CSCI 241

Scott Wehrwein

Tree Rotations

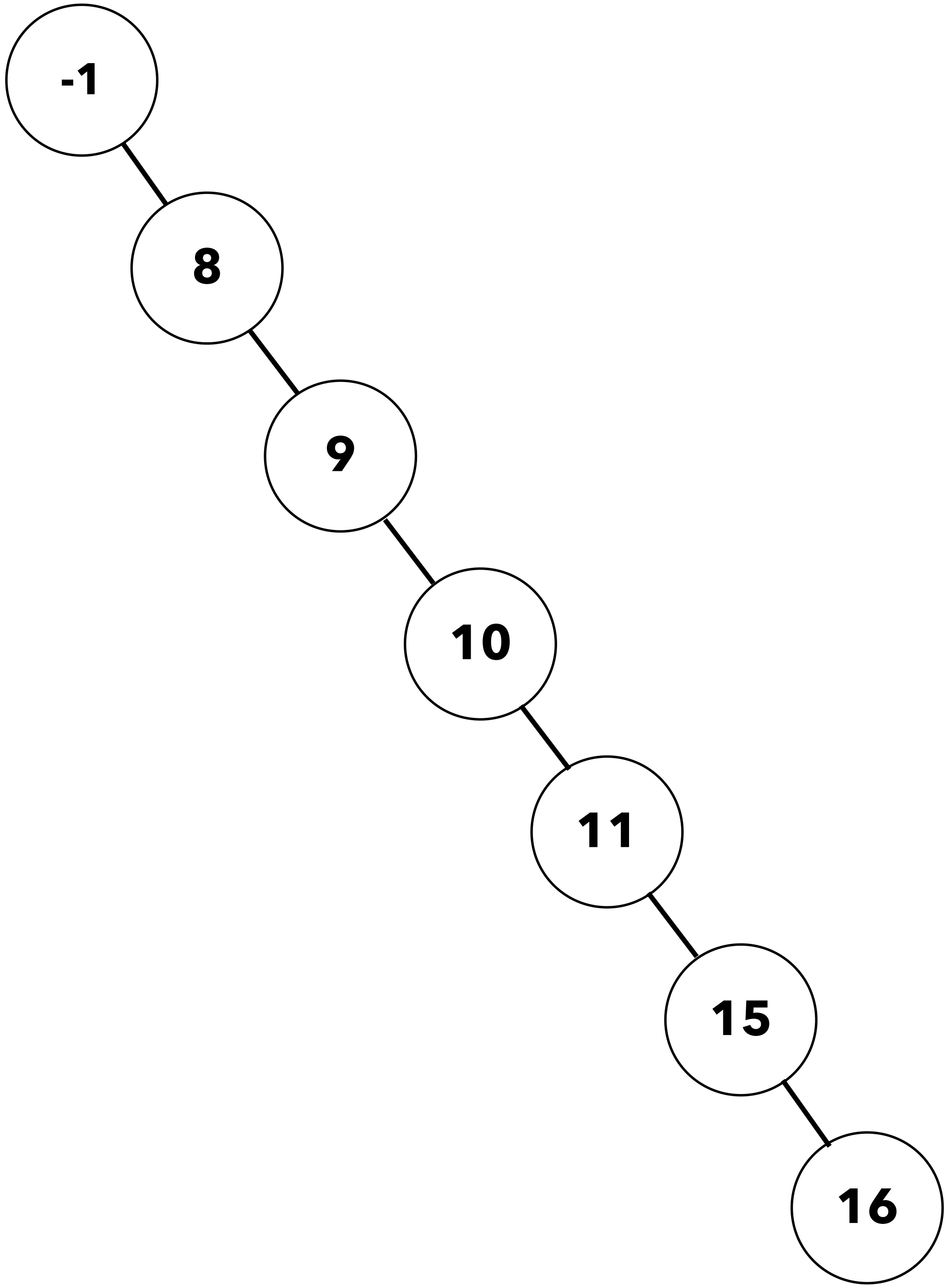
Goals

Be able to execute BST rotations on paper.

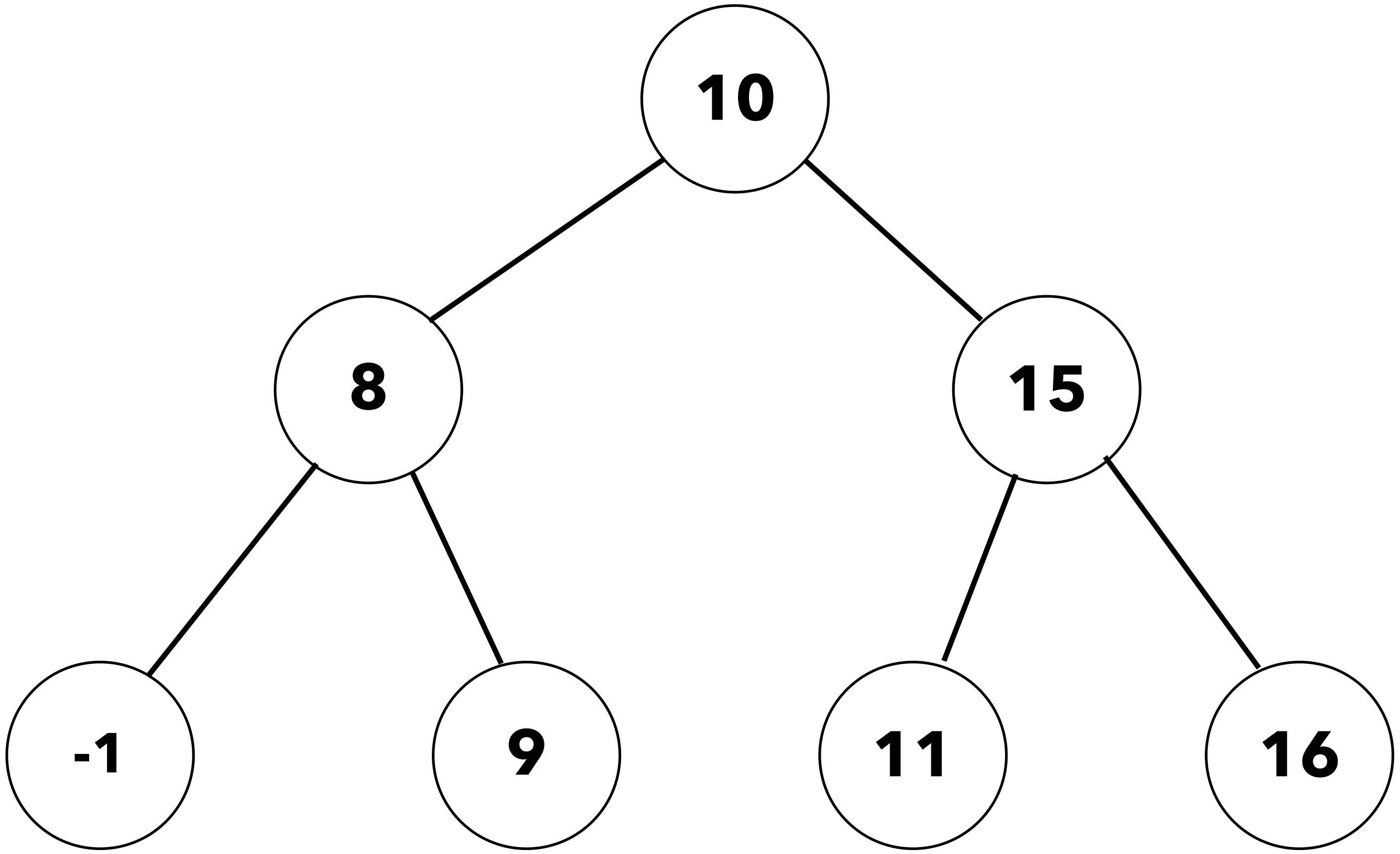
Be prepared to implement BST rotations.

Measuring Badness

Bad tree =(



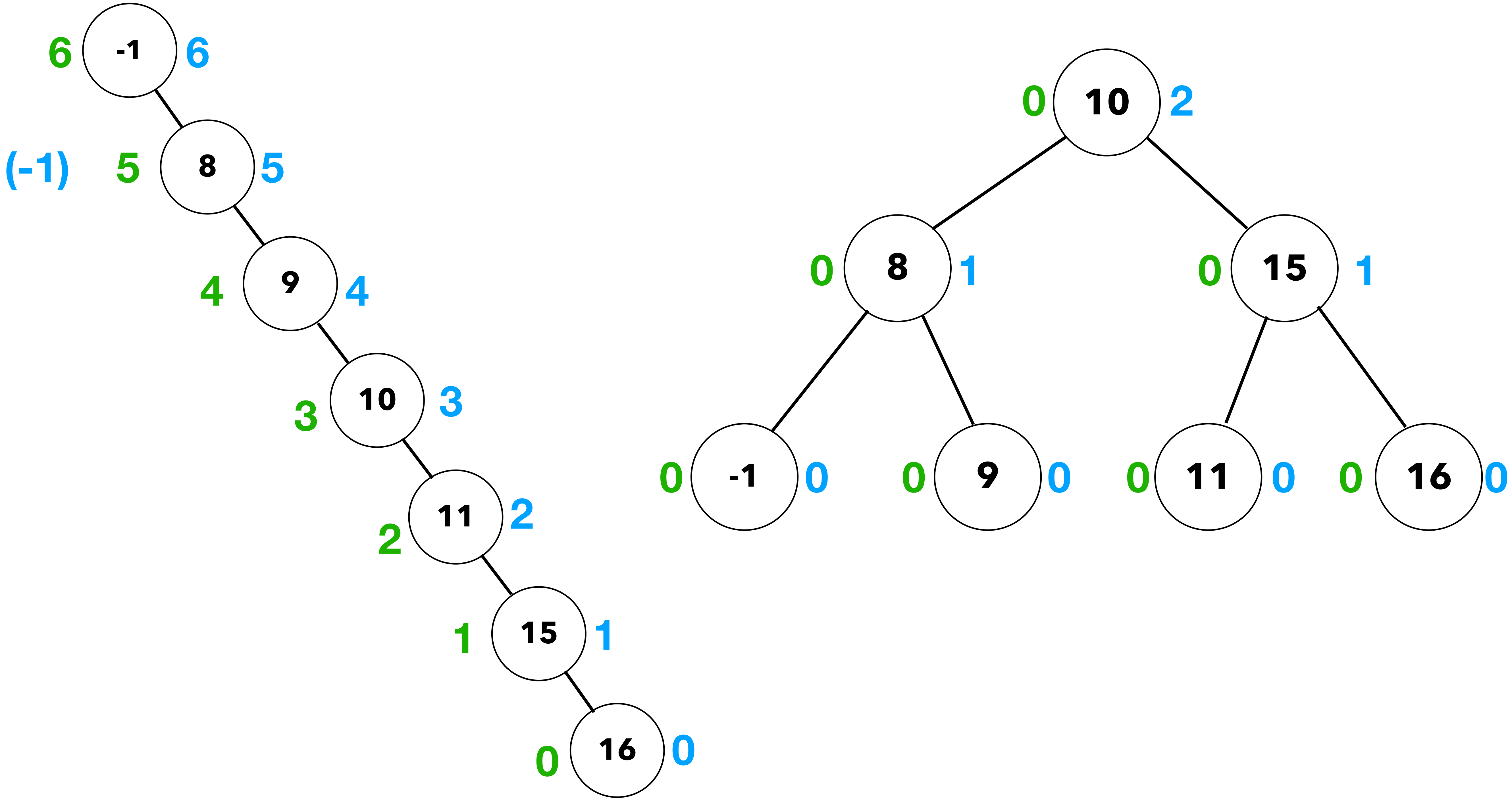
Good tree =)



how bad? how good?

