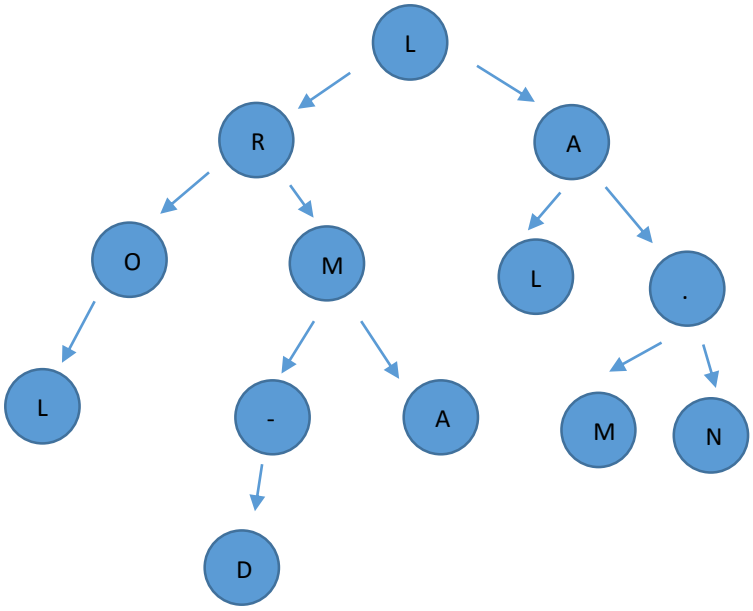
Question: Discuss an algorithm to traverse a tree, depth first.

Let me start by defining a tree to be a logical collection of entities (nodes), linked together to form a hierarchical structure e.g. an organizational chart or random entries. For the cause of this explanation I will be using the tree example shown below with random char entries.



Ok, so we have a tree structure here.

The depth first algorithm I will be discussing is the “Inorder” Traversal. Well simply put it means we will be using the following rule to traverse the tree:

Left,Read,Right, meaning from the root node of “L” we will move to its left child “R”, assuming “R” has no children, we will then read its value and move back to “L” and read its value after which we move to “L”’s right child which is “A” And apply the same rule to assuming it has children. After treating the left child node of “A” and its descendants if any, we return and read “A” before proceeding to the right child and its descendants. Thus, we always start with the deepest node on the left in this case.

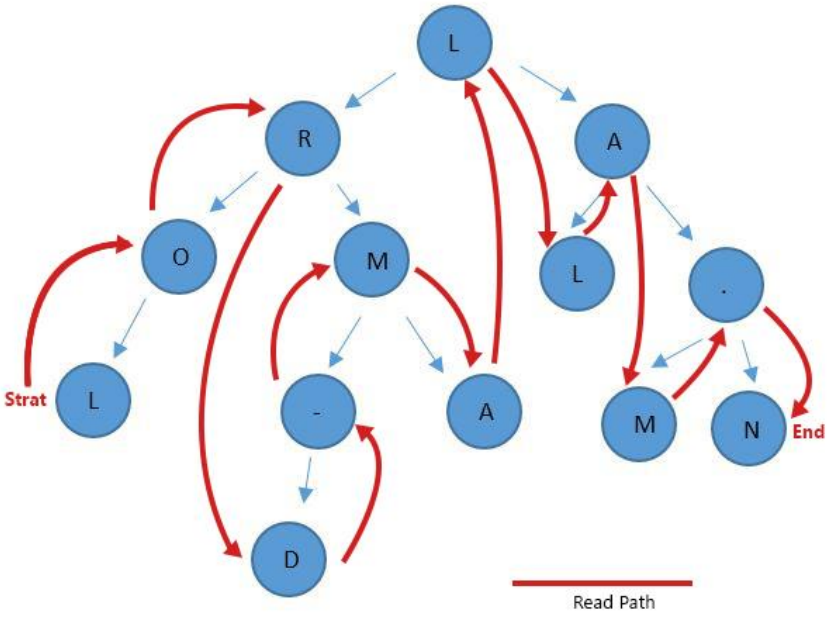
Let’s, write this out in steps:

InOrder Traversal (InOrder)

1. Go to left child node and perform “InOrder” on it.
2. Print out the root value, regardless of it being the actual root.
3. Go to Right child node and perform “InOrder” on it.
4. Do this recursively

Now, using the rules and steps above, traversing the tree above will result in:

LORD-MALLAM.N



The Java pseudocode will be:

```
public class Node {

private Node right;
private Node left;
private char nodeValue;

public Node leftNode() {return left;}
public Node rightNode() {return right;}
public char NodeValue() {return nodeValue;}

}


void inOrder (Node root)
{

if(root == null) return;
```

```
inOrder(root.leftNode());
root.printNodeValue();
inOrder(root.rightNode());

}
```

Alright, I guess I am done.
O yes I won't go into discussing its space complexity etc. at this time.

Yours Truly,
Lord-Mallam