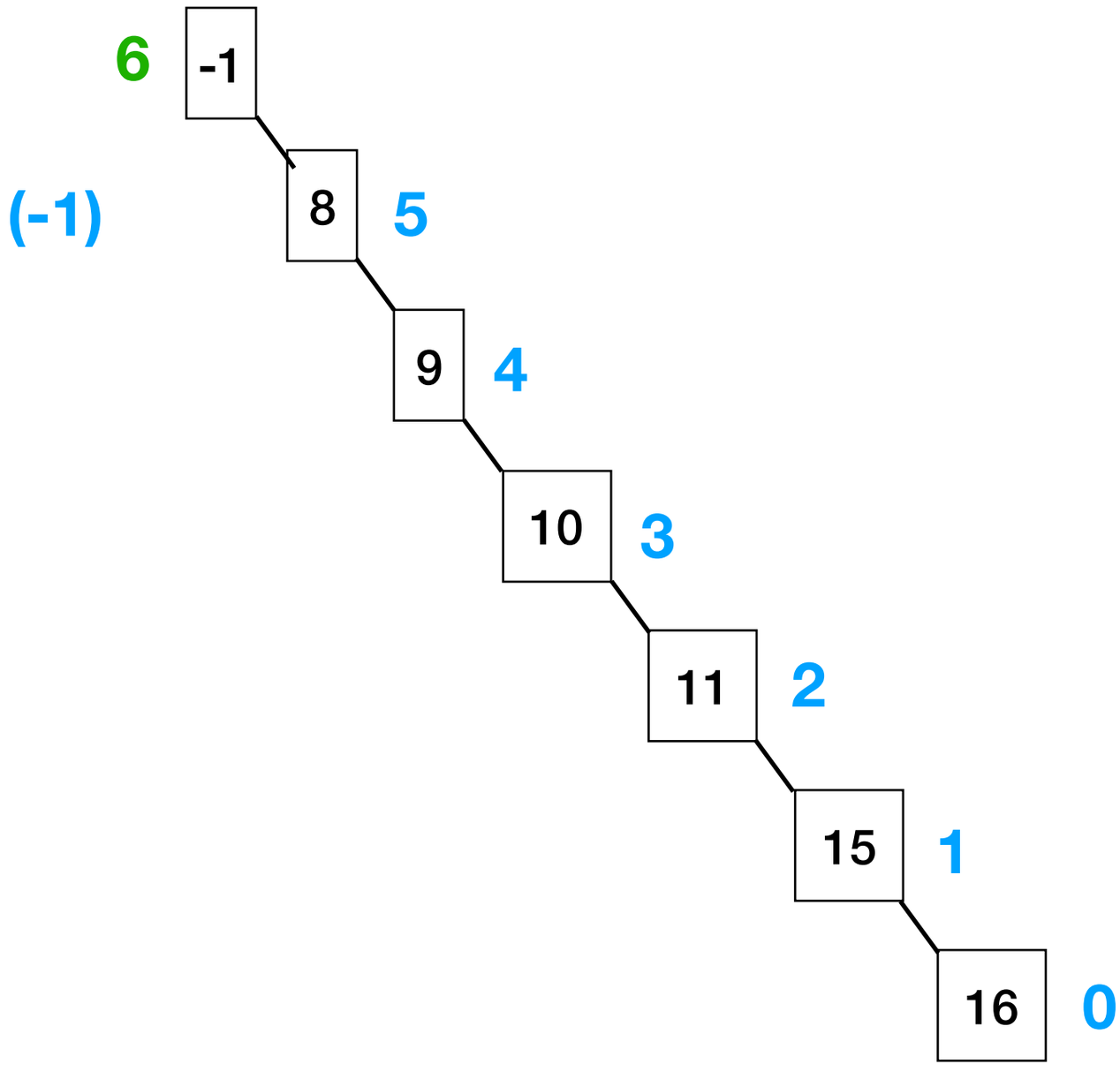
Balance(n): height(n.right) - height(n.left)