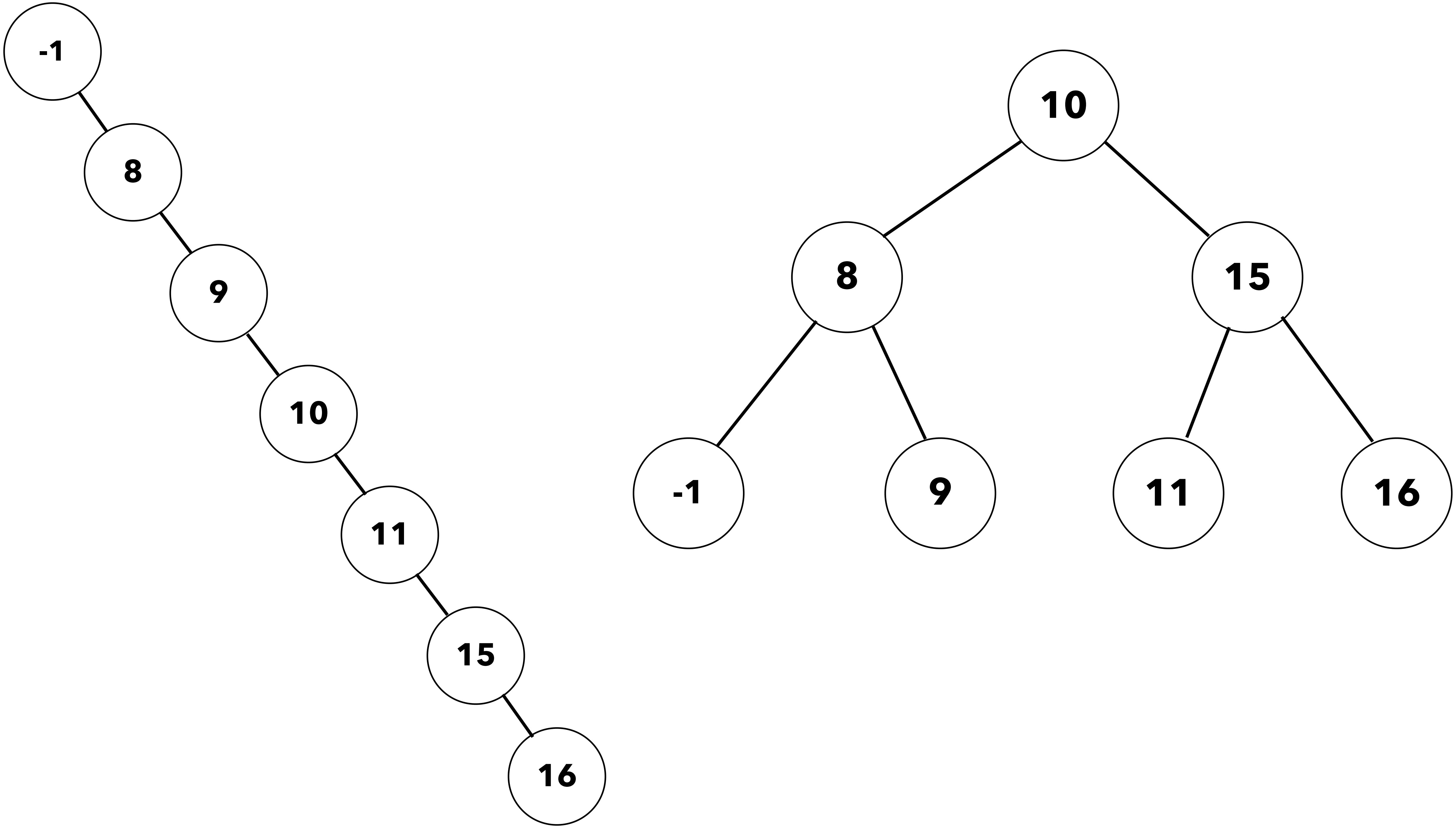
Measuring Badness

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



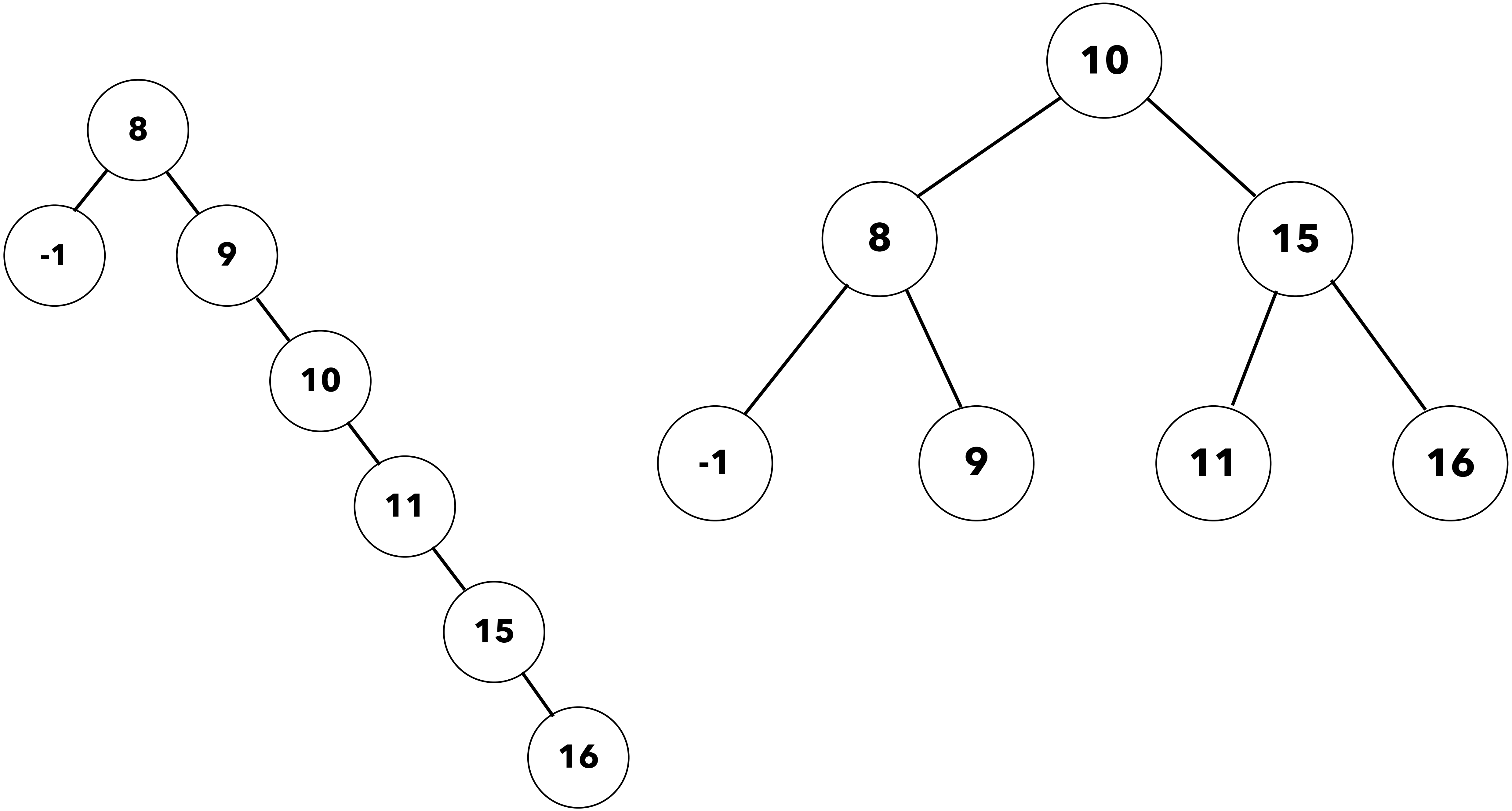
Tree Badness

Hey Jude: can we take a bad tree and make it better?



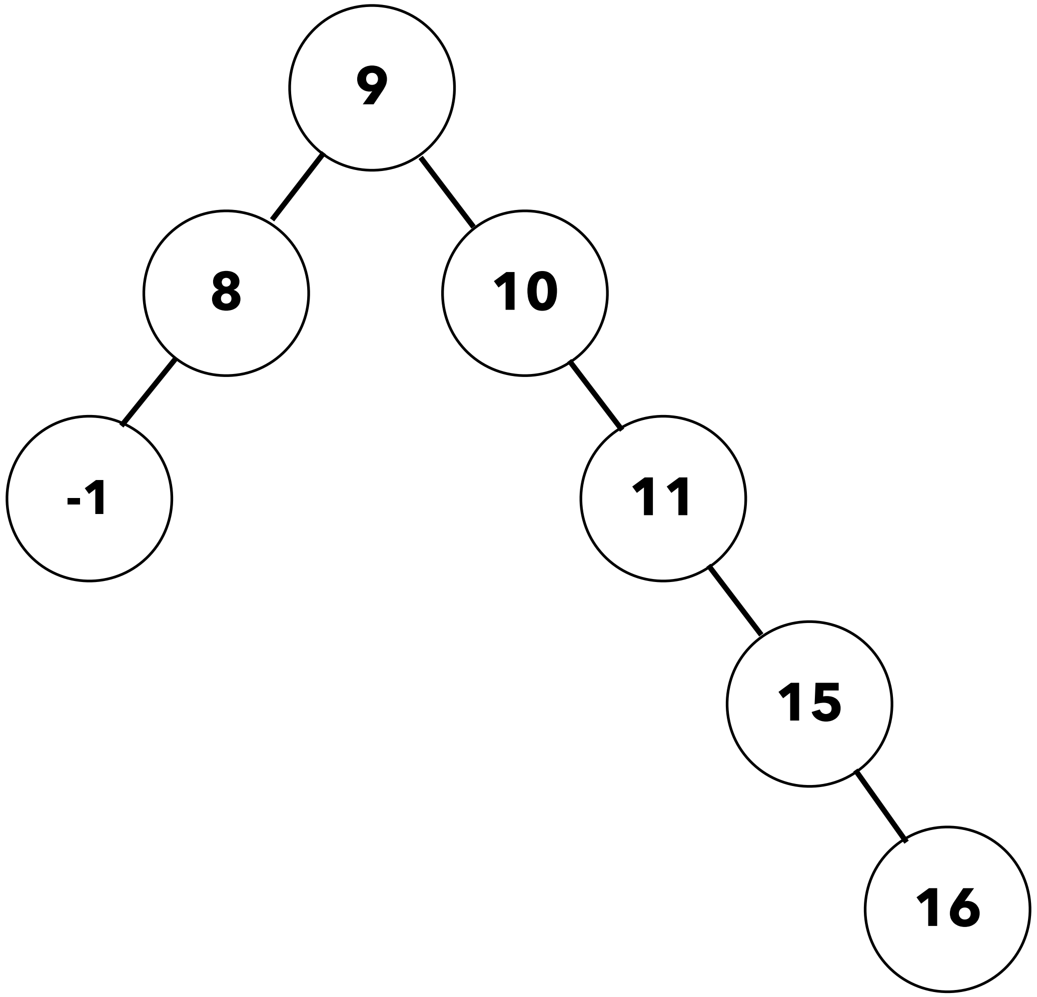
Tree Badness

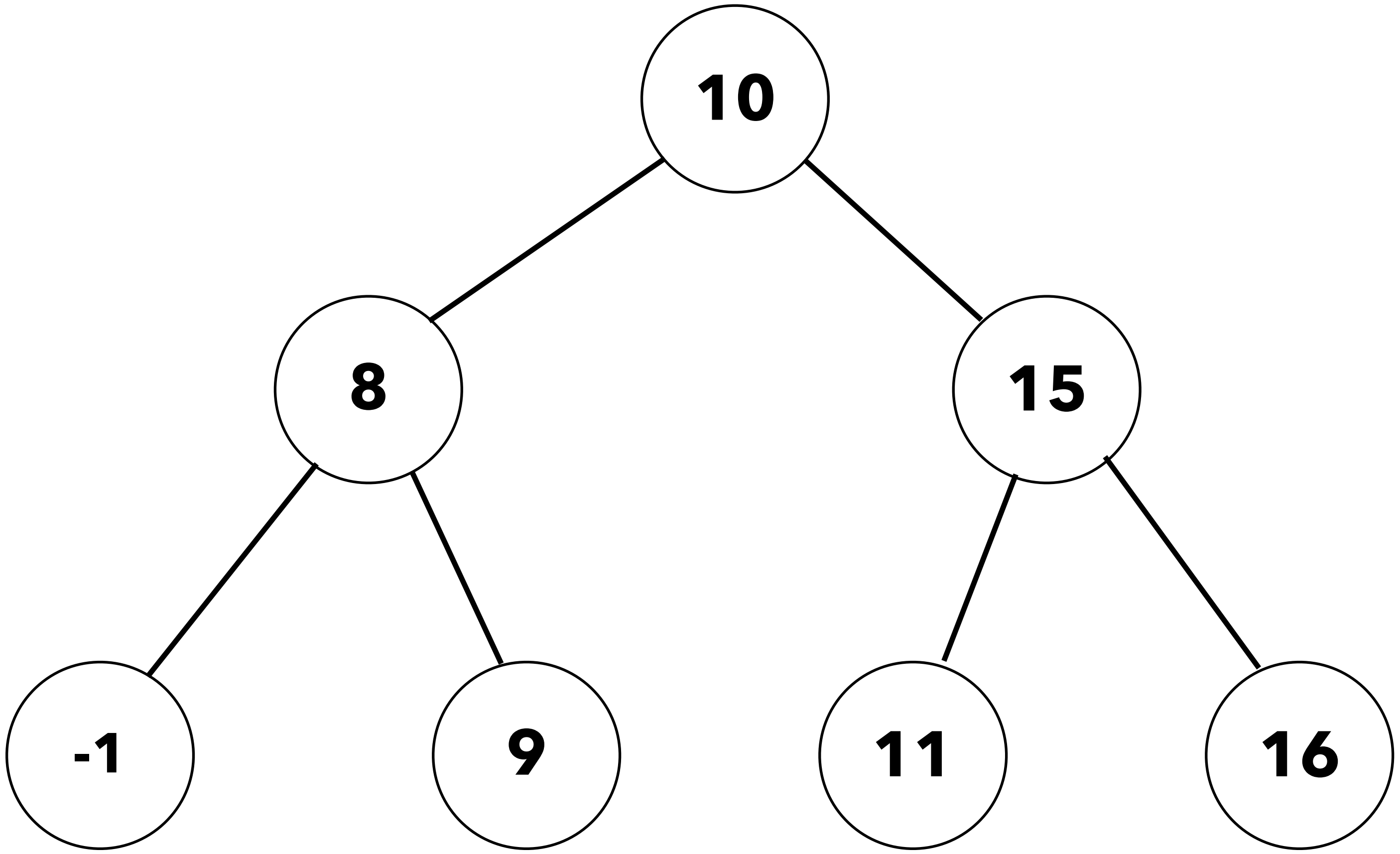
Hey Jude: can we take a bad tree and make it better?



