**DEPTH FIRST TREE TRANSVERSAL**

Tree transversal algorithm focuses on the series of steps taken traverse (move around) the entire nodes of a particular data structure in a fixed order.

There are two types of tree transversal namely the breadth first and depth first transversal.

According to the name "depth first" tree transversal starts search from the parent node, then through the nodes in a downward motion till it reaches a dead end for the particular branch.

Each traversed or visited nodes in the tree are usually being noted by putting them in stacks so as to keep track of the search progress.

When it reaches a dead end node in a particular branch, The dead end node is also put in the stack and checks if the node has a branch or not.

If it has a branch, it moves to the branch and continues the search in the similar manner as above and putting each traversed node into the stack accordingly.

if the node does not have a branch, it removes the node from the stack and checks each node in the stack in a last in first out iterative manner.

If any of the node in the stack has a branch it moves to the branch and continues the process by putting each node in that branch in a stack.

If none of the node has an available branch(that is all branches are checked) it iterates back to the starting node by iterating through the stack till the stack is empty and exits the process.

The depth first transversal loops through the nodes in the tree in a vertical manner (like a top down approach) without considering the branch till it reaches a dead end.

ILLUSTRATION

Using the tree diagram above, assume a Stack S is being initialized at the beginning of the search

1. The algorithm starts with node A. checks node A, puts it into a stack i.e S = A.

2. In alphabetical order, it continues with Branch B. checks node B, puts it into stack S. S = A, B.

3. Moves to node C, checks it and puts it into stack S = A, B, C .

4. Moves to node D, checks it and puts it into stack S = A, B, C, D .

5. Node D is the dead end but it has two branches E and F, in alphabetical order, branch E is chosen. Moves to node E, checks it and puts it into stack S = A, B, C, D, E .

6. Node E is a dead end with an already checked starting node A and node E attached to it. So it opens the stack, removes Node E (dead end). so S = A, B, C, D.

7. It checks Node D again. Node D has an unchecked node F, so it is checked and put into the stack, so that stack S is now S = A,B,C,D,F.

8. Node F is also a dead end, so in a similar way as Node E , it checks the stack S and removes the nodes from the stack S in a last in first out manner and checks for unchecked branches. But since in the illustration, all the branches are already checked, it loops through the stack till it reaches the starting point in the Stack i.e A and empties it.

Iteration Manner

1. S = A,B,C,D,F.

2. S = A,B,C,D.

3. S = A,B,C.

4. S = A,B.

5. S = A. Empties the stack S, then exits the program.