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)
Balance(n): \texttt{height(n.right)} - \texttt{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 $\beta$: 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

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 $\beta$: 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 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 $\beta$: 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

---

$\alpha$

$\beta$

$\rho$

(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 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 $\beta$: 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

(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

- 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
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 $\beta$: 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

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

LEFT-ROTATE(T, x)

RIGHT-ROTATE(T, y)
Pseudocode from CLRS

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

1. \(y = x.\text{right}\)  \hspace{1cm} // set \(y\)
2. \(x.\text{right} = y.\text{left}\) \hspace{1cm} // 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. \(y.p = x.p\) \hspace{1cm} // link \(x\)'s parent to \(y\)
4. **if** \(x.p = T.\text{nil}\)  
   5. \(T.\text{root} = y\)
6. **elseif** \(x = x.p.\text{left}\)  
   7. \(x.p.\text{left} = y\)
7. **else** \(x.p.\text{right} = y\)  
8. \(y.\text{left} = x\) \hspace{1cm} // put \(x\) on \(y\)'s left
9. \(x.p = y\)

Notational quirk: assume \(T.\text{nil}\) means “null”
Tree Rotations

Steps in **left** rotation (move y up to x’s position):
1. Transfer $\beta$
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

**Right rotation**

**Left rotation**