```
- Notes on 03-27-18
- References: Hymnal 88
```

Threaded Trees

How to build the tree initially. Build it one node at a time. (start at the left)

Insert into Threaded Binary Tree (in order)

This algorithm attaches a node pointed to by Q as the right subtree of the node pointed to by P if the right subtree is empty... It inserts as a new leaf or inserts existing.

Insert into Right Leaf

```
Q.Rlink := P.Rlink; -- Insrt into Right leaf
Q.Rtag := P.Rtag;
P.Rlink := Q;
P.Rtag := "+";
Q.Llink := P;
Q.Ltag := “-”;

if Q.Rtag = “+” then
    Q$.Llink := Q; --Q$ == unordered successor.
end if;
```

To insert to the left, Exchange the words "left" and "right and replace 'Q$' with '$Q'

Checking Equivalence

Walk through. There is a difference between equivalent and similar. I believe Similar is: "if all nodes/leaves are in the same place" Equivalence is: "if all the nodes are in the same place and if the nodes' data-type is the same"

Copy a Binary Tree

AVAIL <- next node

/*Look at the stack/table on page ~91 */

Binary Search Tree

Alphabetic Search Tree

/*Pg. ~93, Iterative Algorithm to insert node in search tree. */

Burris likes Iterative over Recursion for most insertion/search etc.

You can use this same code for a search as well.

Binary Search Tree Deletion ~ pg 94

to get in-order successor, take one step to the right, then follow left until the end. that one is moved to the deleted one's spot. (you can choose in-order predecessor as well) but his algorithm is the in-order successor.