Tree Badness

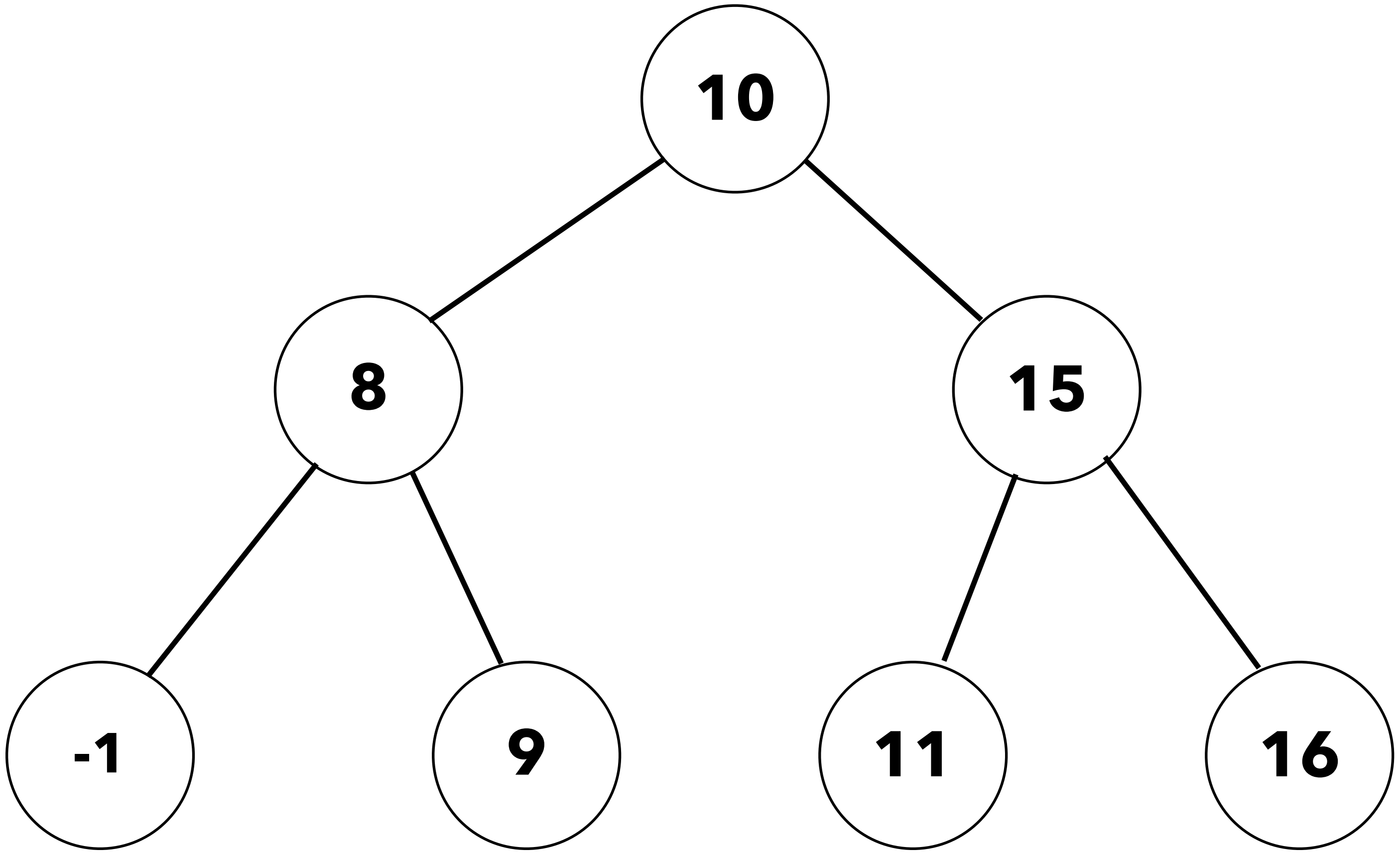
Hey Jude: can we take a bad tree and make it better?

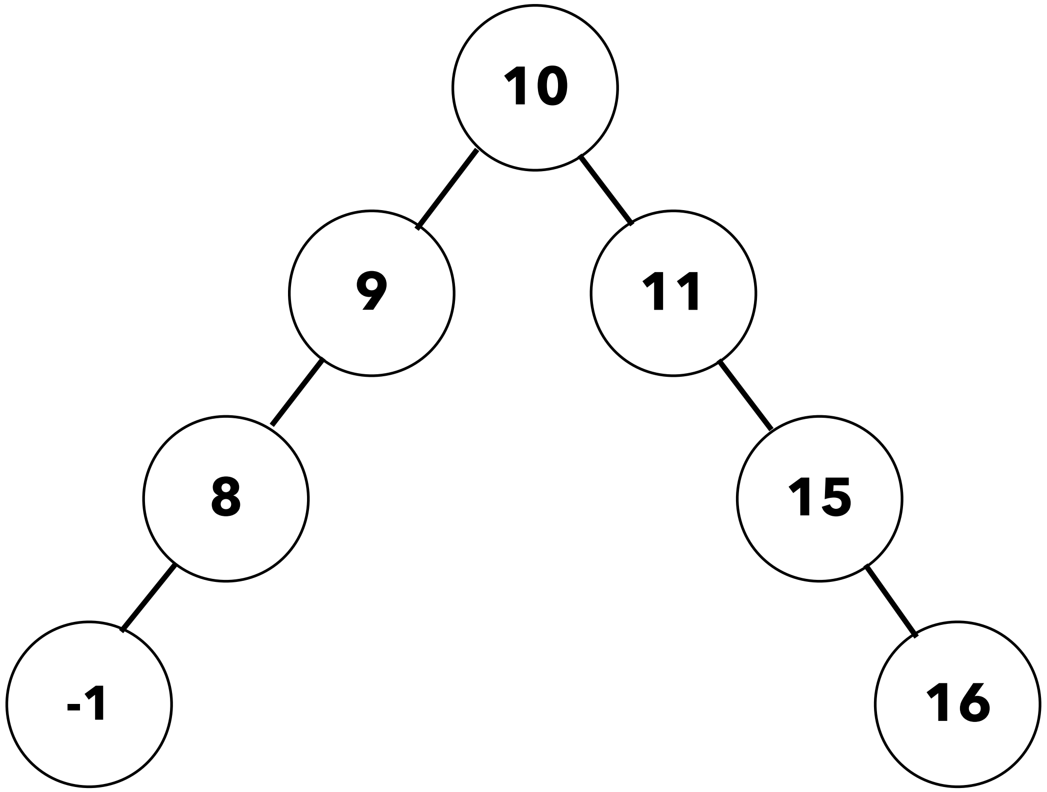




Tree Badness

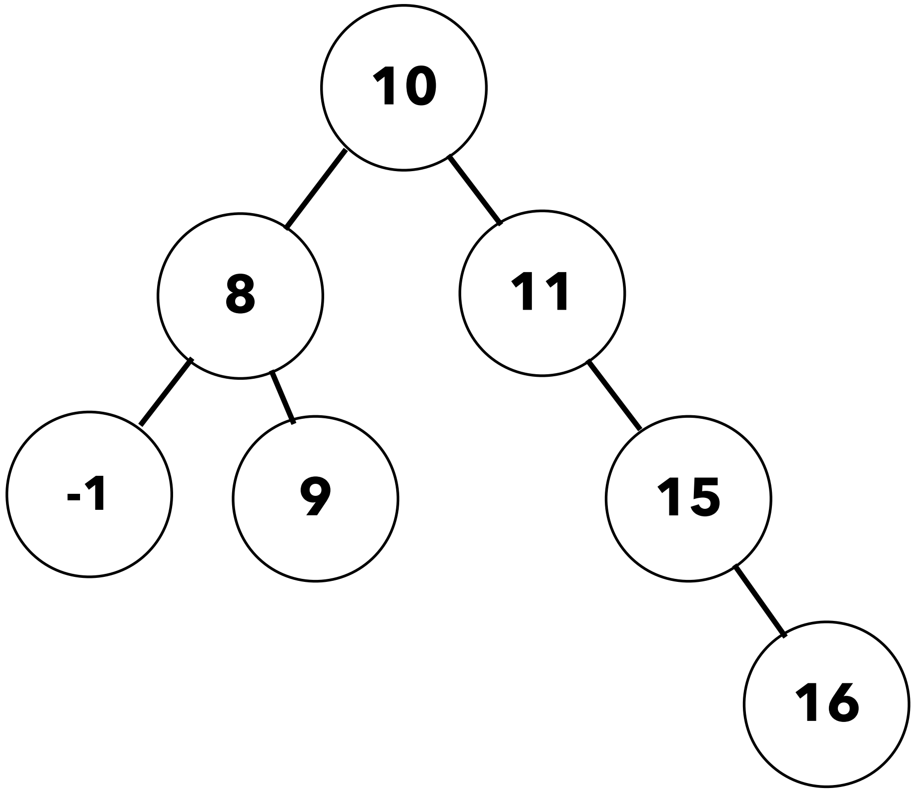
Hey Jude: can we take a bad tree and make it better?

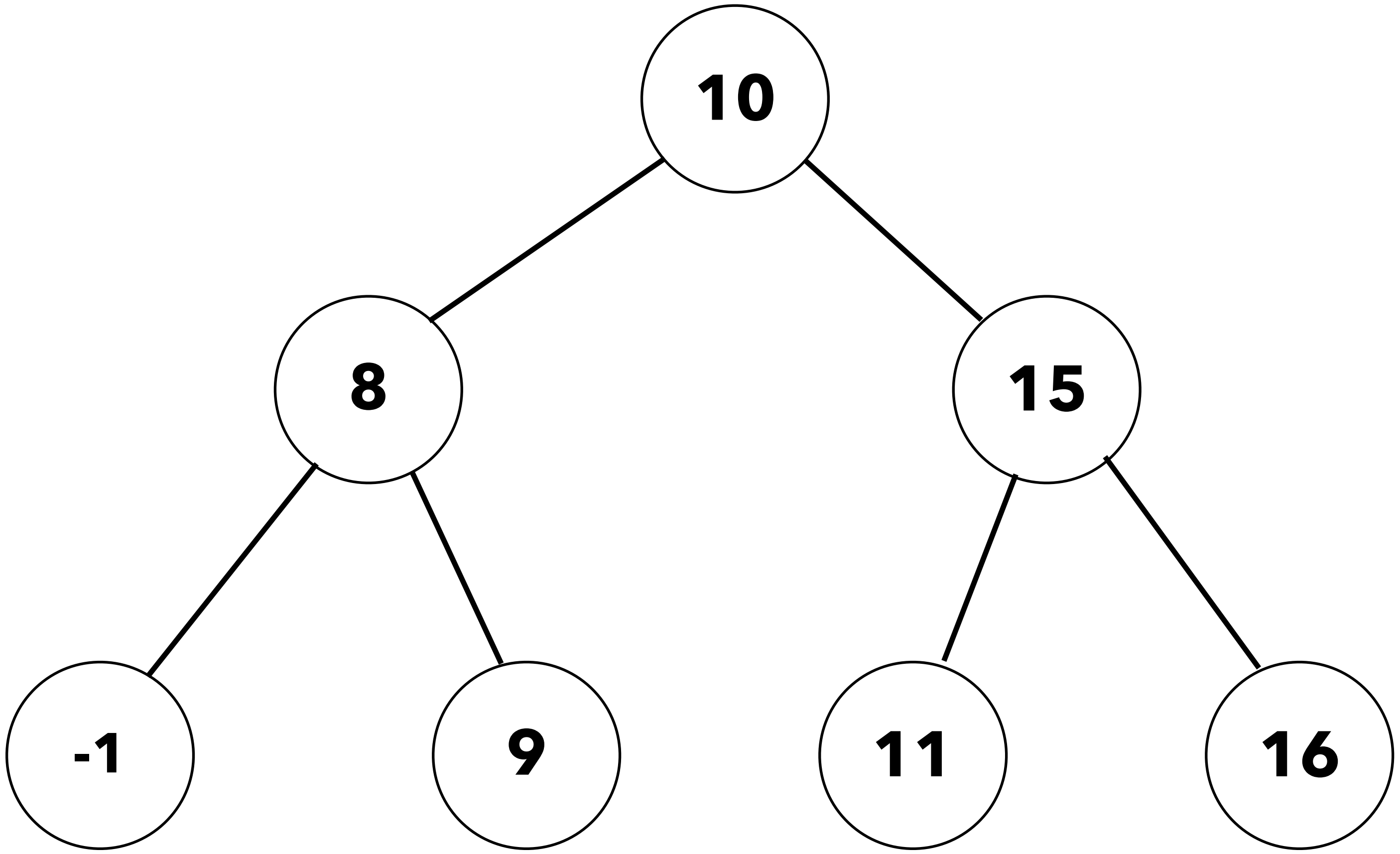




Tree Badness

Hey Jude: can we take a bad tree and make it better?

