October 21

Intro to trees (see slides)

(Still updating these notes)

Post-order

Pre-order

In-order

October 23

Housekeeping

- HW3 due Friday
- "Like radix sort" in the sense that we do multiple passes on each element of input. You can use the sorting method of your choice in each pass (don't have to use counting sort).
- Setting maximum late penalty to 5/10 points
  - Passing all tests, more than five days late -> 50% credit
  - Would have gotten 8/10, more than five days late -> 30% credit

Binary Search Trees

- Function: store elements in sorted order
- Form: use binary tree (since `compareTo` is a binary operator)
  - either empty; or
  - a root node R such that
    - every element of R's left subtree contains data "less than" R's data
    - every element of R's right subtree contains data "greater than" R's data
    - left and right subtrees are also binary search trees

Adding elements to BST

- Adding values to Binary Search Tree
  - Example on board: balanced tree

55, 29, 87, -3, 42, 87, 60, 91

Add 49

Modifying objects in-place

```
String s = "halloween";
s.toUpperCase();
```

```
System.out.println(s); // halloween
s = s.toUpperCase();
System.out.println(s); // HALLOWEEN
```

- General approach: `x = change(x);`
- To modify a tree, we should use functions that take as input, the old state of a node, and return the new state of the node.
- Then we can re-assign our objects, just as we did with linked lists and `next`:

```
root = change(root, parameters);
root.left = change(root.left, parameters);
root.right = change(root.right, parameters);
```

What does this have to do with adding elements to trees? In this case, a recursive algorithm will eventually find a `root` that is `null`: can't just update the fields in the class, have to return a new `Node`.

Removing elements from BST

Cases for the node to be removed:

- leaf node: replace with null
- node with left child only: replace with left child
- node with right child only: replace with right child
- node with both children: replace with min value from right (or max value from left)
  - recursively remove that min/max value from subtree

Return to example tree and remove 55

Tree Balance

Balanced tree: one whose subtrees differ in height by at most 1, and these subtrees are also balanced. What will be the height as function of N?

- Assume we have a complete tree.
- At depth k, we have 2^k nodes (can prove by induction)
- Total number of nodes is $N = 2^k + \cdots + 2^1 + 2^0$
- This is a finite geometric series:

$$a + ar + ar^2 + \cdots + ar^n = \sum_{i=0}^{n} ar^i \tag{1}$$

with sum known to be $a(1 - r^{n+1})/(1 - r)$. So, with $a = 1$ and $r = 2$ we find that $N = (1 - 2^{k+1})/(1 - 2) = 2^{k+1} - 1$. Solving for k, we find:

$k + 1 = \log_2(N + 1)$

$\rightarrow k = \log_2(N+1) - 1$

This is the exact relationship: for estimating algorithm runtime, it suffices to remember that $k \approx \mathcal{O}(\log_2(N))$.