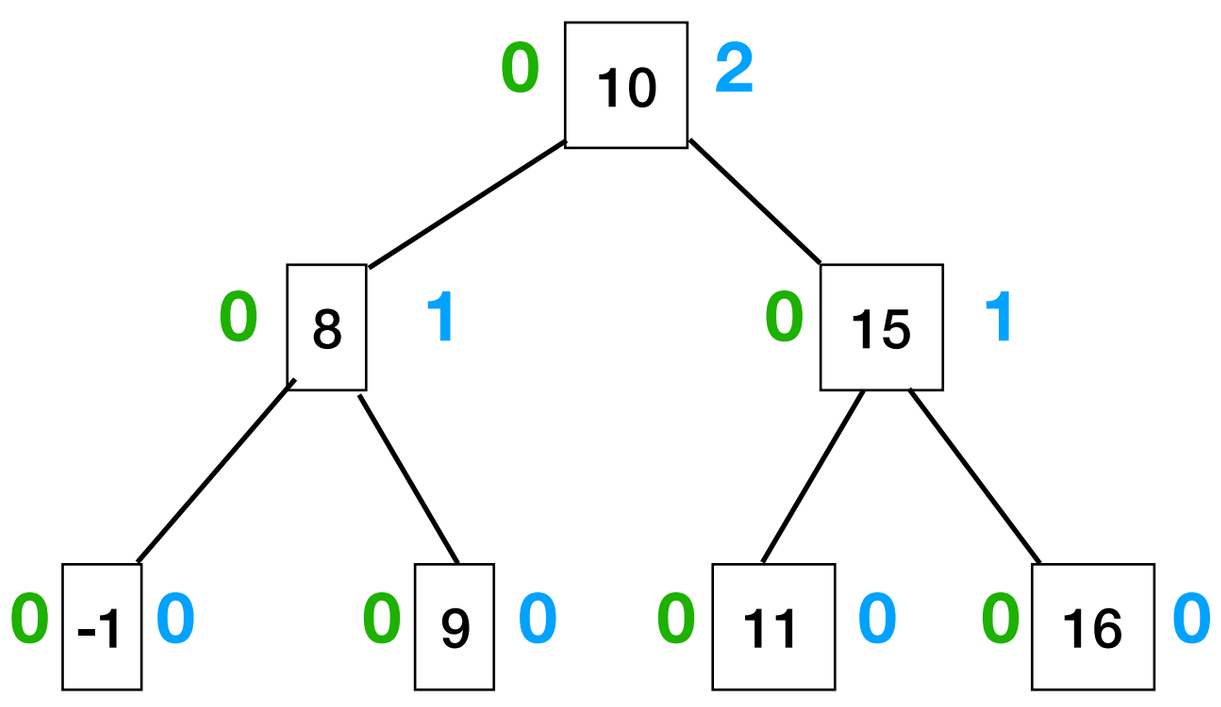




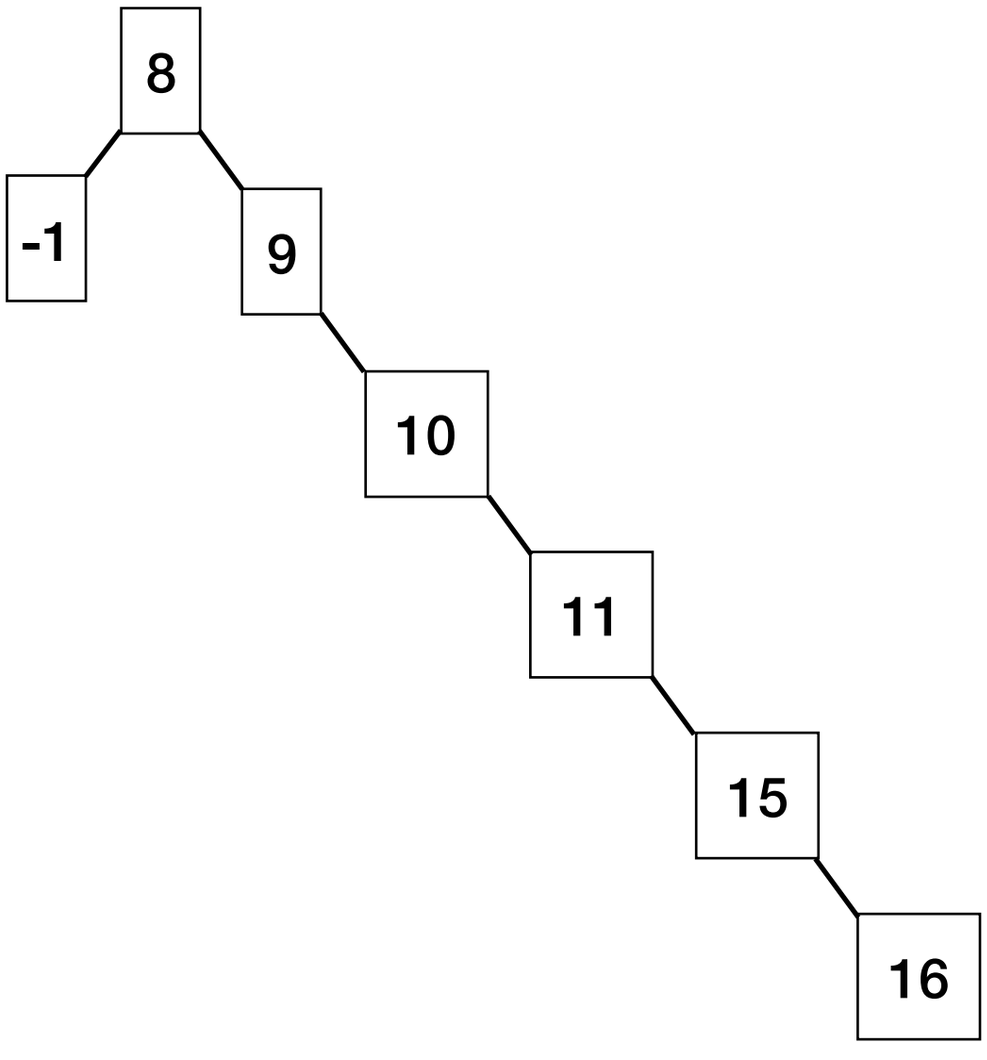
Tree Badness: Balance Factor

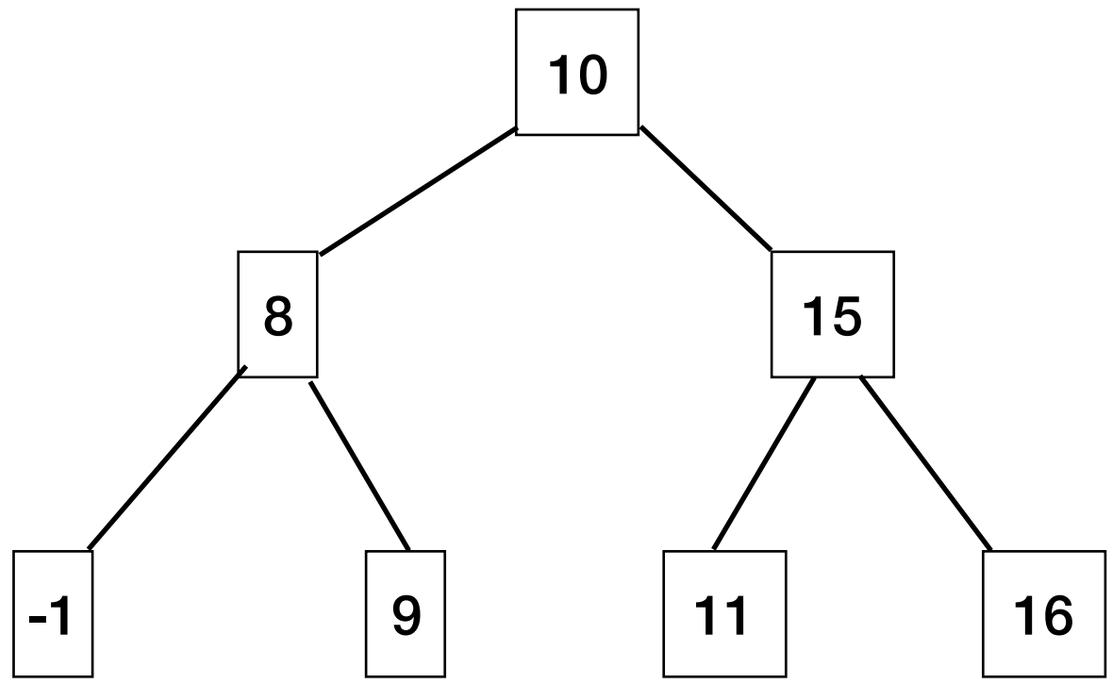
Balance(n): height(n.right) - height(n.left)



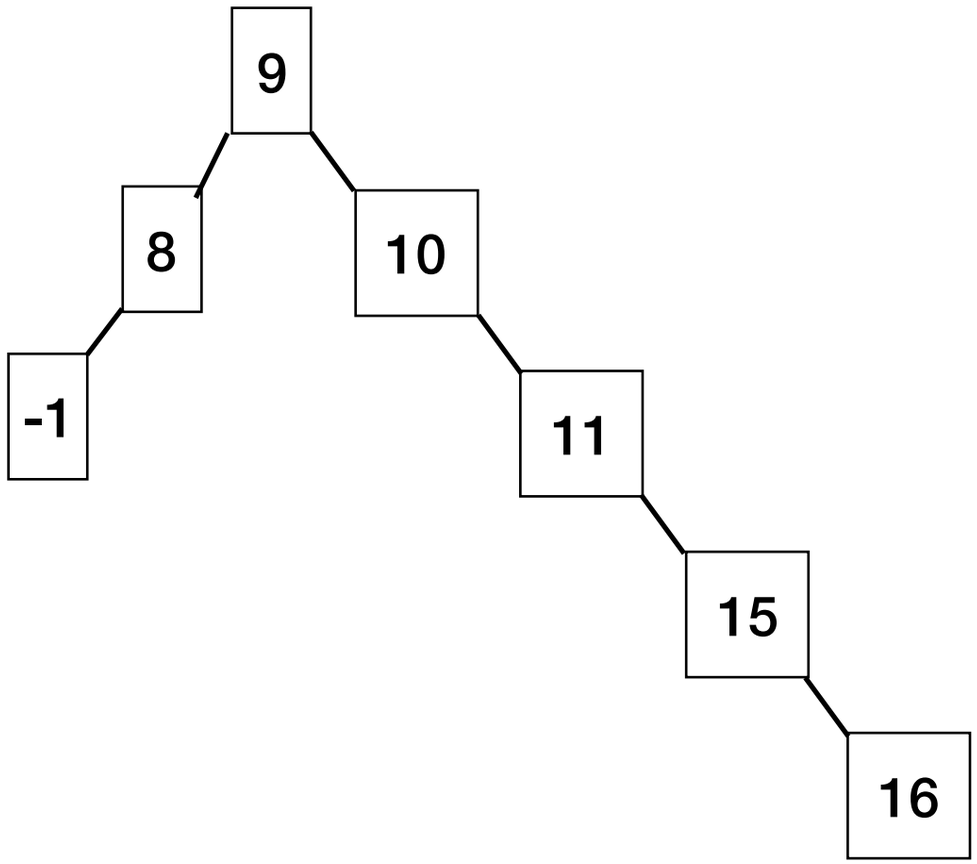


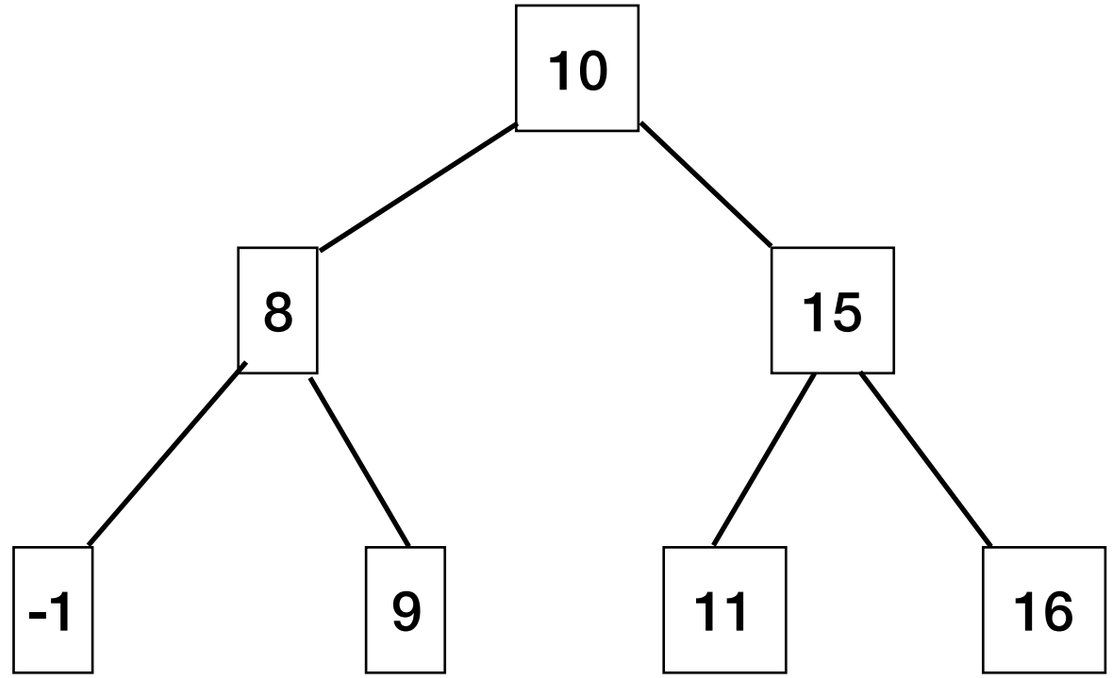
Can we make a tree less bad?