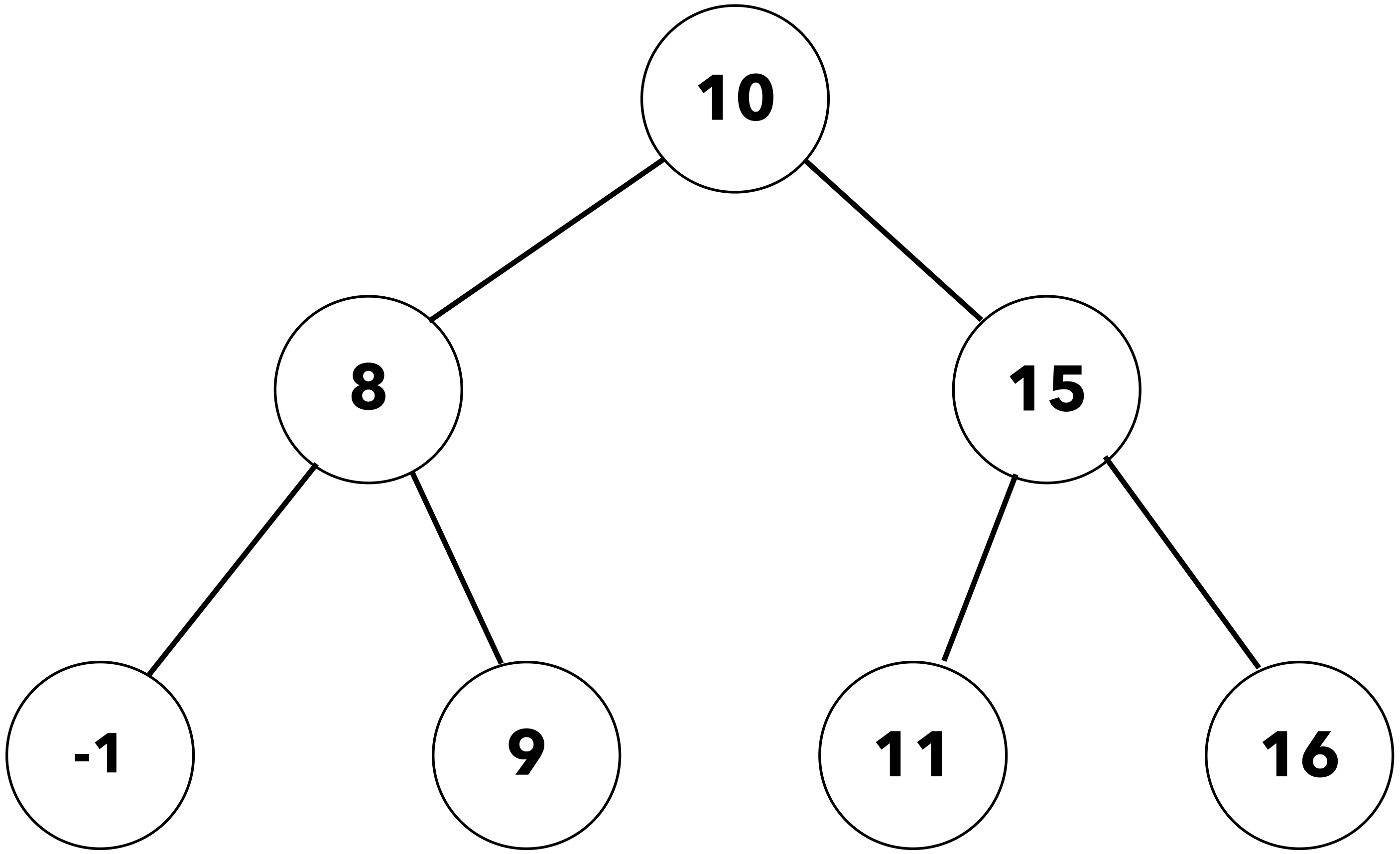


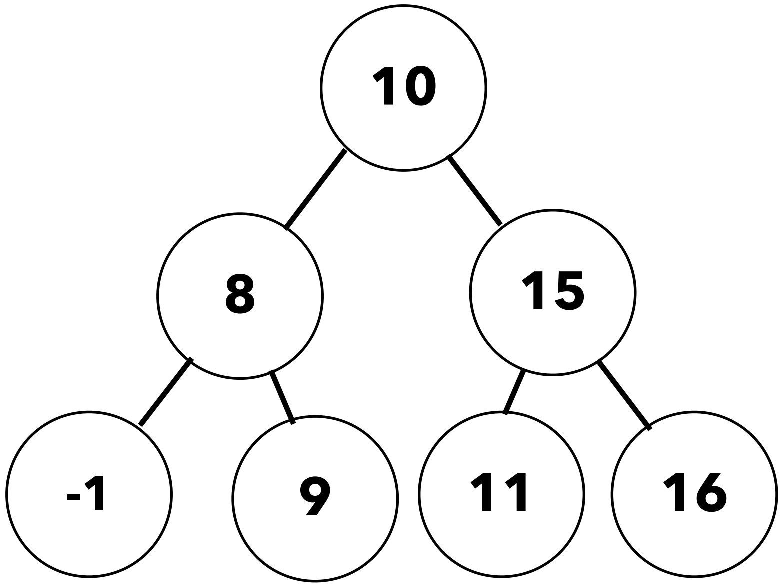


Tree Badness

Hey Jude: can we take a bad tree and make it better?

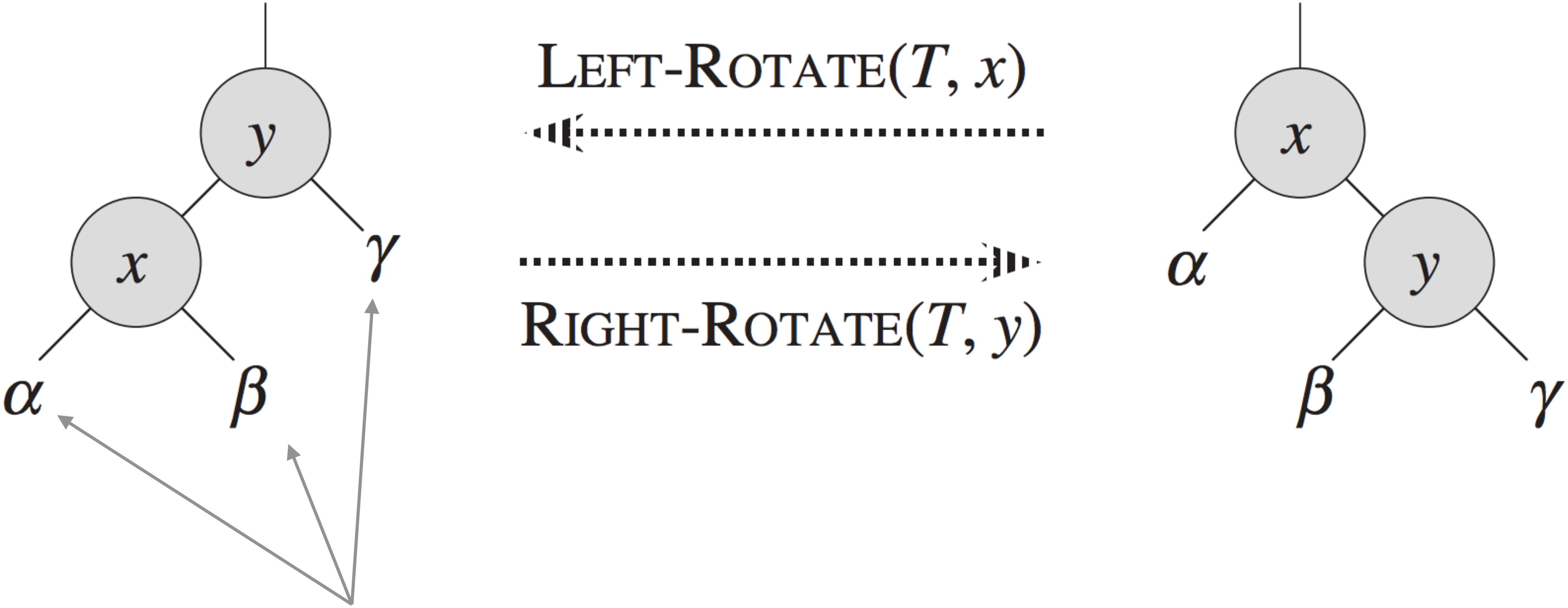
(yes!)





Tree Rotations

modify tree structure without violating the BST property.



subtrees (could be null, leaf, or tree with many nodes)

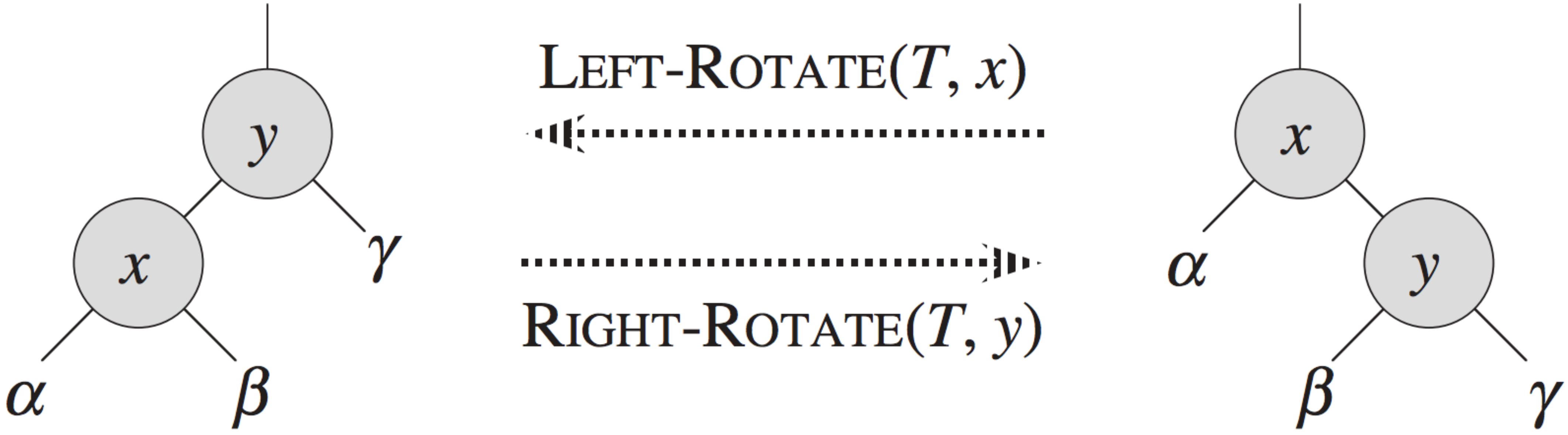
CLRS Fig 13.2, pg 313

Tree Rotations

modify tree 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

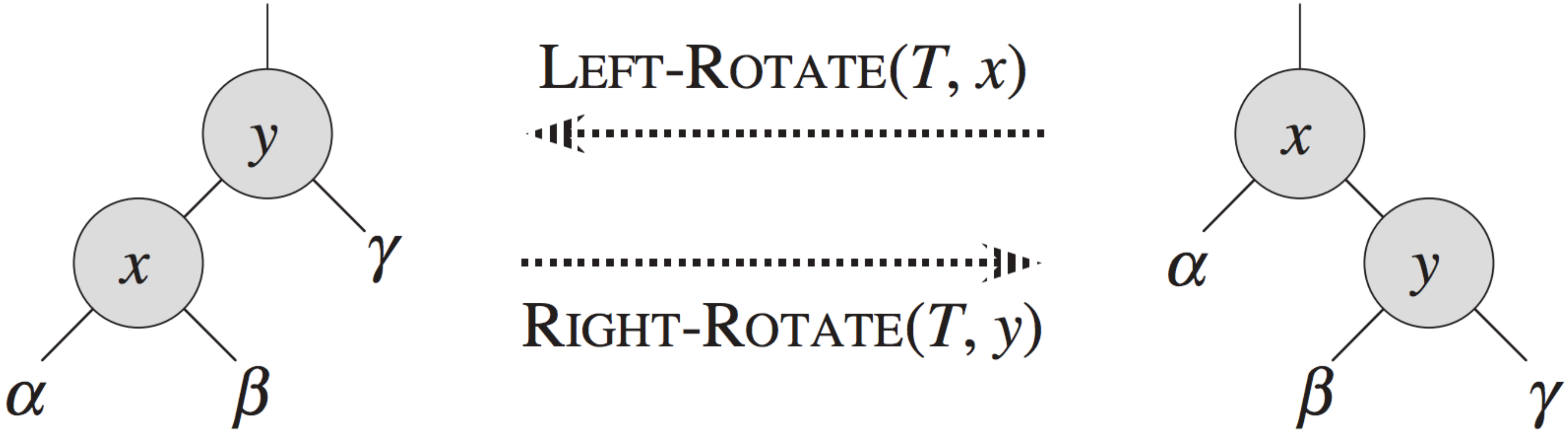
Tree Rotations

modify tree 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.

