Announcements

- MidTerm Exam 1: October 16 in class
- MidTerm Exam 2: Last day of class
- Final: NO FINAL EXAM

BST: Insert

TREEINSERT(*tree* T, *node* z)

$\triangleright$ Insert node z in tree T

$y \leftarrow$ NIL
$x \leftarrow root[T]$
while $(x \neq \text{NIL})$
    do $y \leftarrow x$
        if $(key[z] < key[x])$
            then $x \leftarrow left[x]$
            else $x \leftarrow right[x]$
$p[z] \leftarrow y$
if $(y = \text{NIL})$
    then $root[T] \leftarrow z$
    else **if** $(key[z] < key[y])$
            then $left[y] \leftarrow z$
            else $right[y] \leftarrow z$

Time Complexity: O(h)
h = height of binary search tree

Search for x in T

Insert x as leaf in T

BST: Delete

Time Complexity: O(h)

h = height of binary search tree

```
TreeDelete(tree T, node z)
    ▷ Delete node z from tree T
if ((left[z] = NIL) or (right[z] = NIL))
    then y ← z
    else y ← Tree-Successor(z)
if (left[y] ≠ NIL)
    then x ← left[y]
    else x ← right[y]
if (x ≠ NIL)
    then p[x] ← p[y]
if (p[y] = NIL)
    then root[T] ← x
    else if (y = left[p[y]])
             then left[p[y]] ← x
             else right[p[y]] ← x
if (y ≠ z)
    then key[z] ← key[y]
         cop y's satellite data into z
return y
```

Set y as the node to be deleted. It has at most one child, and let that child be node x

If y has one child, then y is deleted and the parent pointer of x is fixed.

The child pointers of the parent of x is fixed.

The contents of node z are fixed.

Animations

- **BST:**
  http://babbage.clarku.edu/~achou/cs160/examples/bst_animation/BST-Example.html
- **Rotations:**
  http://babbage.clarku.edu/~achou/cs160/examples/bst_animation/index2.html
- **RB-Trees:**
  http://babbage.clarku.edu/~achou/cs160/examples/bst_animation/RedBlackTree-Example.html

Red-Black (RB) Trees

- Every node in a red-black tree is colored either red or black.
  - The root is always black.
  - Every path on the tree, from the root down to the leaf, has the same number of black nodes.
  - No red node has a red child.
  - Every NIL pointer points to a special node called NIL[T] and is colored black.
- Every RB-Tree with n nodes has **black height** at most $\log n$
- Every RB-Tree with n nodes has **height** at most $2\log n$

Red-Black Tree Insert

```
RB-Insert (T,z)                // pg 280
      // Insert node z in tree T
      y = NIL[T]
      x = root[T]
      while (x ≠ NIL[T]) do
                y = x
                if (key[z] < key[x])
                                x = left[x]
                                x = right[x]
      p[z] = y
      if (y == NIL[T])
                root[T] = z
      else if (key[z] < key[y])
                left[y] = z
      else right[y] = z
      // new stuff
      left[z] = NIL[T]
      right[z] = NIL[T]
      color[z] = RED
      RB-Insert-Fixup (T,z)
```

```
RB-Insert-Fixup (T,z)
      while (color[p[z]] == RED) do
                if (p[z] = left[p[p[z]]]) then
                     y = right[p[p[z]]]
                     if (color[y] == RED) then                    // C-1
                                color[p[z]] = BLACK
                                color[y] = BLACK
                                z = p[p[z]]
                                color[z] = RED
                     else       if (z == right[p[z]]) then // C-2
                                     z = p[z]
                                     LeftRotate(T,z)
                                color[p[z]] = BLACK       // C-3
                                color[p[p[z]]] = RED
                                RightRotate(T,p[p[z]])
                else
                     // Symmetric code: "right" ↔ "left"
                     . . .
      color[root[T]] = BLACK
```

Rotations

```
LeftRotate(T,x)    // pg 278
    // right child of x becomes x's parent.
    // Subtrees need to be readjusted.
    y = right[x]
    right[x] = left[y]          // y's left subtree becomes x's right
    p[left[y]] = x
    p[y] = p[x]
    if (p[x] == NIL[T]) then
            root[T] = y
    else if (x == left[p[x]]) then
            left[p[x]] = y
    else right[p[x]] = y
    left[y] = x
    p[x] = y
```