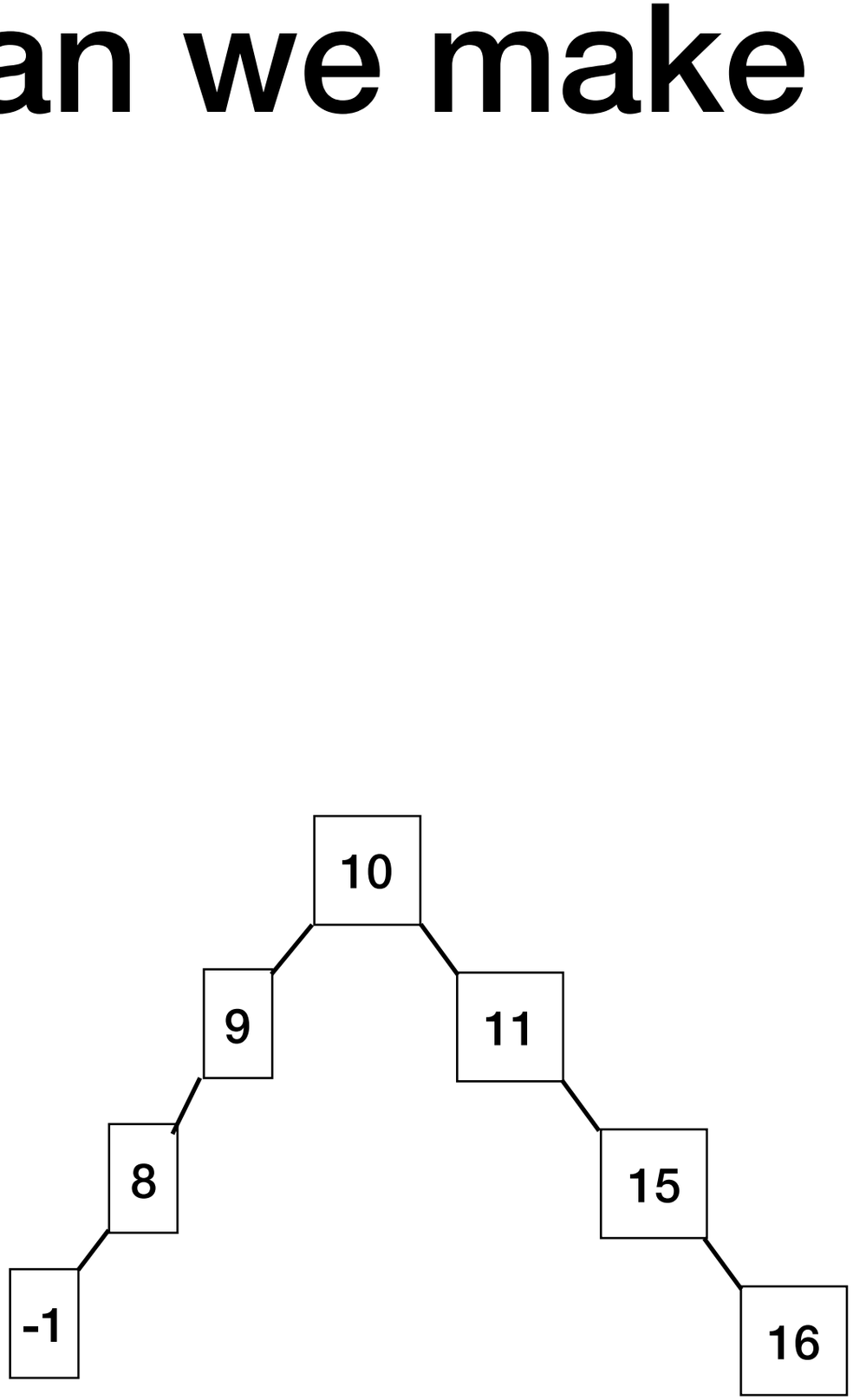


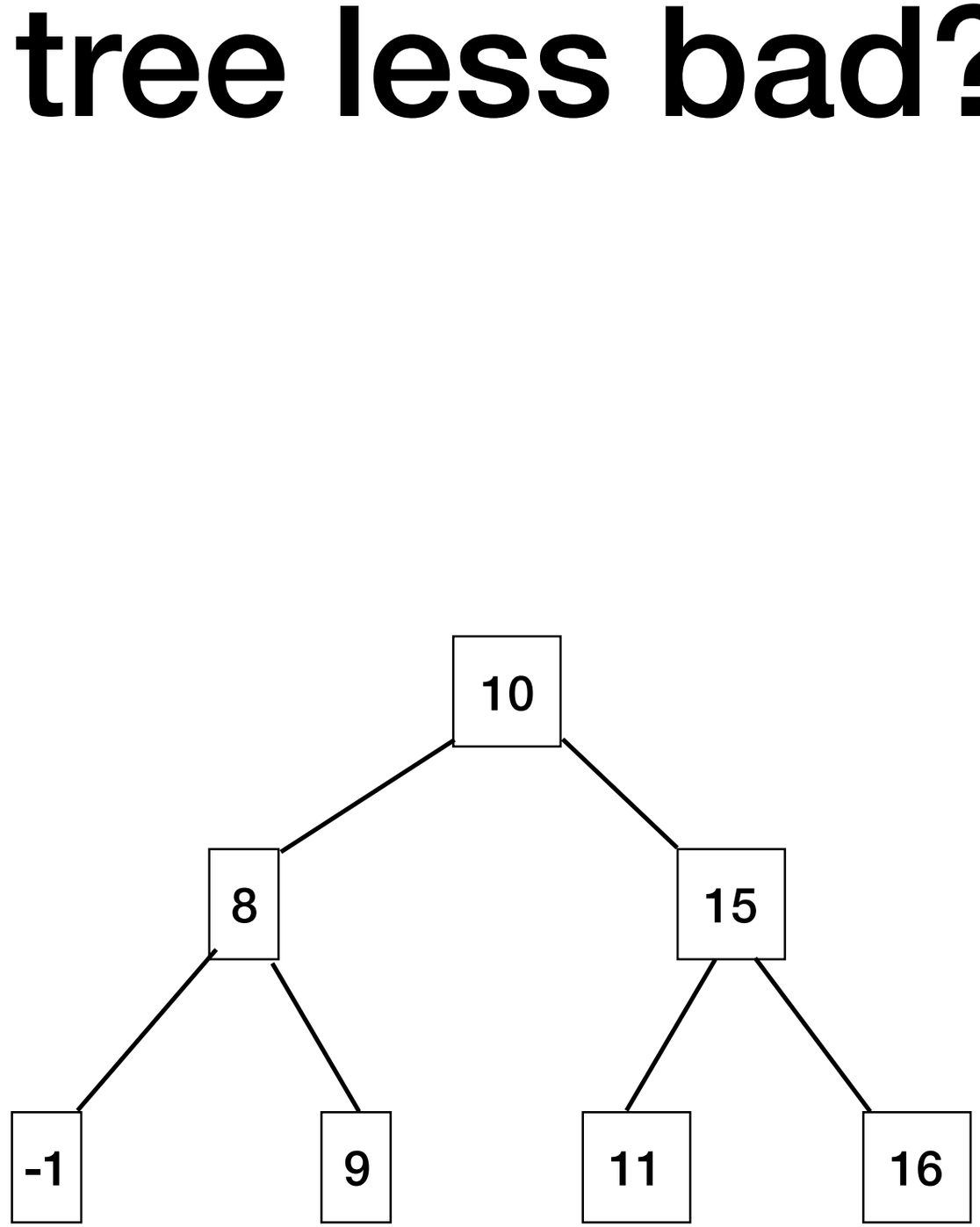
Can we make a tree less bad?



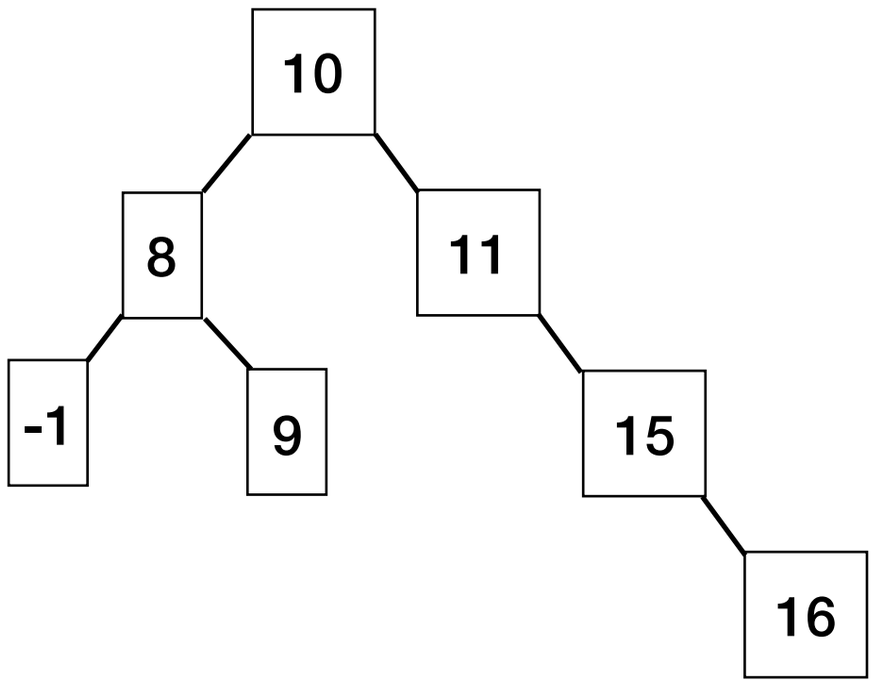


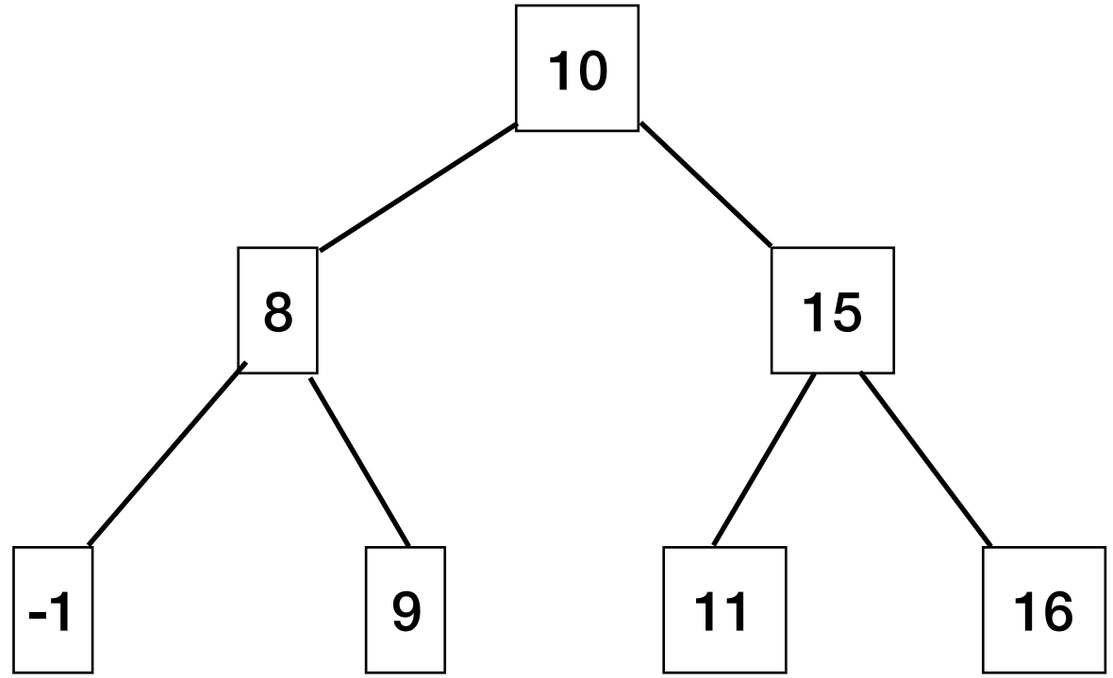
Can we make a tree less bad?



