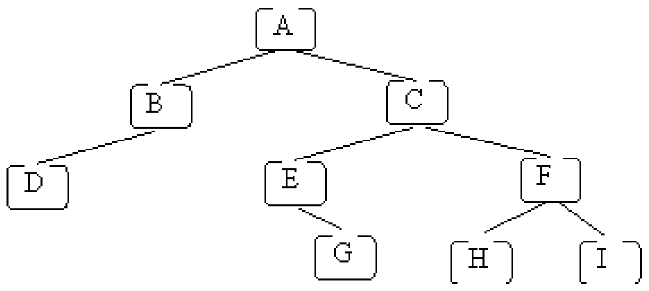
Tree Traversals

Trees are non-linear data structures, and therefore can't be traversed in a linear fashion (as you can for arrays or linked lists). For trees, there are many different orders in which we might visit the nodes. There are some common traversal orders for trees: ***pre-order***, ***in-order***, ***post-order***, and ***level-order***, all described below. We will use the following tree to illustrate each traversal:



In addition to the descriptions below, this video has a nice explanation of the traversals, along with the "VLR" (and "LVR" / "LRV") methods for remembering the traversals.

Pre-order

A pre-order traversal can be defined (recursively) as follows:

1. **visit the root**
2. perform a pre-order traversal of the first sub-tree of the root
3. perform a pre-order traversal of the second sub-tree of the root
4. etc. for all the sub-trees of the root

If we use a pre-order traversal on the example tree given above, and we print the letter in each node when we visit that node, the following will be printed: `A B D C E G F H I`.

Post-order

A post-order traversal is similar to a pre-order traversal, except that the root of each sub-tree is visited **last** rather than first:

1. perform a post-order traversal of the first sub-tree of the root
2. perform a post-order traversal of the second sub-tree of the root
3. etc. for all the sub-trees of the root
4. **visit the root**

If we use a post-order traversal on the example tree given above, and we print the letter in each node when we visit that node, the following will be printed: `D B G E H I F C A`.

In-order

An in-order traversal involves visiting the root "in between" visiting its left and right sub-trees. Therefore, an in-order traversal only makes sense for binary trees. The (recursive) definition is:

1. perform an in-order traversal of the left sub-tree of the root
2. **visit the root**
3. perform an in-order traversal of the right sub-tree of the root

If we print the letters in the nodes of our example tree using an in-order traversal, the following will be printed: `D B A E G C H F I`

The primary difference between the pre-order, post-order and in-order traversals is where the node is visited in relation to the recursive calls; i.e., before, after or in-between.

Level order

The idea of a level-order traversal is to visit the root, then visit all nodes "1 level away" (depth 2) from the root (left to right), then all nodes "2 levels away" (depth 3) from the root, etc. For the example tree, the goal is to visit the nodes in the following order:

$$\underbrace{A}_{\text{root}}\quad \underbrace{B\quad C}_{\substack{\text{1 level}\\ \text{away}}}\quad \underbrace{D\quad E\quad F}_{\substack{\text{2 levels}\\ \text{away}}}\quad \underbrace{G\quad H\quad I}_{\substack{\text{3 levels}\\ \text{away}}}$$

A level-order traversal requires using a queue (rather than a recursive algorithm, which implicitly uses a stack). Here's how to print the data in a tree in level order, using a queue Q, and using an iterator to access the children of each node (we assume that the root node is called `root`, and that the TreeNode class provides a `getChildren` method):

```
Q.enqueue(root);
while (!Q.empty()) {
   Treenode n = Q.dequeue();
   System.out.print(n.getData());
   List L = n.getChildren();
   Iterator it = L.iterator();
   while (it.hasNext()) {
         Q.enqueue(it.next());
   }
}
```