Yu, Ching

CSC 205 Lab 13 : Binary Search Trees

Goals

After completing this lab, you should be able to:

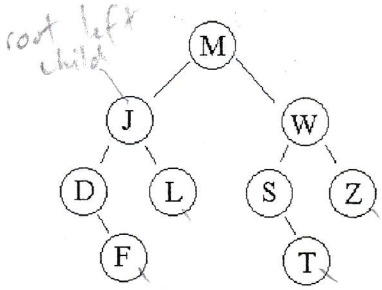
- Describe and use different binary tree properties such as height, full, complete, and balanced.
- Produce preorder, inorder, and postorder traversals of a binary search tree.
- Understand how to implement a binary search tree using a reference based implementation.
- Be able to write class methods that use the Binary Search Tree ADT, and instance methods that involve the private attributes.

Lab Startup

Change into your `Labs` directory, and let's create and change into a `Lab13` directory. Now, let's copy over some files by typing : `cp /pub/digh/CSC205/Lab13/* .`

Binary Tree Properties

Consider the following binary search tree. Answer each of the questions which follow.



$2^{(h+1)} - 1$

max size of binary tree

- How many leaves does this tree have? 4
- How many nodes are in the right subtree of the root's left child? 1 L
- What is the height of this tree? 3
- What is the maximum number of nodes this tree can have at this height? 15
- Is the tree full? Explain. No
- Is the tree complete? Explain. No because there are gaps
- Is the tree balanced? Explain. Yes because the height differs by 1

Building A Binary Search Tree

Create a program `MyTree` that declares an object `t` of type `BinarySearchTree`. Add the lines of code to your client file that would be needed to allow the root of your object T to point to the tree above. You will need to insert each letter one by one as a `new KeyedItem( )`. For example, to add the root node you would use the line which follows.

```
t.insert(new KeyedItem("M"));
```