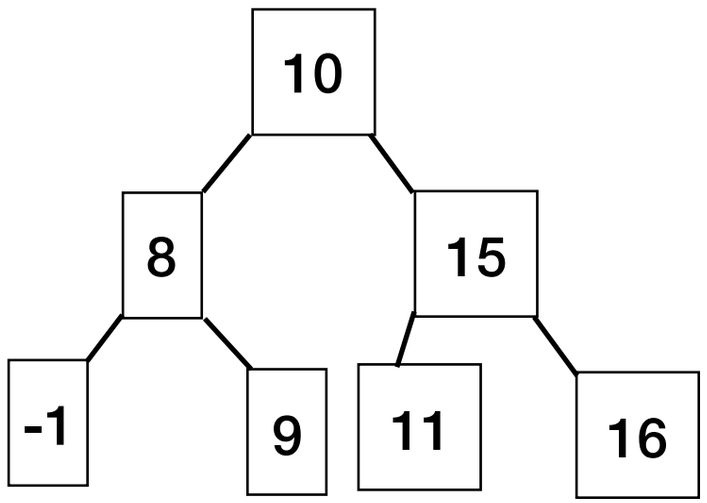


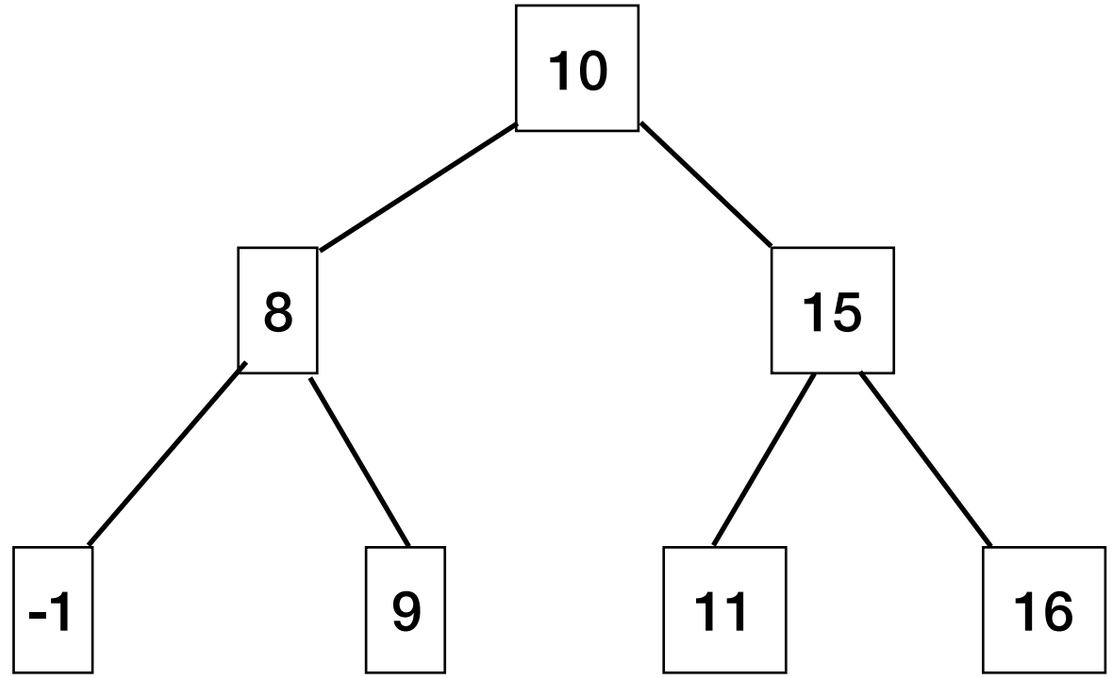
Can we make a tree less bad?





We can make a tree less bad.



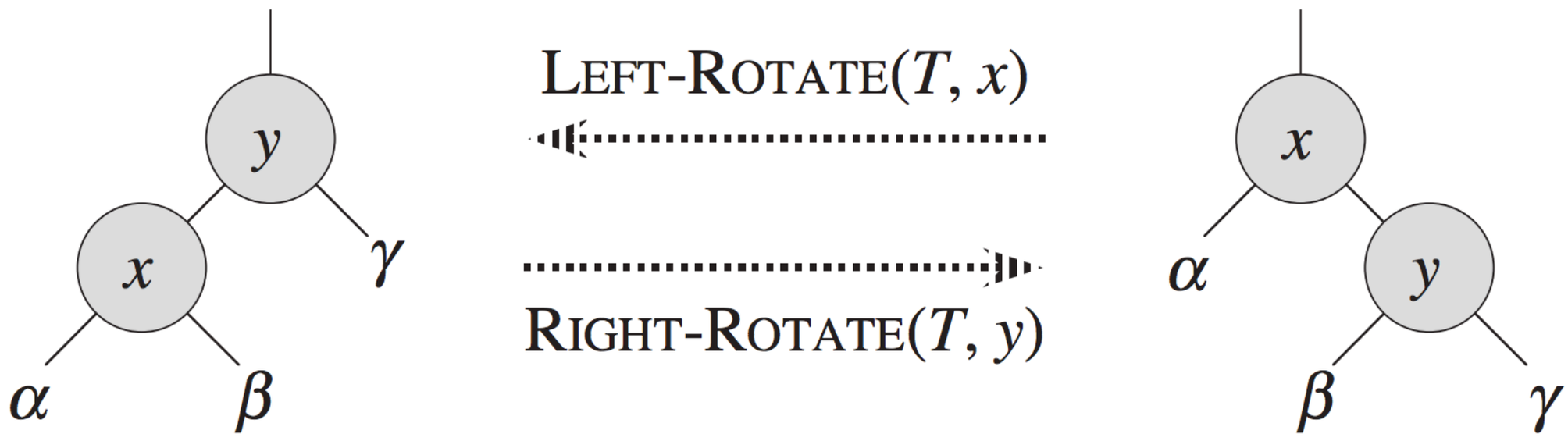


Tree Rotations

modify the structure without violating the BST property.

Steps in left rotation (move y up to its parent's position):

1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. Transfer x itself: x becomes y's left subtree



CLRS Fig 13.2, pg 313

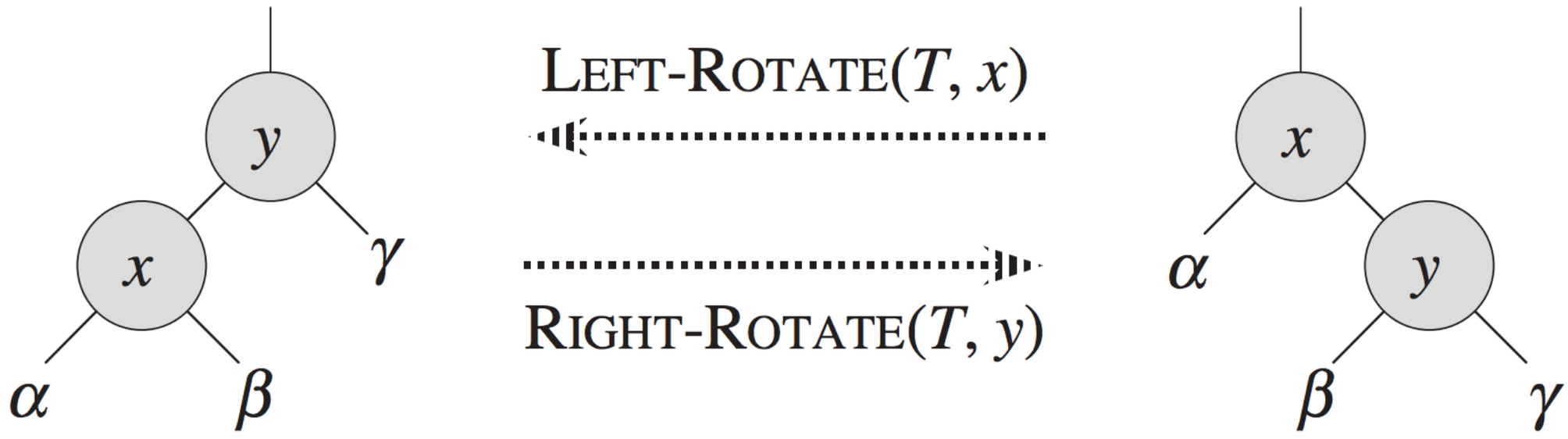
Tree Rotations

modify the structure without violating the BST property.

Steps in left rotation (move y up to its parent's position):

1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. Transfer x itself: x becomes y's left subtree

Details: need to update child, parent, and (possibly) root pointers.

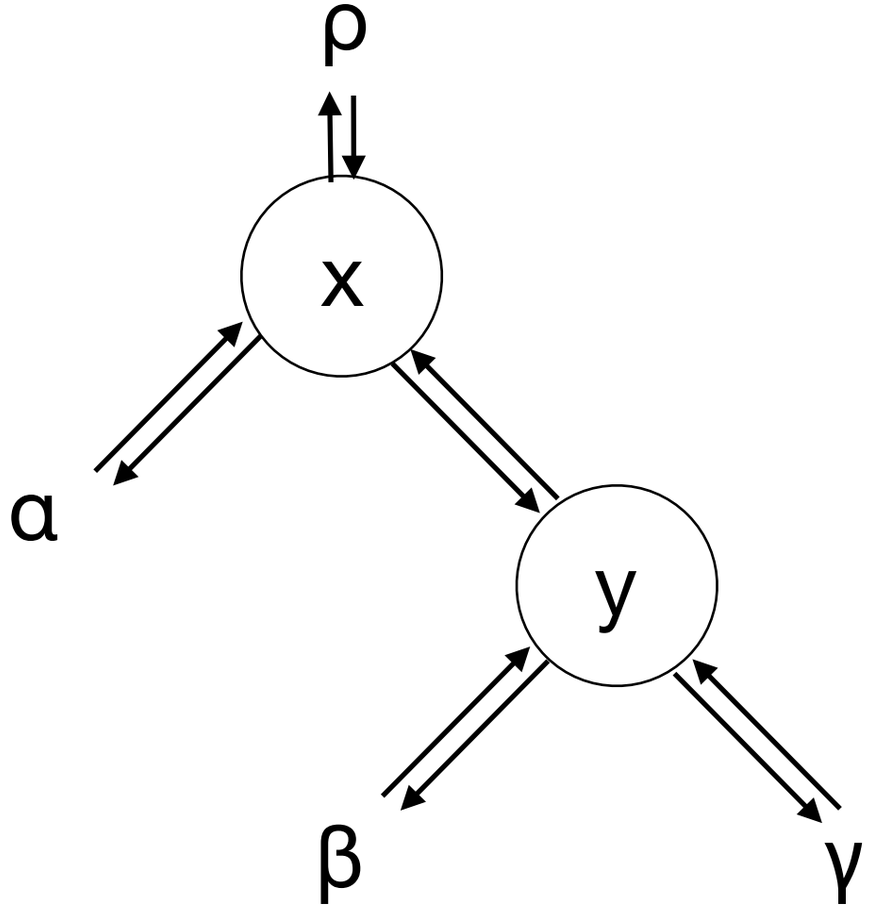


CLRS Fig 13.2, pg 313

Tree Rotations

Steps in left rotation (move y up to its parent's position):

1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. Transfer x itself: x becomes y's left subtree