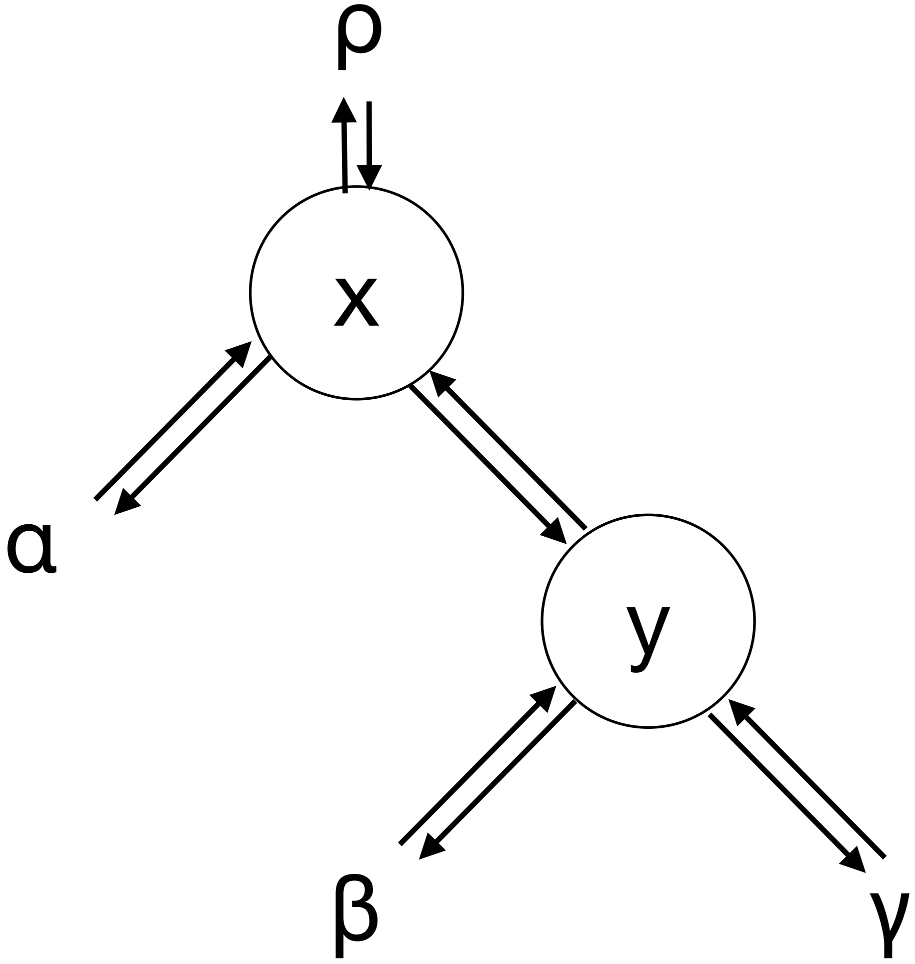


CLRS Fig 13.2, pg 313

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



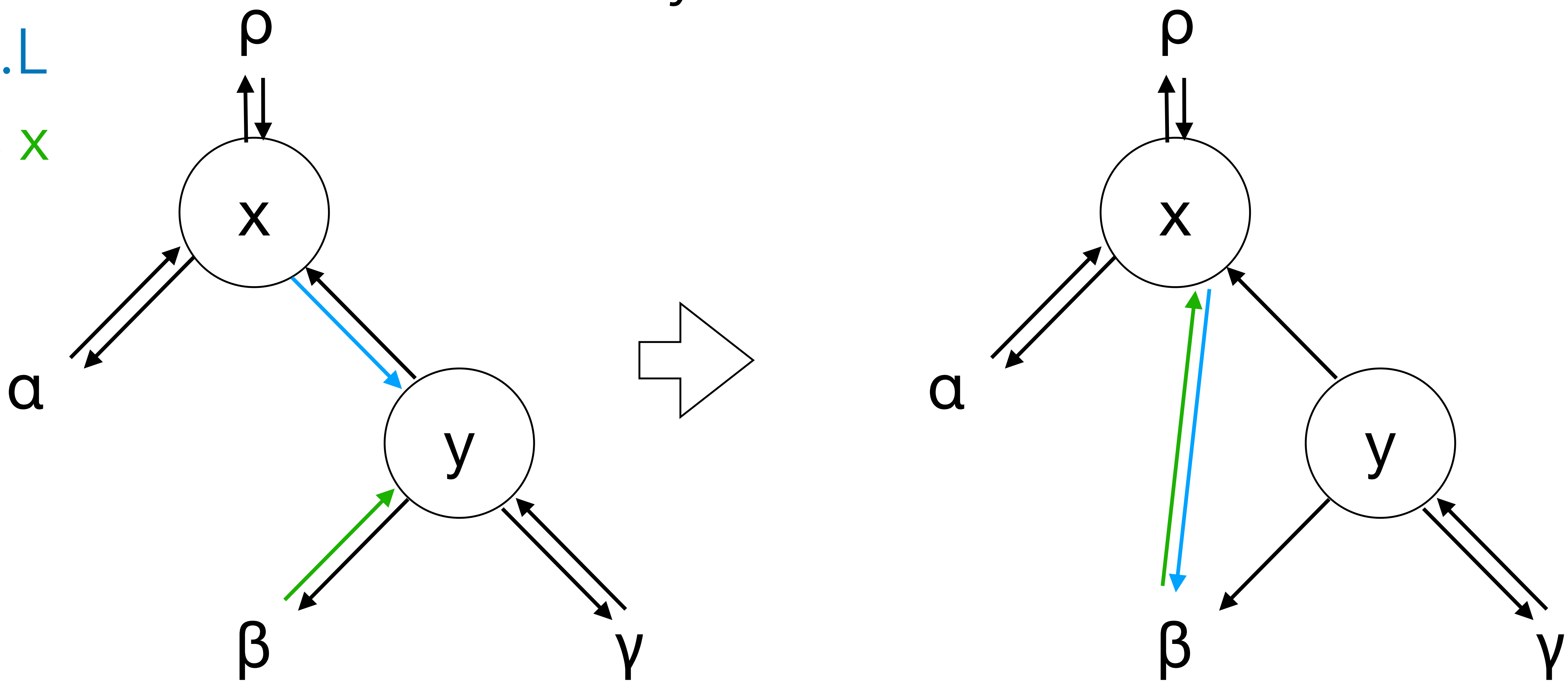
Tree Rotations

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



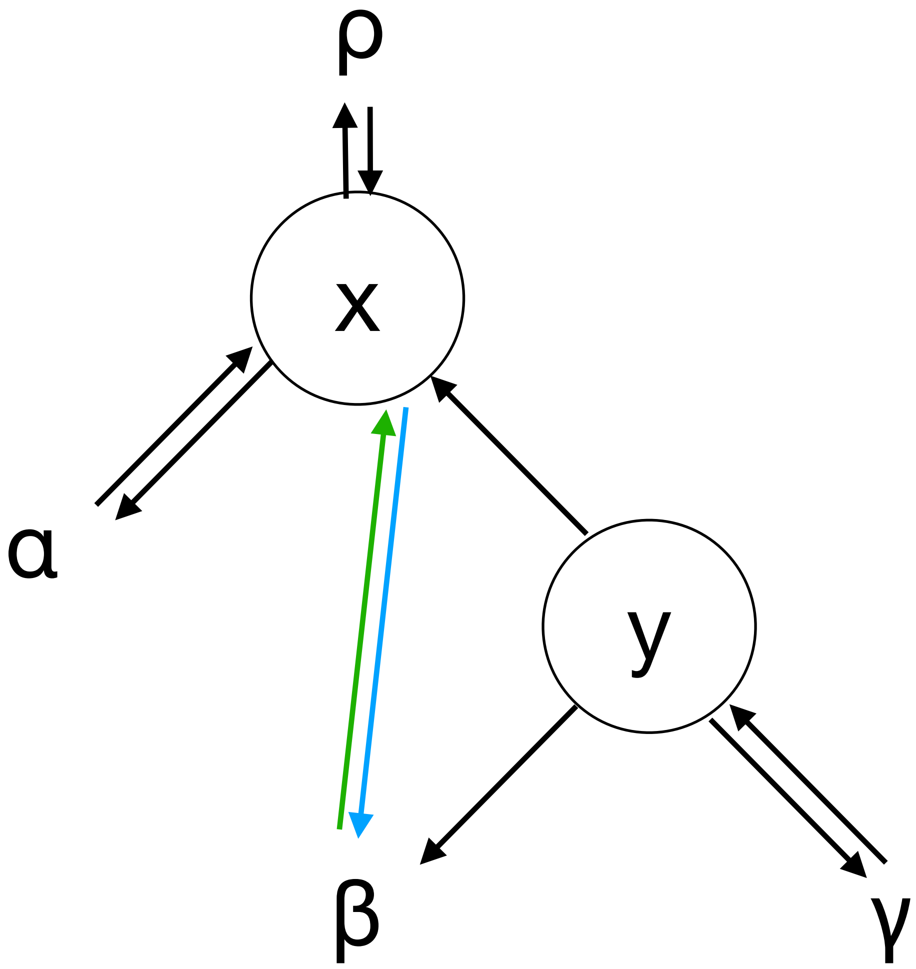
Tree Rotations

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



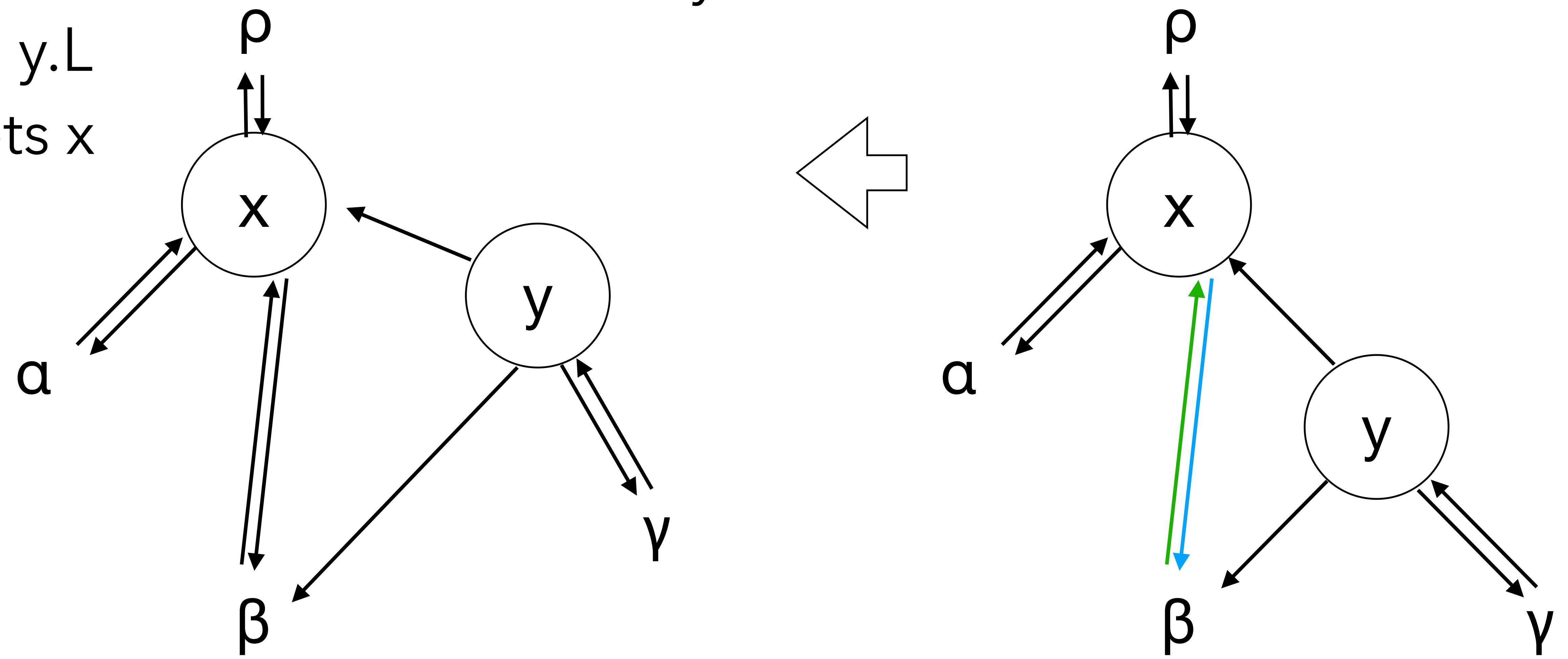
Tree Rotations

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)

