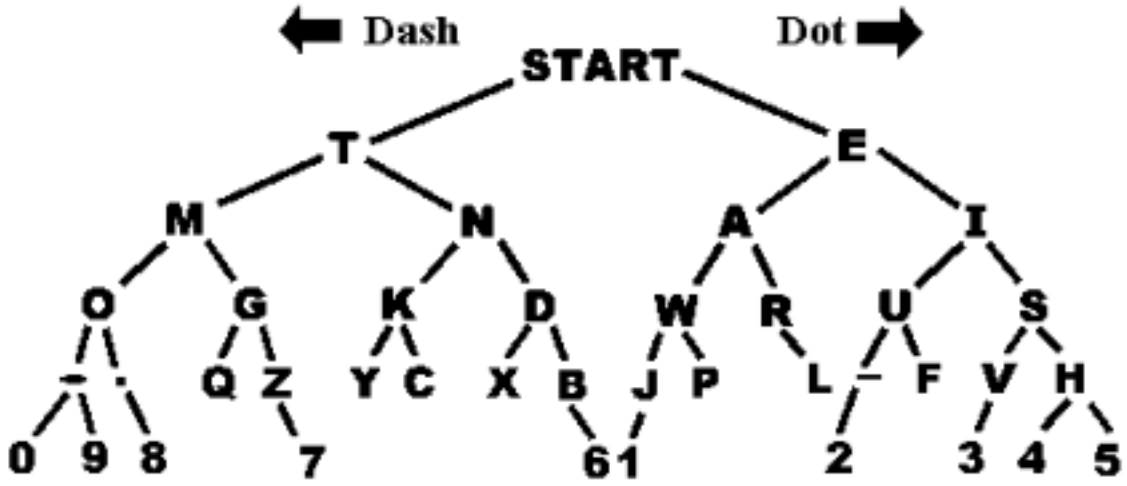
Lab10: Binary Search Tree and Morse Code[1]

CSSSKL162: Programming Methodology Skills

Summary

In this lab, we will build a k-nary tree (where k == 2) that also maintains an ordering property. Samuel Morse's code is pictured below, where the rule is: "dashes go left" and "dots go right". Once we've built and populated our ***Binary Search Tree*** so it looks like the tree below**,** we will be able to traverse the tree and use it to translate a given Character ('S') or String ("SOS") into corresponding dots and dashes ("…" and "…---…", respectively).



1. Warmup with CharTree & CharNode

Build a small **Binary Search** tree that stores characters in its nodes. It should observe the ***class invariant*** that for any given node, all nodes in the left subtree are less than that node (alphabetically) and all nodes in

[1] This lab is created by Rob Nash. Minor edits by A.Retik (Winter 2016).

the right subtree are greater than that node. Use the **driver provided** to print out your tree of characters in order.

```
public static void main(String[] args) {
        CharTree tree = new CharTree();

        tree.add('c');
        tree.add('a');
        tree.add('t');

        System.out.println(tree.toString());   //inorder prints a c t
        System.out.println(tree.countNodes());
        System.out.println(tree.getDepth());

        //System.out.println(tree.contains('c'));
        //System.out.println(tree.getParent('a'));  //can be tricky
}
```

2. Introduction to the Morse Tree

In this section, we'll build our TreeNode **inner class** that will be used by our MorseTree outer class. Once we've made our TreeNode, we'll complete some methods in MorseTree that will make use of these TreeNodes to build a data structure analog of the tree pictured above. Let's start by downloading the skeleton file (MorseTree.java) and read the comments in their entirety. They will draw your attention to specific sections in the code you must complete, starting with the TreeNode class below.

2.1 TreeNode Inner Class

In this section, we'll build an **internal class using generics** that can store one data item and two TreeNodereferences (one for the left child, one for the right). Start by uncommenting the private inner class called TreeNode, at the end of MorseTree.java.

Data & Method Members

- Declare a "Object data;" item to hold a given node's data
- Declare two TreeNode references called left and right
- Declare one constructor that takes three parameters and populates the data members for this structure.

2.2 The MorseTree (Enclosing) Class

In this section, we'll build our MorseTree class by declaring only one private data item (a TreeNode) and multiple public (and private) methods. A common pattern here will be as follows: Clients of our code will

call some public method (say, add(int data)) and our public method will redirect to a private method (insertIntoSubtree(data, root)). We'll follow this pattern in the methods we implement below. **See the Tree code in your Savitch text for additional code samples and guidance.**

Data Members

- Declare a private TreeNode called root (which is similar to head in linked lists)

Method Members

- public void MorseTree() { //Constructor
    - Use this to load data from a file ("data.txt") and populate your Binary Tree.
        - Each line in the file is a pair, as in "S …", which is the letter followed by the morse code equivalent
        - Call the add() function below for each pair read from the file.
- private void insertInSubtree(String morseStr, Character letter, TreeNode subtree)
    - Note that the public add() function has been provided for you
    - Walk the tree while morseStr.length() > 0, removing the leading character from the morse string and…
        - Create a new TreeNode if your subtree is null.
        - Recursively move down the tree, going right if a "." and left if a "-".
- public void findInSubtree(String morseStr, TreeNode subtree)
    - Note that the outer (wrapper) function translate() has been provided for you.
    - Walk the tree while the morseStr.length() is greater than 0, removing the leading character from the morse string and…
        - Recursively move down the tree, going right if we encounter a "." and left otherwise.
- public void toMorseCode()
    - Look at this function inside MorseTree.java
        - Completing this step is optional for this lab
- public String toString()
    - Look at this function inside MorseTree.java
        - Completing this step is optional for this lab