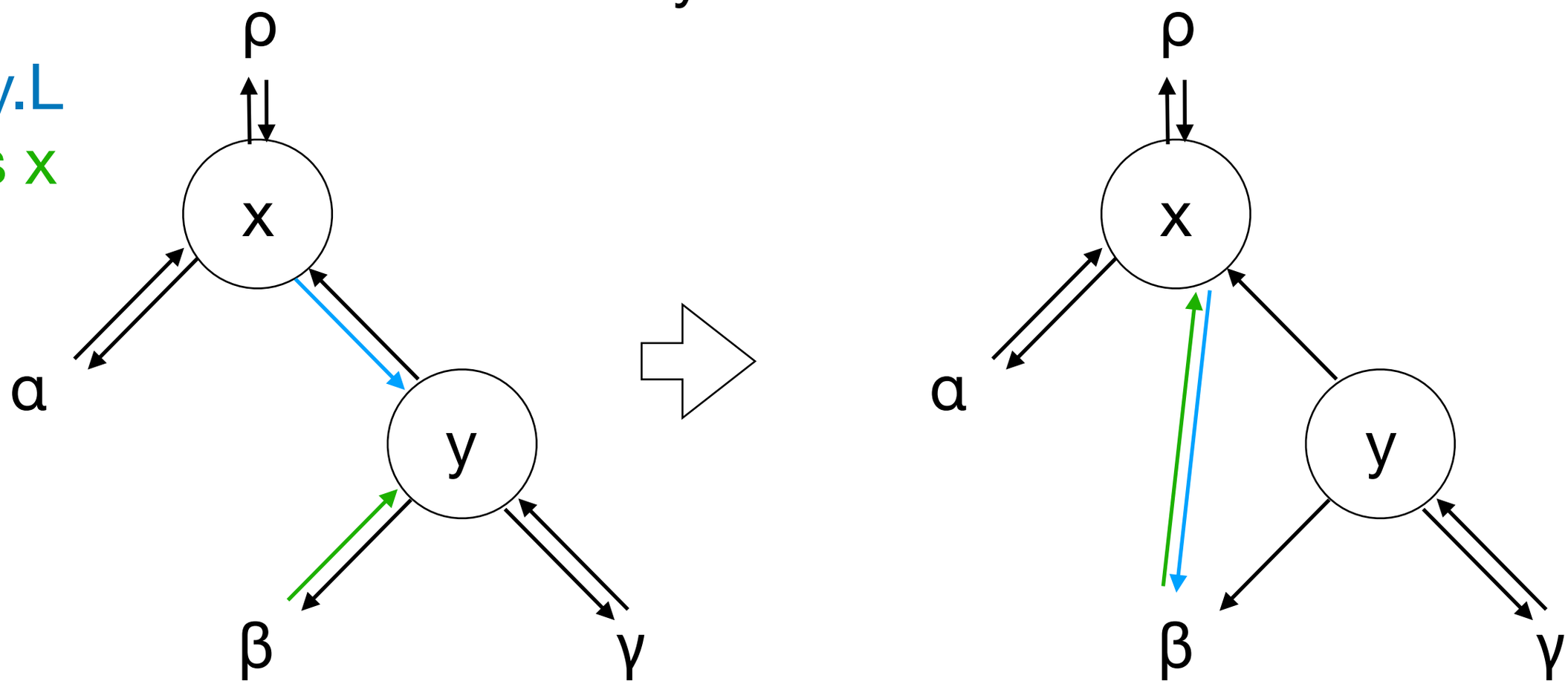
Tree Rotations

Steps in left rotation (move y up to its parent's position):

1. **Transfer β:** x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. Transfer x itself: x becomes y's left subtree

x.R gets y.L

y.L.p gets x

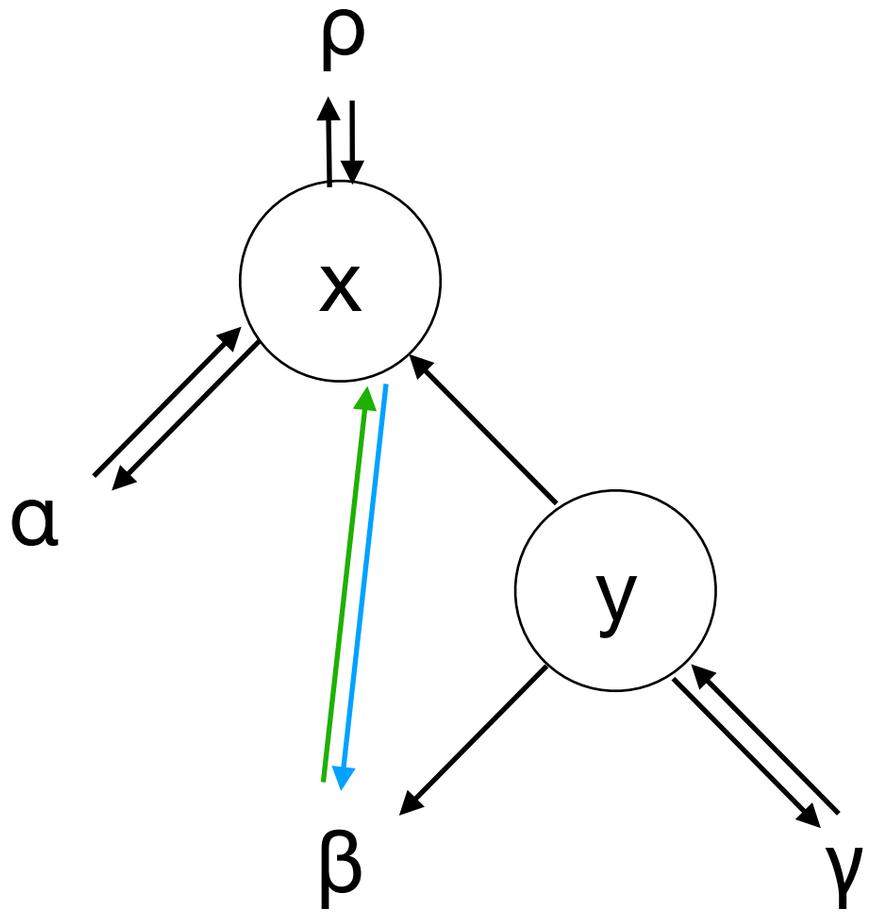


Tree Rotations

Steps in left rotation (move y up to its parent's position):

1. **Transfer β:** x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. Transfer x itself: x becomes y's left subtree

x.R gets y.L
y.L.p gets x

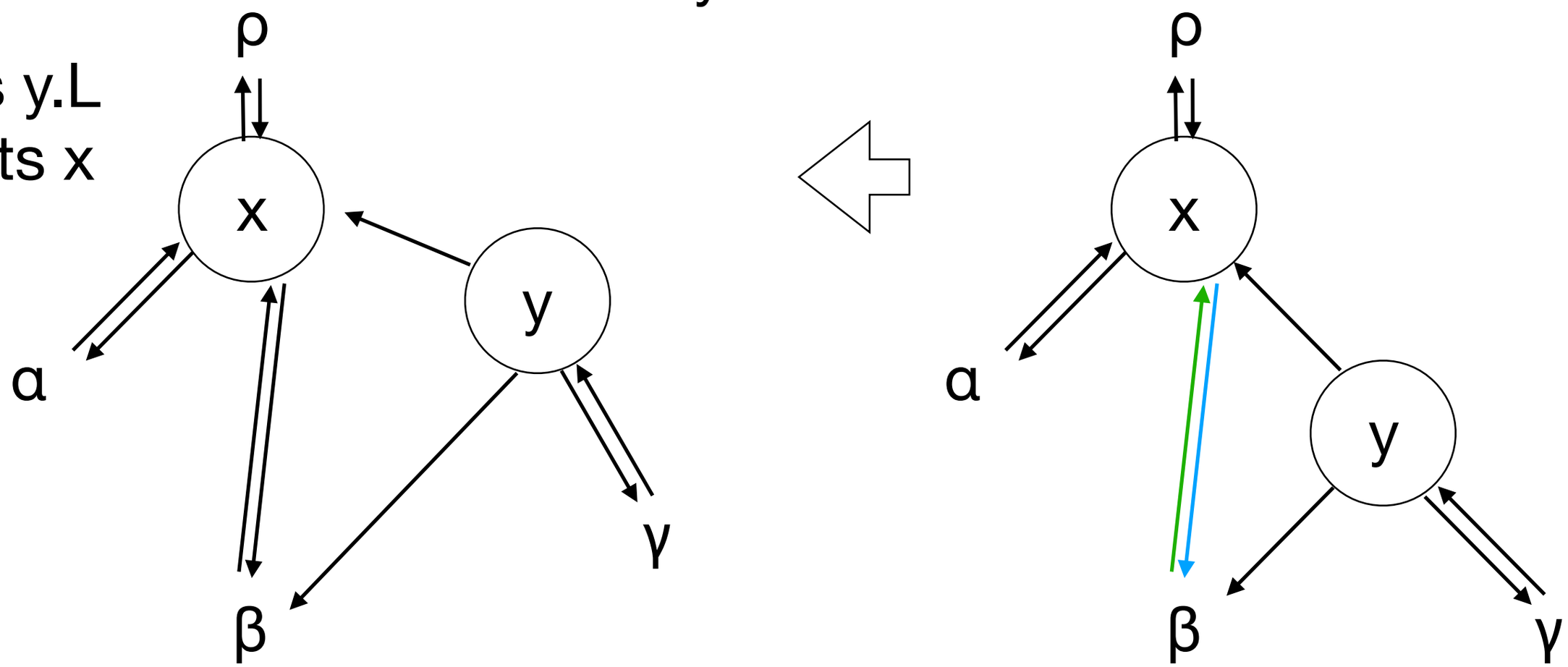


Tree Rotations

Steps in left rotation (move y up to its parent's position):

1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. Transfer x itself: x becomes y's left subtree

x.R gets y.L
y.L.p gets x



(**only** rearranged the picture)