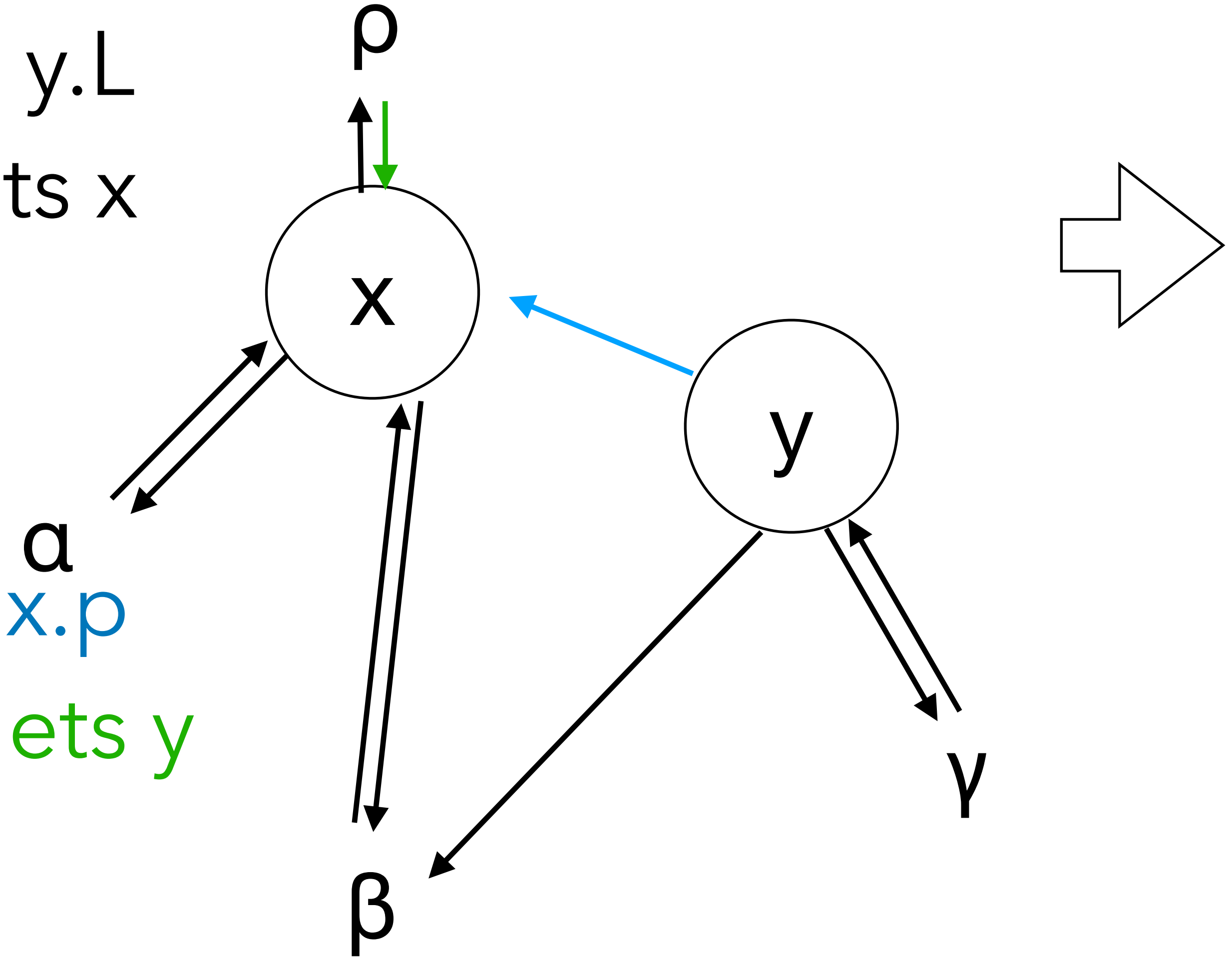
Tree Rotations

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

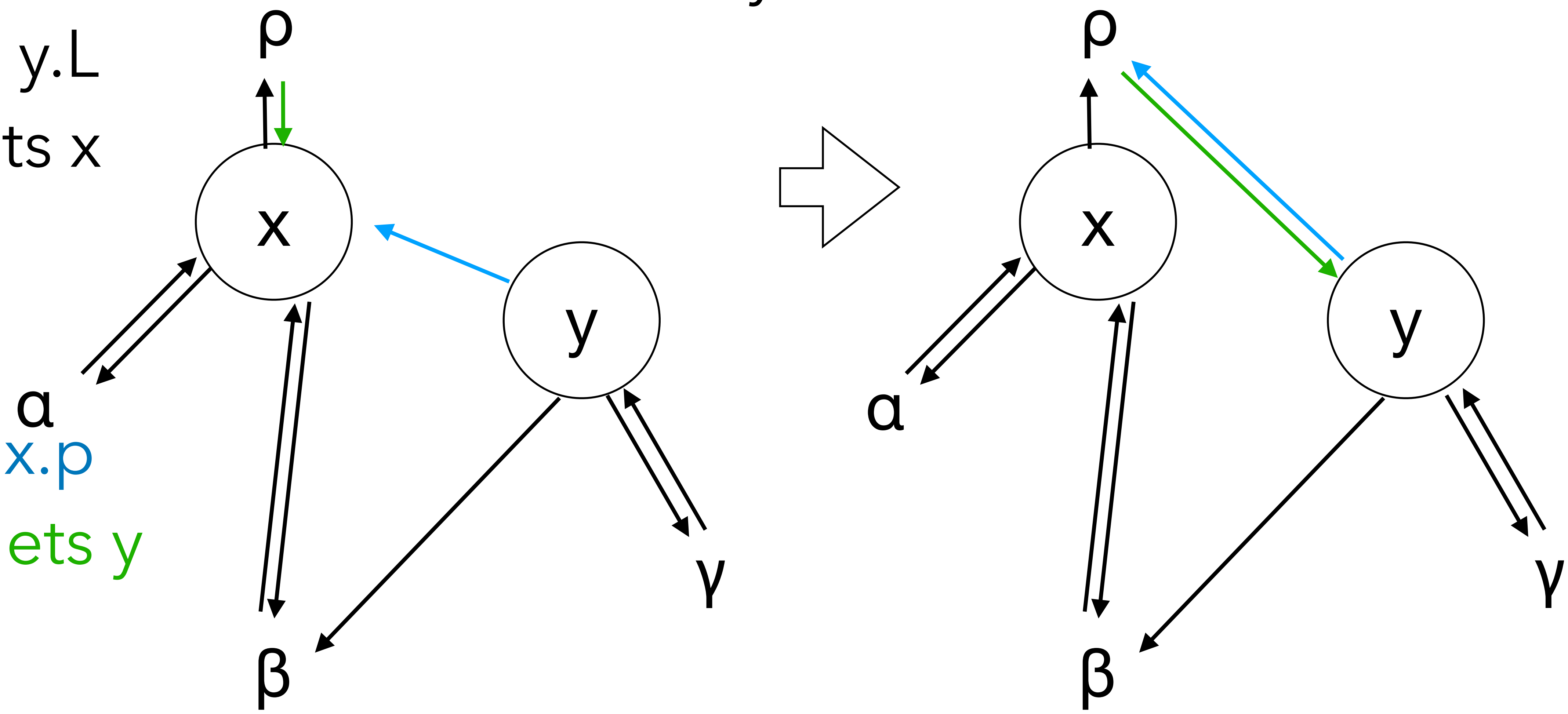
Tree Rotations

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

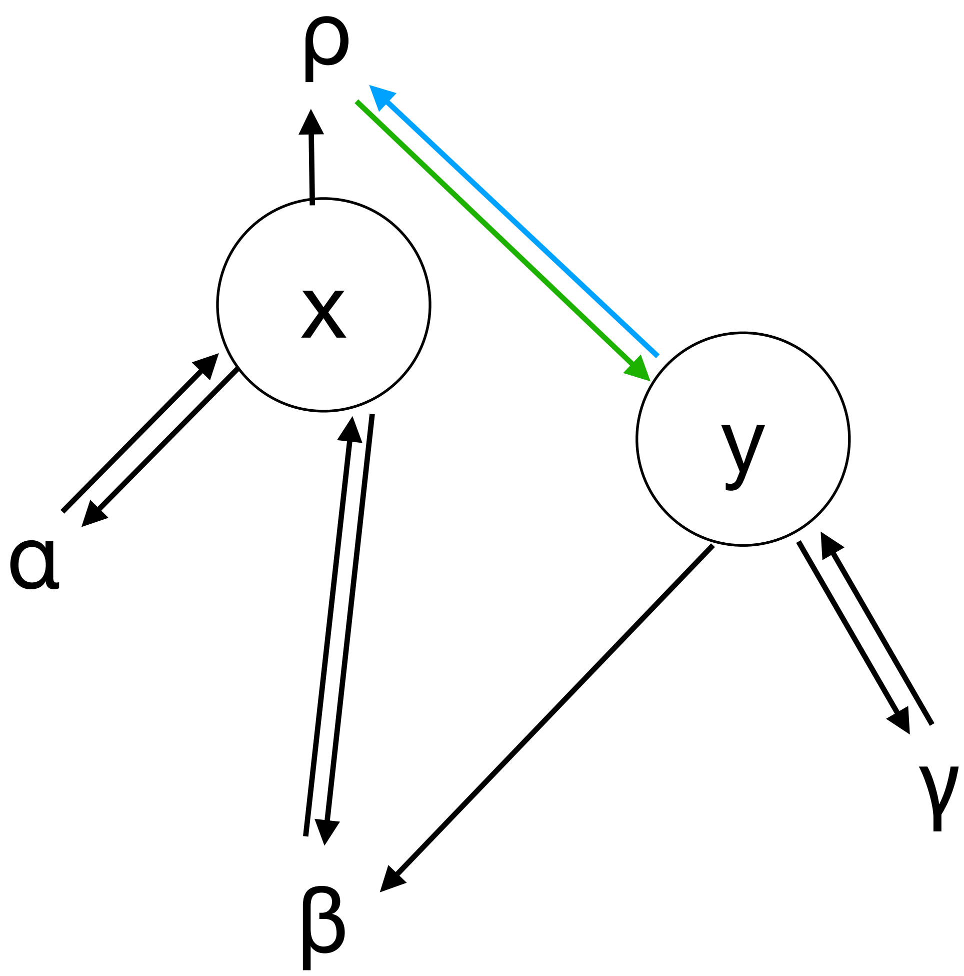
(what if ρ is null / x was root?)

