Computer Science 2530
April 8, 2020

Happy Wednesday, April 8.

Today we continue to look at binary trees.

Traversing trees

When you *traverse* a binary tree, you do something at each node. What you do depends on the job that you are doing.

Read page **39A** in the notes, on traversing trees. It should be simple and clear.

Do exercises 1-3 at the bottom of page **39A**. (You can try exercise 4, but it is more difficult than the others. It requires that you think carefully from an example.)

Destructive functions on trees

A destructive function modifies a tree. The simplest kind of destructive function just traverses a tree and does something to each item. For example, suppose doubleAll(t) is intended to double the item in each node of tree t. There are two cases.

1. An empty tree has no nodes. So doubleAll(t) has nothing to do when t is empty (NULL).

2. To double all of the items in a nonempty tree t, you need to double the item (t->item) in the root of t. Statement

   ```
   t->item = 2 * t->item;
   ```

   accomplishes that. Next, double all of the items in the left subtree of tree and double all of the items in the right subtree of t. Statements

   ```
   doubleAll(t->left);
   doubleAll(t->right);
   ```

   doubles all of the items in subtrees of t.

Putting those together gives the following definition of doubleAll(t).

```
void doubleAll(Node* t)
{
  if(t != NULL)
  {
    t->item = 2 * t->item;
    doubleAll(t->left);
    doubleAll(t->right);
  }
}
```

You should be able to see that doubleAll does a preorder traversal of t. In fact, any traversal order would work.

Modifying the pointers in a tree

Some destructive functions change the pointers t->left and t->right. Page **39B** describes function removeLeftmostNode(t) that removes the node reached by following the left pointers until a node is found whose left pointer is NULL.

Read page **39B** in the notes. Do all three exercises at the bottom of the page. To do exercise 2, draw a small example tree and do a careful hand simulation. Exercise 3 is a tree traversal. You need to choose a traversal order so that you don't delete a node before you are finished with it. What traversal order should you choose?