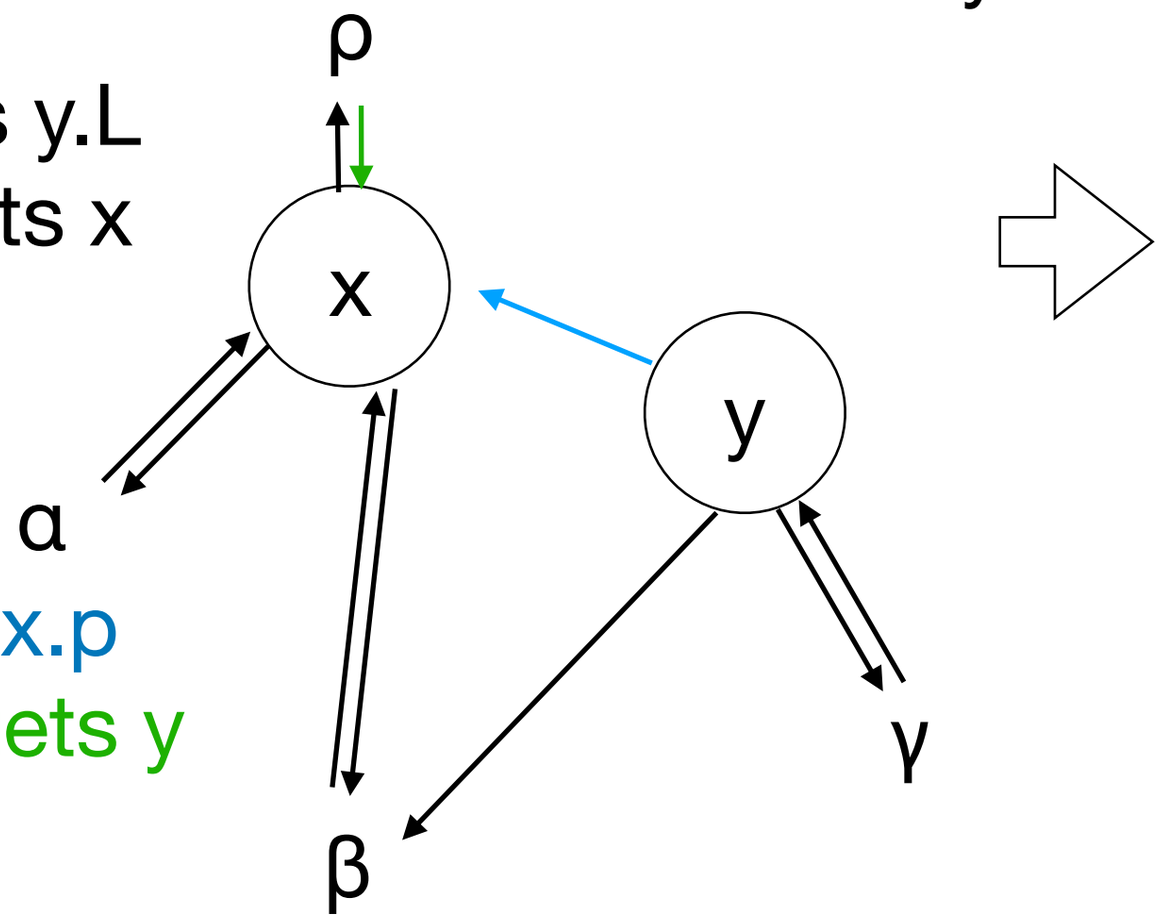
Tree Rotations

Steps in left rotation (move y up to its parent's position):

1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. **Transfer the parent:** y's parent becomes x's old parent
3. Transfer x itself: x becomes y's left subtree

x.R gets y.L

y.L.p gets x



y.p gets x.p

p.[L/R] gets y

Tree Rotations

Steps in left rotation (move y up to its parent's position):

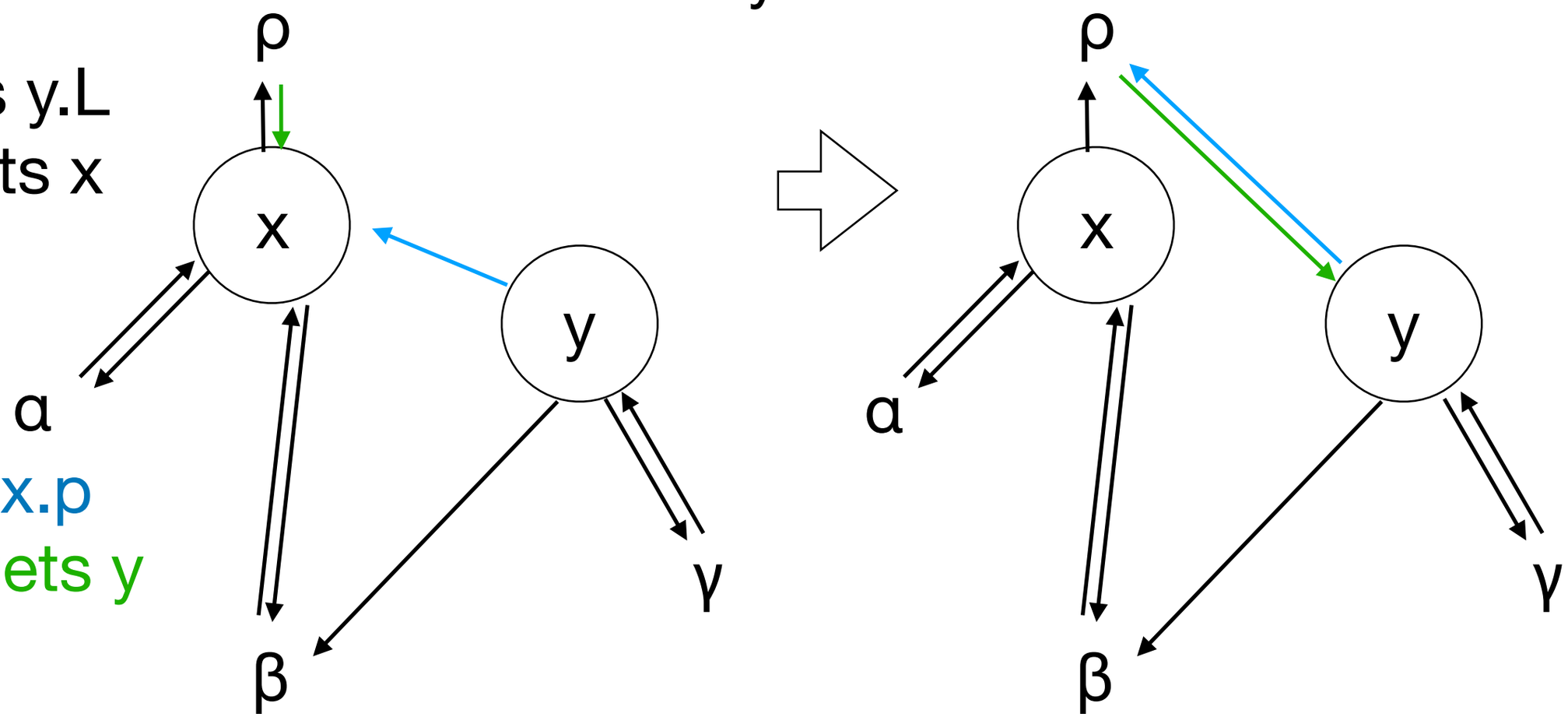
1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. **Transfer the parent:** y's parent becomes x's old parent
3. Transfer x itself: x becomes y's left subtree

x.R gets y.L

y.L.p gets x

y.p gets x.p

p.[L/R] gets y



(what if ρ is null / x was root?)

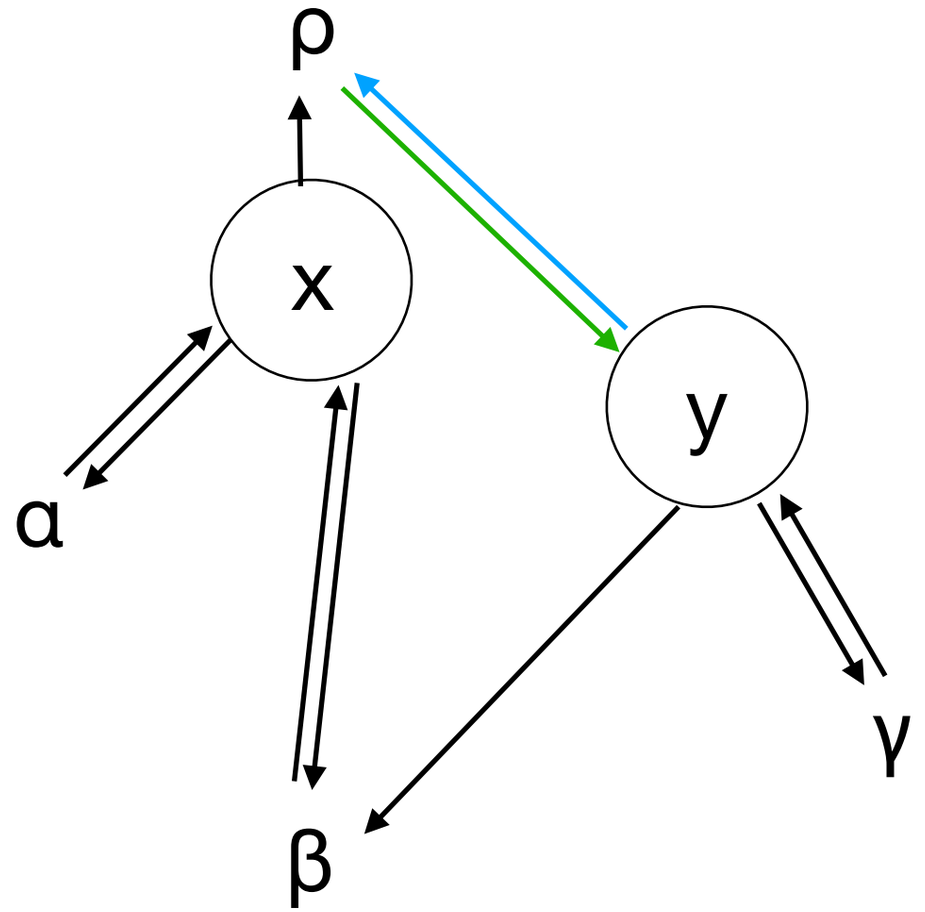
Tree Rotations

Steps in left rotation (move y up to its parent's position):

1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. **Transfer the parent:** y's parent becomes x's old parent
3. Transfer x itself: x becomes y's left subtree

x.R gets y.L
y.L.p gets x

y.p gets x.p
p.[L/R] gets y



(what if ρ is null / x was root?)