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):

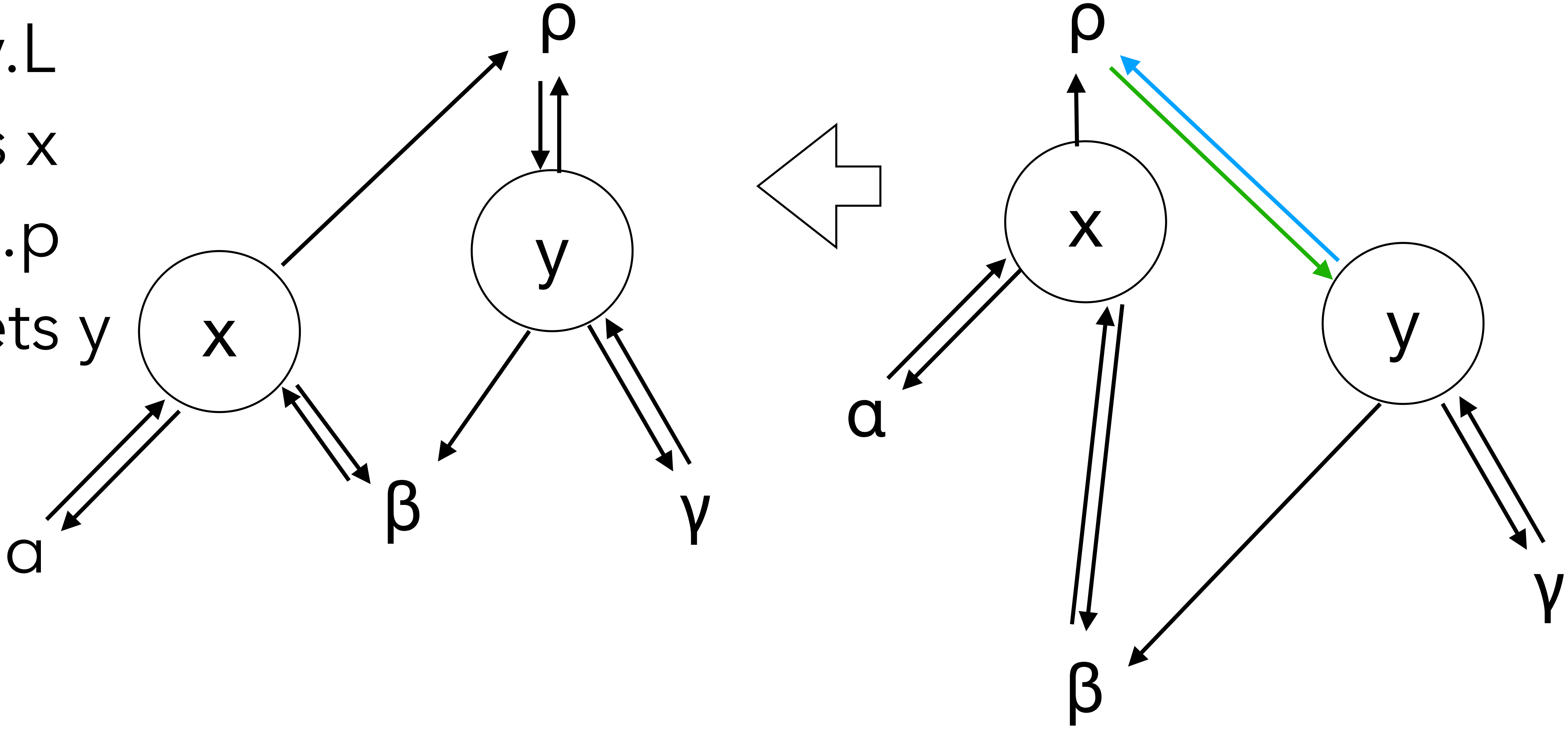
1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. Transfer x itself: x becomes y's left subtree

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):

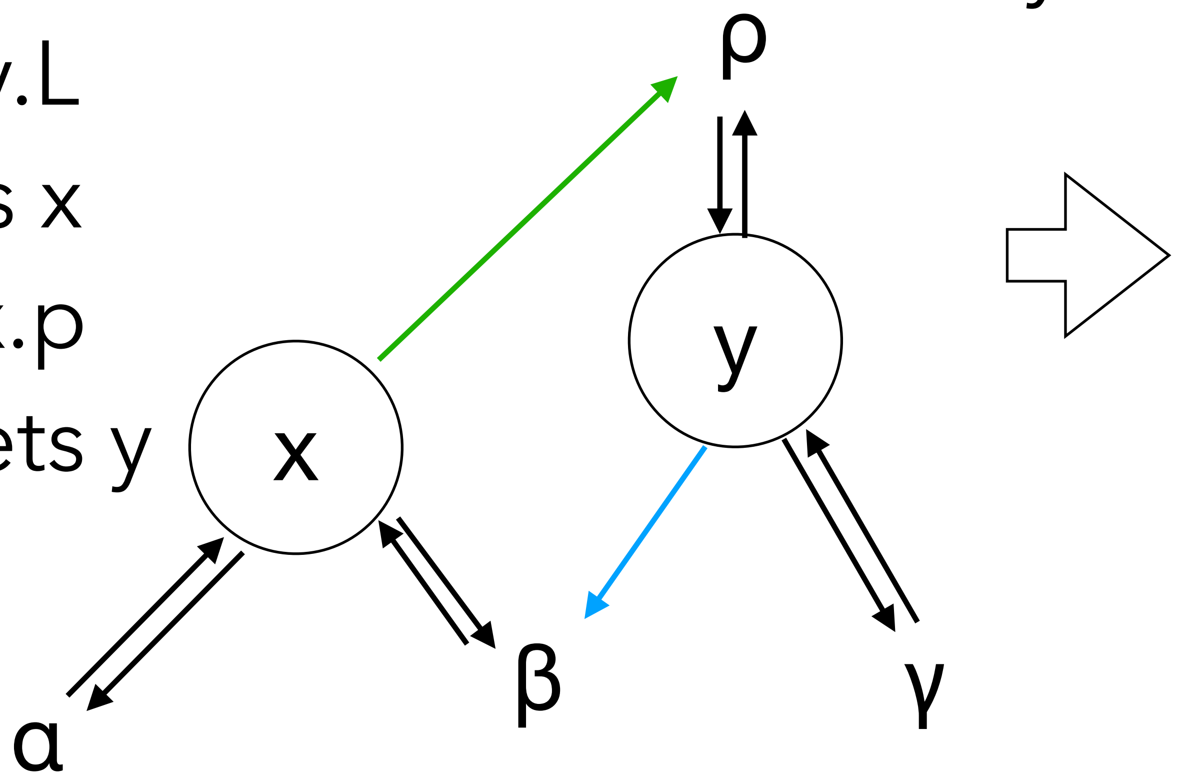
1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. **Transfer x itself:** x becomes y's left subtree

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):

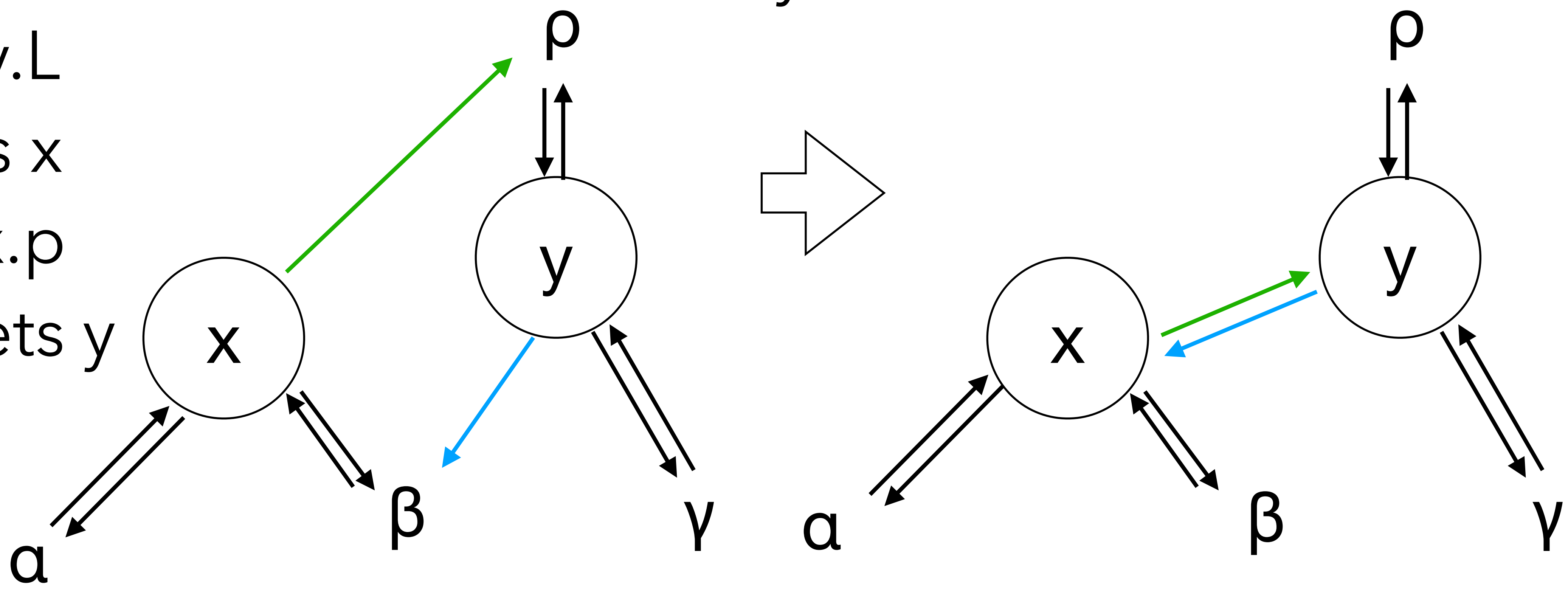
1. Transfer β: x's right subtree becomes y's old left subtree (β)
2. Transfer the parent: y's parent becomes x's old parent
3. **Transfer x itself:** x becomes y's left subtree

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):

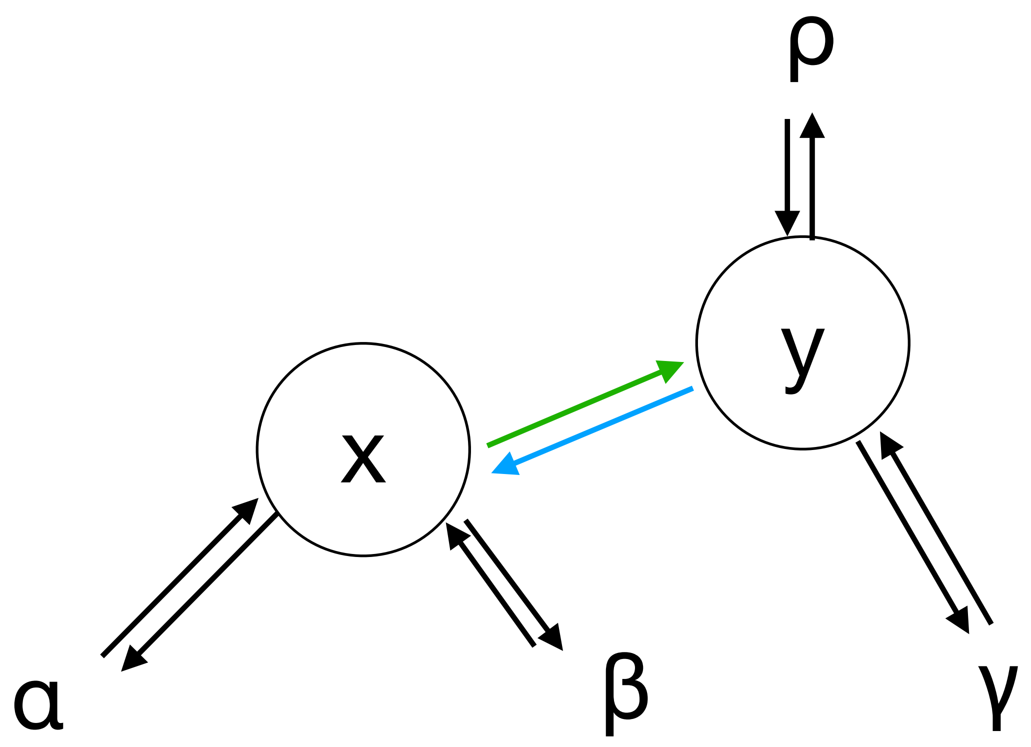
1. Transfer β: x’s right subtree becomes y’s old left subtree (β)
2. Transfer the parent: y’s parent becomes x’s old parent
3. **Transfer x itself:** x becomes y’s left subtree

