Announcements

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

Data Structure Evolution

- Standard operations on data structures
  - **Search**
  - **Insert**
  - **Delete**
- Linear Lists
  - Implementation: **Arrays** (**Unsorted and Sorted**)
- **Dynamic** Linear Lists
  - Implementation: **Linked Lists**
- **Dynamic** Trees
  - Implementation: **Binary Search Trees**

BST: Search

```
TreeSearch(node x, key k)
    ▷ Search for key k in subtree rooted at node x
1   if ((x = NIL) or (k = key[x]))
2       then return x
3   if (k < key[x])
4       then return TreeSearch(left[x], k)
5       else  return TreeSearch(right[x], k)
```

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

Not O(log n) — Why?

BST: Insert

```
TreeInsert(tree T, node z)
    ▷ Insert node z in tree T
 1  y ← NIL
 2  x ← root[T]
 3  while (x ≠ NIL)
 4        do y ← x
 5           if (key[z] < key[x])
 6              then x ← left[x]
 7              else  x ← right[x]
 8  p[z] ← y
 9  if (y = NIL)
10     then root[T] ← z
11     else  if (key[z] < key[y])
12              then left[y] ← z
13              else  right[y] ← 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
 1  if ((left[z] = NIL) or (right[z] = NIL))
 2      then y ← z
 3      else y ← Tree-Successor(z)
 4  if (left[y] ≠ NIL)
 5      then x ← left[y]
 6      else x ← right[y]
 7  if (x ≠ NIL)
 8      then p[x] ← p[y]
 9  if (p[y] = NIL)
10      then root[T] ← x
11      else if (y = left[p[y]])
12               then left[p[y]] ← x
13               else right[p[y]] ← x
14  if (y ≠ z)
15      then key[z] ← key[y]
16           cop y's satellite data into z
17  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