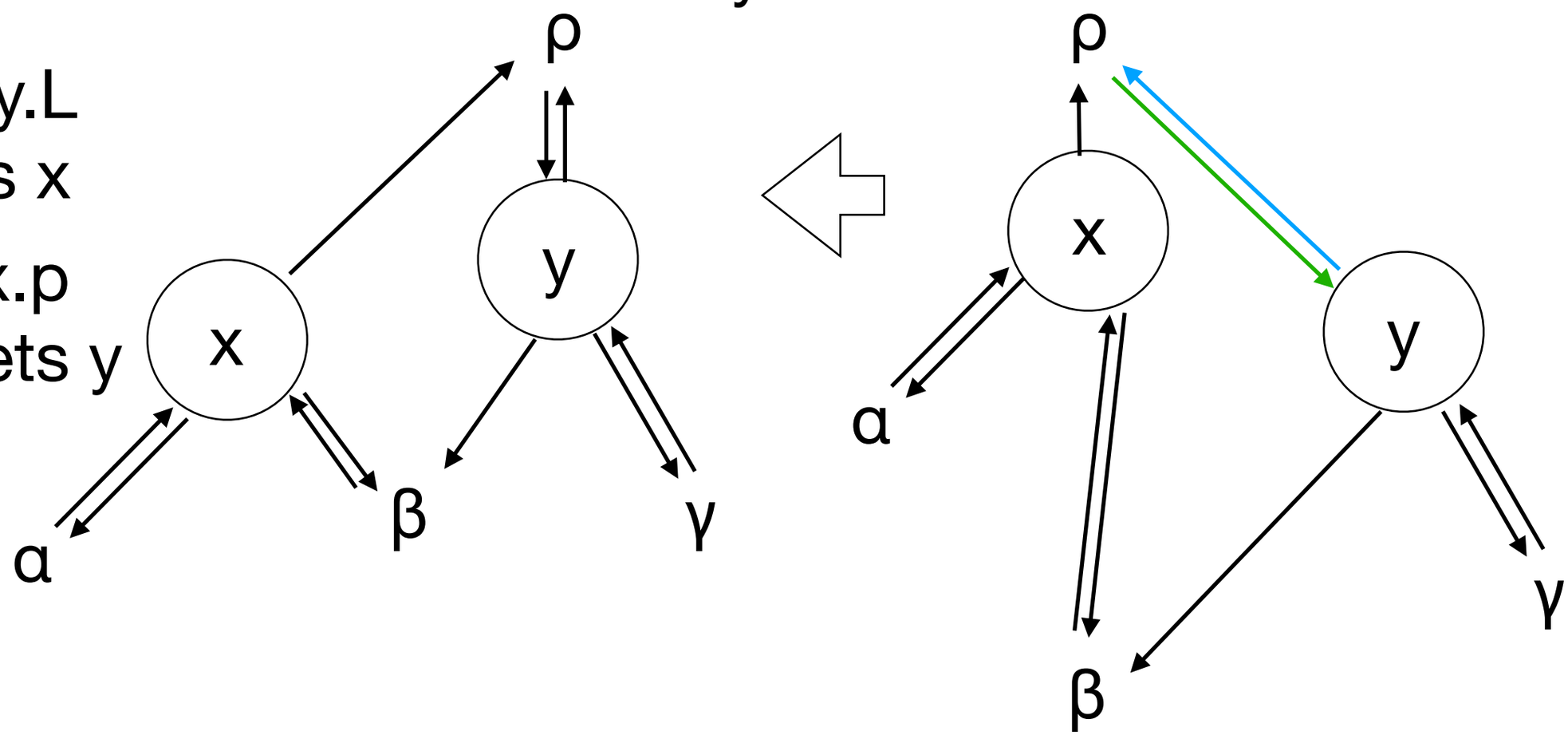
Tree Rotations

Steps in left rotation (move y up to its parent's position):

1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. Transfer x itself: x becomes y's left subtree

x.R gets y.L
y.L.p gets x

y.p gets x.p
p.[L/R] gets y



(**only** rearranged the picture)

Tree Rotations

Steps in left rotation (move y up to its parent's position):

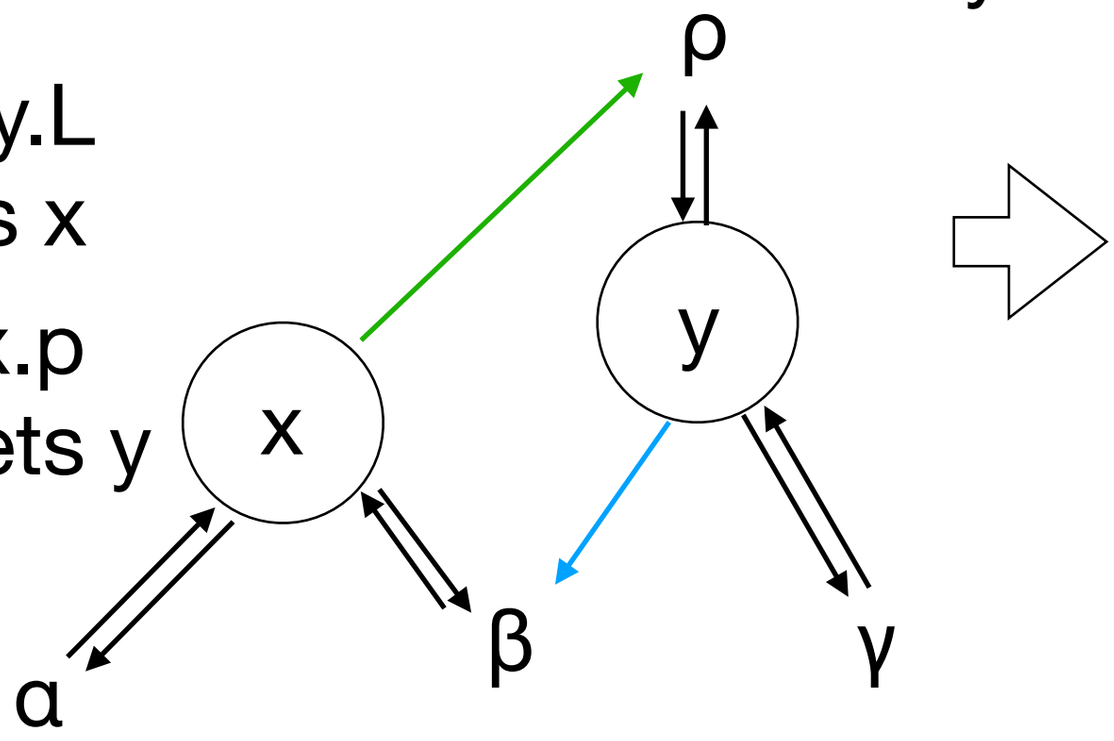
1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. **Transfer x itself:** x becomes y's left subtree

x.R gets y.L

y.L.p gets x

y.p gets x.p

p.[L/R] gets y



y.L gets x

x.p gets y

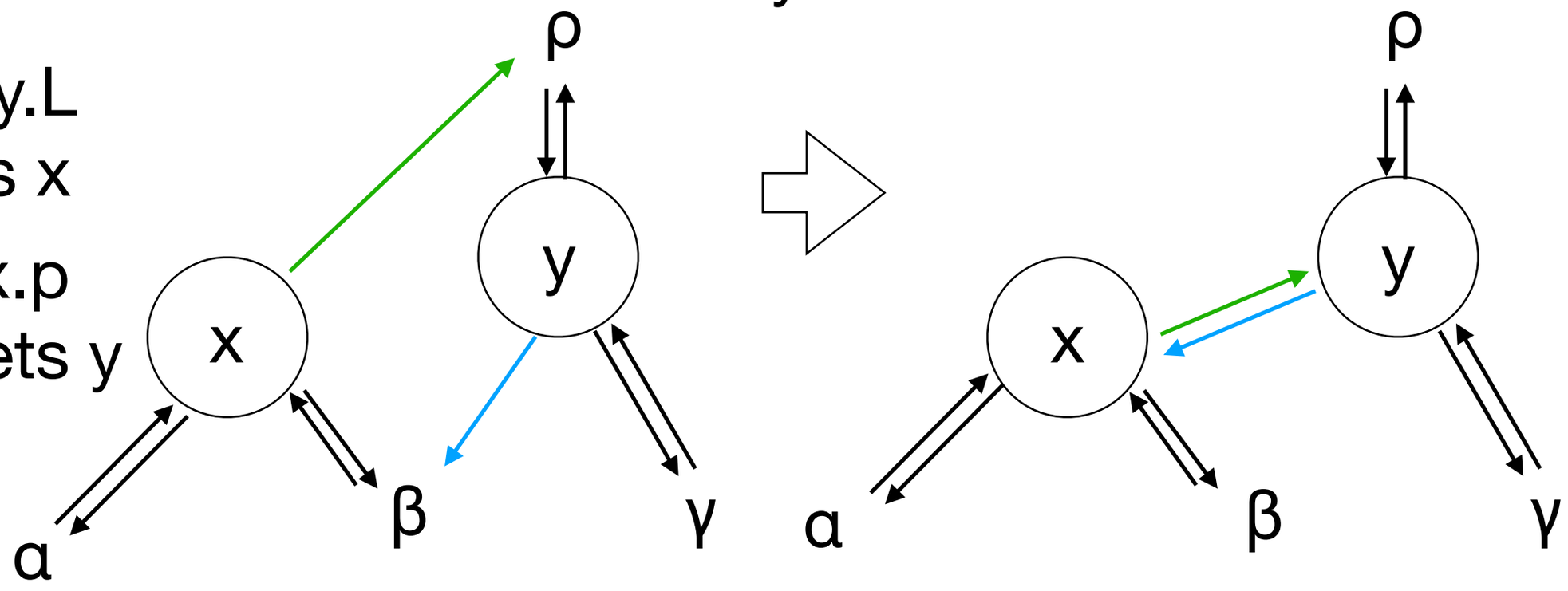
Tree Rotations

Steps in left rotation (move y up to its parent's position):

1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. **Transfer x itself:** x becomes y's left subtree

x.R gets y.L
y.L.p gets x

y.p gets x.p
p.[L/R] gets y



y.L gets x
x.p gets y

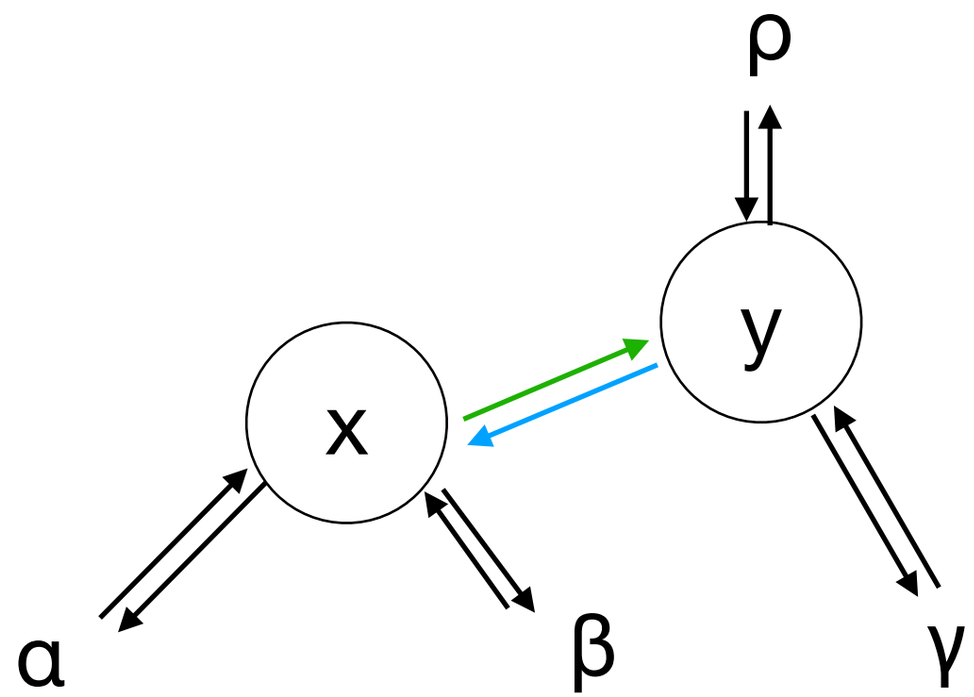
Tree Rotations

Steps in left rotation (move y up to its parent's position):