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

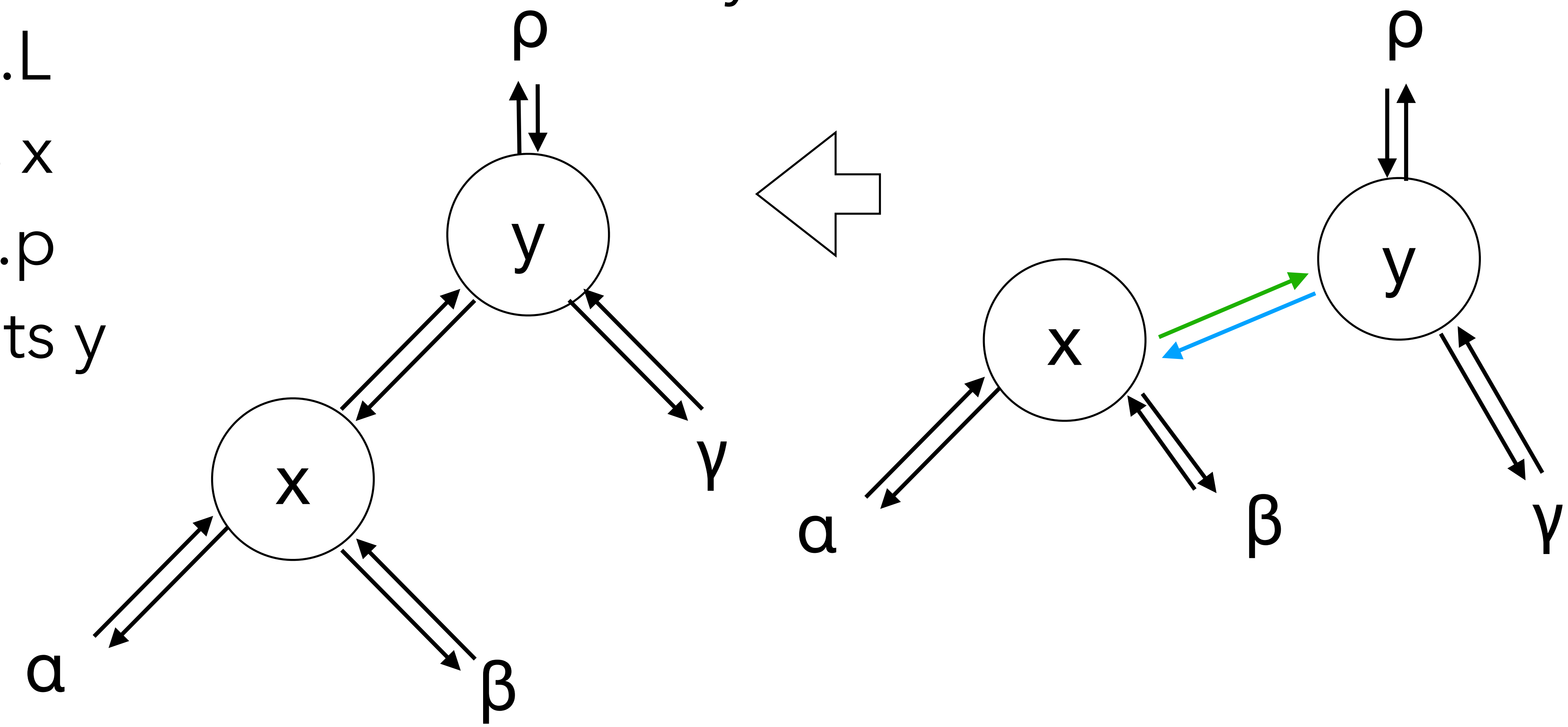
y.L.p gets x

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

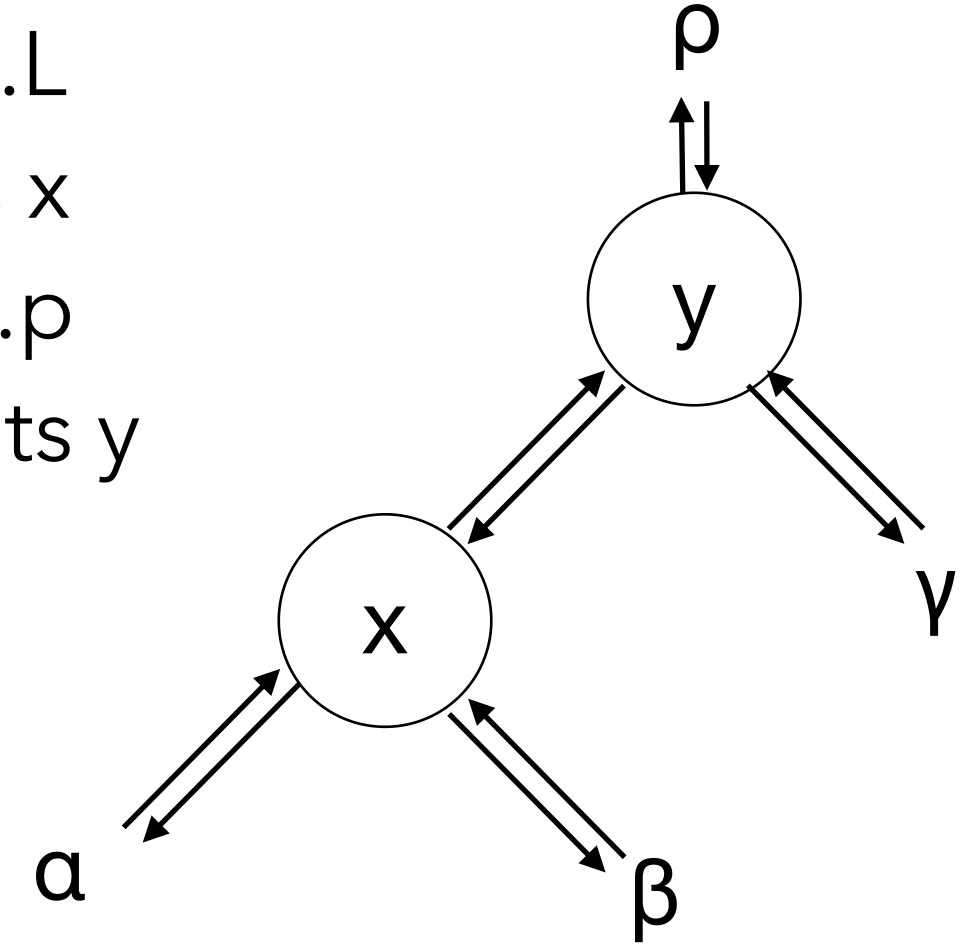
y.L.p gets x

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

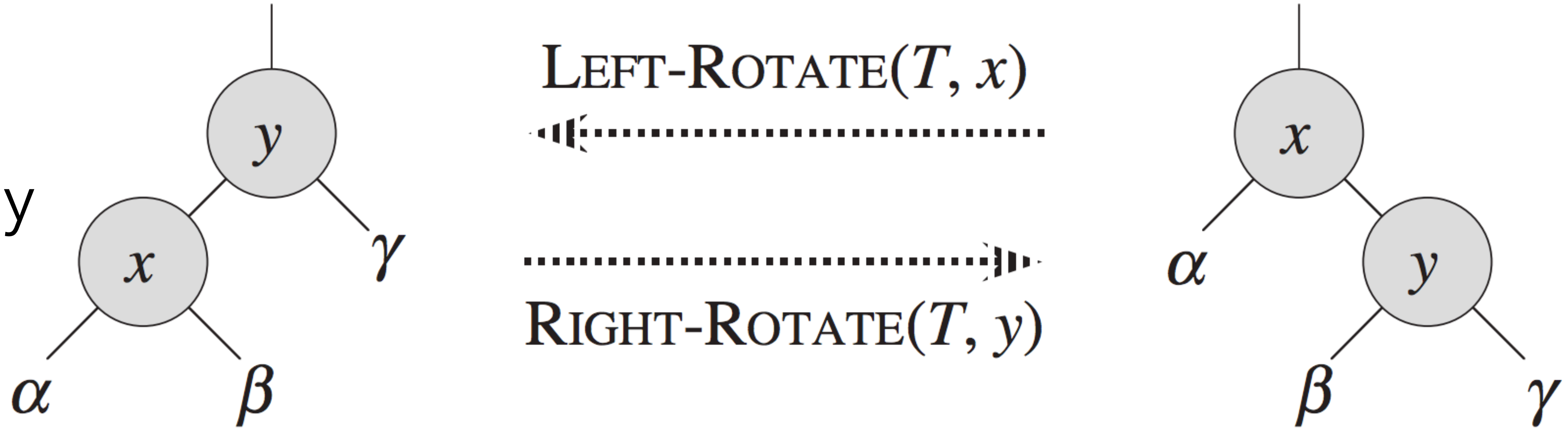
y.L.p gets x

y.p gets x.p

p.[L/R] gets y

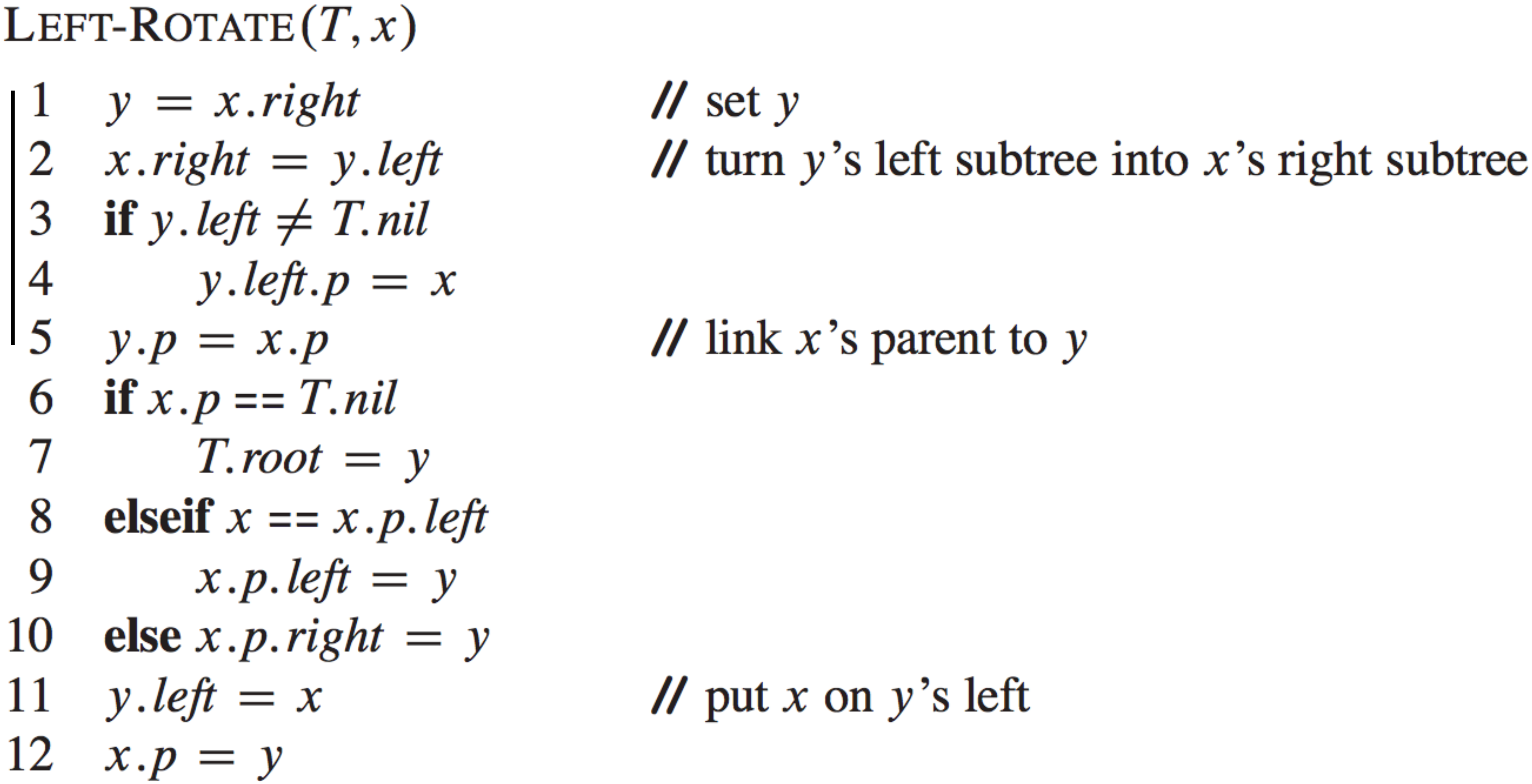
y.L gets x

x.p gets y



Overall Transformation

Pseudocode from CLRS



LEFT-ROTATE(T, x)

1. xfer β

```
y = x.right                      // set y
x.right = y.left                 // turn y's left subtree into x's right subtree
if y.left ≠ T.nil
    y.left.p = x
y.p = x.p                        // link x's parent to y
```

2. xfer parent

```
if x.p == T.nil
    T.root = y
elseif x == x.p.left
    x.p.left = y
else x.p.right = y
```

3. xfer x

```
y.left = x                       // put x on y's left
x.p = y
```

Notational quirk: assume T.nil means "null"