Goals

Be able to execute BST rotations on paper.

Be prepared to implement BST rotations.
Measuring Badness

Bad tree =(

-1
8
9
10
11
15
16

Good tree =)

8
-1
9
11
15
16

how bad? how good?
Measuring Badness

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

**LEFT-ROTATE**(T, x)

**RIGHT-ROTATE**(T, y)

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
Tree Rotations

modify tree structure without violating the BST property.

Steps in left rotation (move y up to its parent’s position):
1. Transfer $\beta$: x’s right subtree becomes y’s old left subtree ($\beta$)
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.
Tree Rotations

Steps in left rotation (move y up to its parent’s position):
1. Transfer $\beta$: x’s right subtree becomes y’s old left subtree ($\beta$)
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 $\beta$:** x’s right subtree becomes y’s old left subtree ($\beta$).
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 $\beta$: x’s right subtree becomes y’s old left subtree ($\beta$)
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 $\beta$: x’s right subtree becomes y’s old left subtree ($\beta$)
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 $\beta$: x’s right subtree becomes y’s old left subtree ($\beta$)
2. **Transfer the parent**: y’s parent becomes x’s old parent
3. Transfer x itself: x becomes y’s left subtree

\[ x.R \text{ gets } y.L \]
\[ y.L.p \text{ gets } x \]

\[ y.p \text{ gets } x.p \]
\[ p.[L/R] \text{ gets } y \]

(what if $\rho$ is null / x was root?)
Tree Rotations

Steps in left rotation (move y up to its parent’s position):
1. Transfer $\beta$: x’s right subtree becomes y’s old left subtree ($\beta$)
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 $\rho$ is null / x was root?)
Tree Rotations

Steps in left rotation (move y up to its parent’s position):
1. Transfer $\beta$: $x$’s right subtree becomes $y$’s old left subtree ($\beta$)
2. Transfer the parent: $y$’s parent becomes $x$’s old parent
3. Transfer $x$ itself: $x$ becomes $y$’s left subtree

$x.R \text{ gets } y.L$
$y.L.p \text{ gets } x$
$y.p \text{ gets } x.p$
$p.[L/R] \text{ gets } y$

(only rearranged the picture)
Tree Rotations

Steps in left rotation (move y up to its parent’s position):
1. Transfer $\beta$: x’s right subtree becomes y’s old left subtree ($\beta$)
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 $\beta$: x’s right subtree becomes y’s old left subtree ($\beta$)
2. Transfer the parent: y’s parent becomes x’s old parent
3. **Transfer x itself**: x becomes y’s left subtree

\[
\begin{align*}
\text{x.R gets y.L} \\
y.L.p \text{ gets x} \\
y.p \text{ gets x.p} \\
p.[L/R] \text{ gets y}
\end{align*}
\]

\[
\begin{align*}
y.L \text{ gets x} \\
x.p \text{ gets y}
\end{align*}
\]
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 $\beta$: x’s right subtree becomes y’s old left subtree ($\beta$)
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 $\beta$: x’s right subtree becomes y’s old left subtree ($\beta$)
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. \( y = x.\text{right} \) // set \( y \)
2. \( x.\text{right} = y.\text{left} \) // turn \( y \)'s left subtree into \( x \)'s right subtree
3. if \( y.\text{left} \neq T.\text{nil} \)
4. \( y.\text{left}.p = x \)
5. \( y.p = x.p \) // link \( x \)'s parent to \( y \)
6. if \( x.p = T.\text{nil} \)
7. \( T.\text{root} = y \)
8. elseif \( x = x.p.\text{left} \)
9. \( x.p.\text{left} = y \)
10. else \( x.p.\text{right} = y \)
11. \( y.\text{left} = x \) // put \( x \) on \( y \)'s left
12. \( x.p = y \)

Notational quirk: assume \( T.\text{nil} \) means "null"