1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. **Transfer x itself:** x becomes y's left subtree

x.R gets y.L
y.L.p gets x

y.p gets x.p
p.[L/R] gets y



y.L gets x
x.p gets y

Tree Rotations

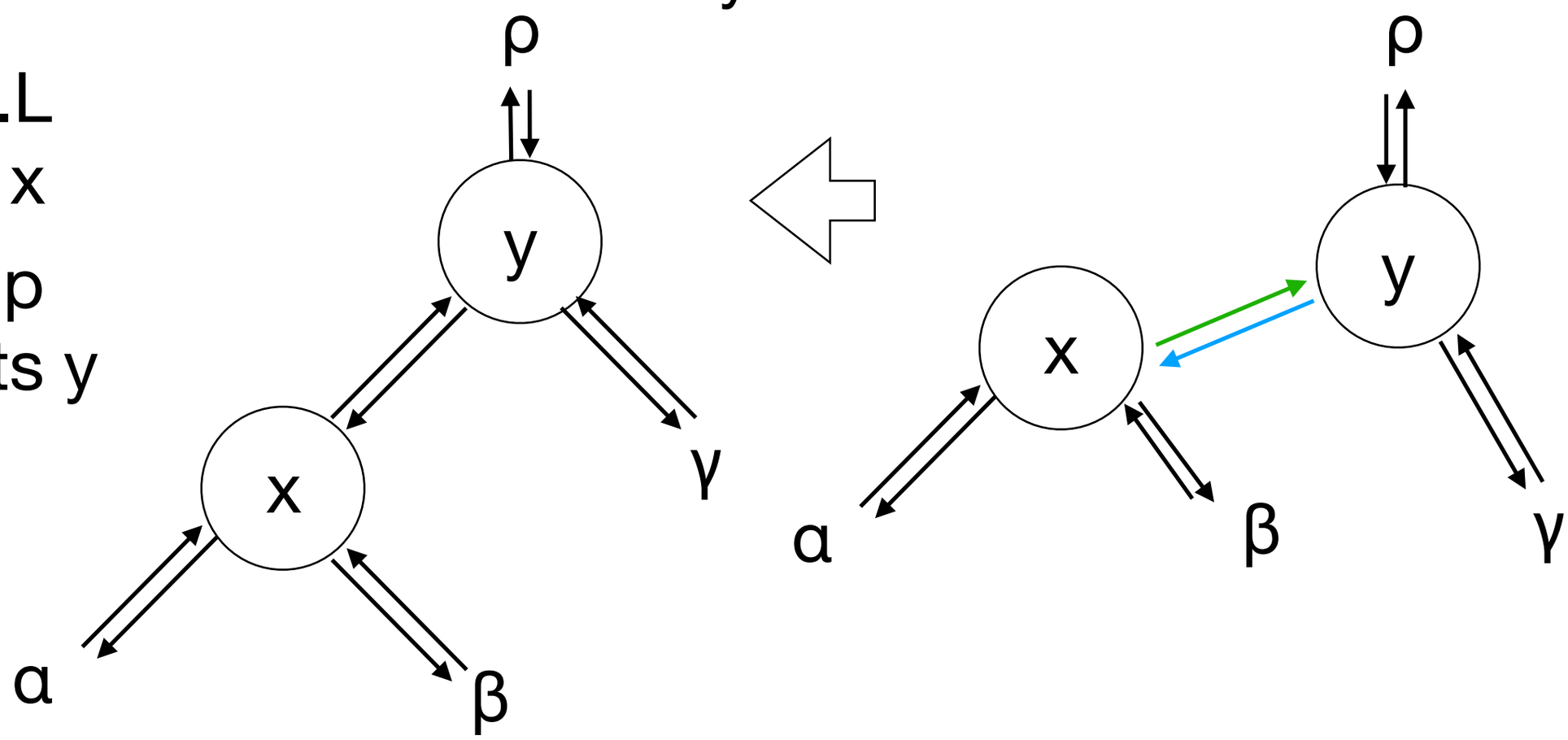
Steps in left rotation (move y up to its parent's position):

1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. Transfer x itself: x becomes y's left subtree

x.R gets y.L
y.L.p gets x

y.p gets x.p
p.[L/R] gets y

y.L gets x
x.p gets y



(**only** rearranged the picture)

Tree Rotations

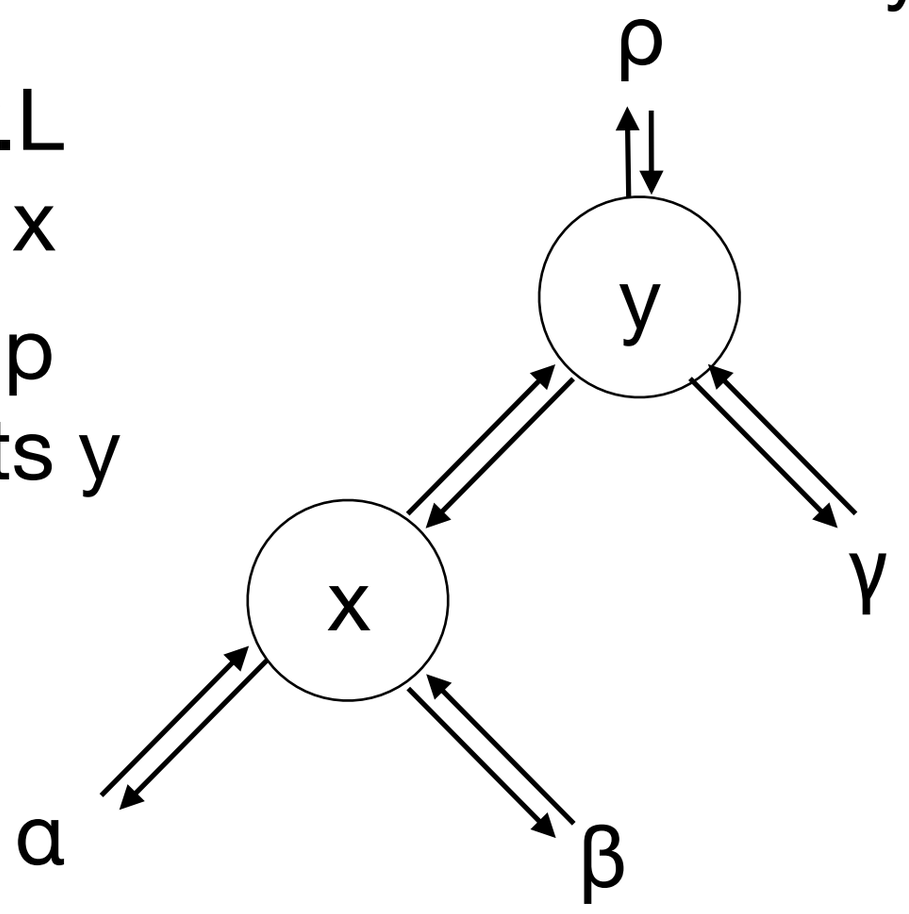
Steps in left rotation (move y up to its parent's position):

1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. Transfer x itself: x becomes y's left subtree

x.R gets y.L
y.L.p gets x

y.p gets x.p
p.[L/R] gets y

y.L gets x
x.p gets y



Tree Rotations

Steps in left rotation (move y up to its parent's position):

1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. Transfer x itself: x becomes y's left subtree

x.R gets y.L

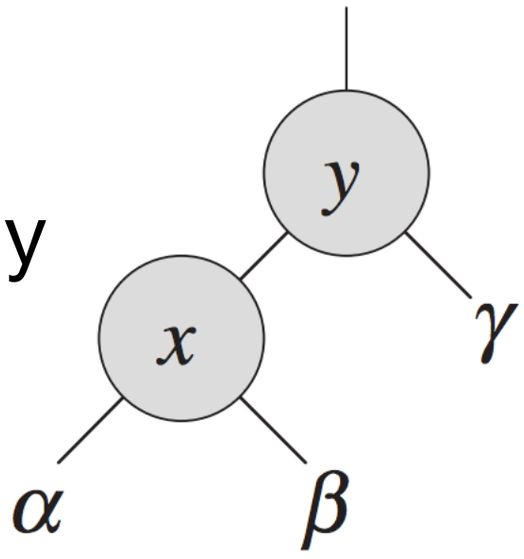
y.L.p gets x

y.p gets x.p

p.[L/R] gets y

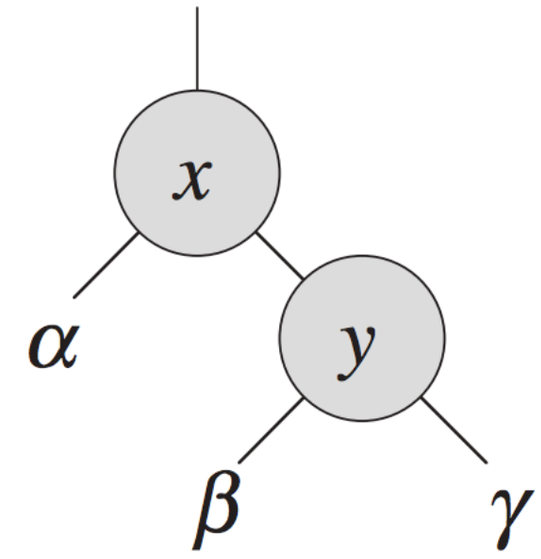
y.L gets x

x.p gets y



LEFT-ROTATE(T, x)

RIGHT-ROTATE(T, y)



Overall Transformation

Pseudocode from CLRS

```
LEFT-ROTATE(T, x)
1   y = x.right                    // set y
2   x.right = y.left               // turn y's left subtree into x's right subtree
3   if y.left ≠ T.nil
4       y.left.p = x
5   y.p = x.p                      // link x's parent to y
6   if x.p == T.nil
7       T.root = y
8   elseif x == x.p.left
9       x.p.left = y
10  else x.p.right = y
11  y.left = x                     // put x on y's left
12  x.p = y
```

1. xfer β (lines 1–4)

2. xfer parent (lines 6–10)

3. xfer x (lines 11–12)

Notational quirk: assume T.nil means “null”

Tree Rotations

Steps in **left** rotation (move y up to x's position):

1. Transfer β
2. Transfer the parent
3. Transfer x itself

x.R gets y.L
y.L.p gets x

y.p gets x.p
p.[L/R] gets y

y.L gets x
x.p gets y