CS 120: Intro to Computer Programming II

In-Class Activity - 06 Trees - Day 2

Activity 1 - Turn in this one

Choose a selection of 10 integers, all different. Mix them up, so that they aren't all in order. Then (working as a group!) insert each one of the integers, one at a time, into a BST. (The BST should start empty, and end up with 10 nodes.) Remember: in this class, we'll never move any nodes around after we've inserted them; we can only **add new leaves** at every step.

Make sure to keep everybody involved. Why not take turns - have each member of the group insert a different value into the tree?

Turn in the sequence of integers you chose (the order is important), along with the picture your group generated.

Now, take the **same numbers,** but shuffle them up. Then, starting again with an empty tree, insert them again.

Turn in this second sequence of numbers, too - along with the tree that the group drew.

What does the group notice about the two trees? Are they the same tree? Do they have the same root, or the same internal shape? Are they balanced or not?

Activity 2 - Turn in this one

Here's another one which can **involve the whole group.** Take the **same set of numbers**, but everybody shuffles them again - in secret. Don't tell anybody else what your order was (at first). Then, draw the tree again. When everybody is done, take turns screen sharing - show the group the sequence of values that you selected, and also the tree you came up with.

Turn in at least 3 more trees, along with the integer sequences that generated them.

Solution: Every group's trees will be different. However, what you should have seen, across the past two exercises, is that even with the same data values, you can end up with **very** different trees.

This is most obvious at the root: whatever you insert first will be the root. And of course, this partitions the input into two sets: that which is less than the root, and that which is more. So it can vastly change how the tree looks.

But it's not just at the root - at **every node,** changing the order in which values are inserted can change how that subtree is divided - and thus the shape of tree that results.

Activity 3 - Turn in this one

Discuss with the group: we inserted the same set of values into a BST many times. Each time, it was possible to build a legal BST, and yet there were **many** different BSTs! How is this possible?

Next, let's consider the height of the trees. I'm sure that you had one or two jokers in the group, who inserted the values in sorted order (or maybe reversed). What happened when they did that?

But more importantly, what happened when you inserted shuffled values into the tree? How high was the tree, roughly?

Solution: The "worst" BST you can build is one that looks like a linked list; this happens when you insert the values in order (or in reverse order, or a few other, more complex variations).

The "best" BST you can build is one that is as short as possible, which means making sure that the nodes are all full (two children each), at every level, (except for a few leaves). In that case, you will find that the tree height is proportional to $\lg n$, where n is the number of nodes.

Thus, if you doubled the number of nodes, you would only **increase the height by one!!!**

(activity continues on next page)

Activity 4 - Optional

OPTIONAL. Complete this if you have time, and turn it in. If you don't have time, you may report to your TA that you ran out of time.

Have somebody load up this website in a screen share:
`https://www.cs.usfca.edu/~galles/visualization/BST.html` Now, have somebody else run the following Python program, which will generate integers (in the range 0-1000), forever:

```
import random
while True:
    input()
    print(random.randint(0,1000)
```

Have the random number person read out numbers, and add them into the tree as you go (try to avoid duplicates, although it's not terrible if one or two of them sneak in).

Insert quite a few values into the tree (see if you can get 16, or even 32, of them before you run out of time). Measure the head of the tree. How tall is the tree, compared to the number of nodes it holds?

If we added **twice as many nodes,** how much would expect the tree to grow, **on average?**

Solution: See solutions above.

Challenge Activity - Do not turn in this one

Now, let's think a little more abstractly about how the order of values affects the shape of the tree. Take any one of the trees that your group has built - I don't care which, so long as it doesn't look like a linked list - and read out the numbers in **exactly** the following order:

- The root value
- Then, the child (or children) of the root, moving left to right
- Then, the third layer of values, moving left to right
- Continuing, layer by layer...

Now, start over with an empty BST, and insert the values in exactly the order we specified. Does something interesting happen? Can your group explain **why** it happens?

How many different sequences of the values can you find, which all generate the same tree?

Solution: Some easy ways to build a sequence of values that will replicate a BST:

- Breadth-First Search (which is what I described above)
- Recurse only into left children, adding each node as you go; when you can't add any more children that way, back up a little, go right once, and then go left again (forever)
- Same thing, only prefer to go right
- Iterate through the leaves. For each leaf, go up through its ancestors - to the root, if necessary - and add any values that are missing (starting near the root and working toward the leaf).
- Add the root to the sequence. Then randomly select, over and over, any value which is *not* in the sequence, but whose parent *is*. Repeat until all values are in the sequence.
- etc...