Tree walks | Traversals

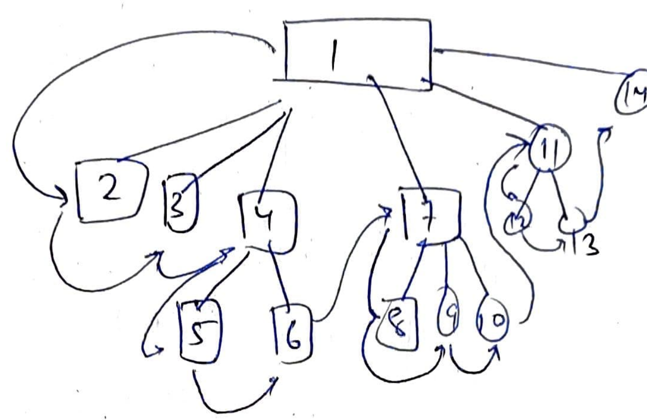
→ A tree walk is a way of visiting all the nodes in a tree in a specific order.

→ A preorder tree walk processes each node before processing its children.

→ A postorder tree walk processes each node after processing its children.

Preorder example (Book contents)

Suppose, we need to make a table of contents from this tree., then we go in order of pre order traversal.



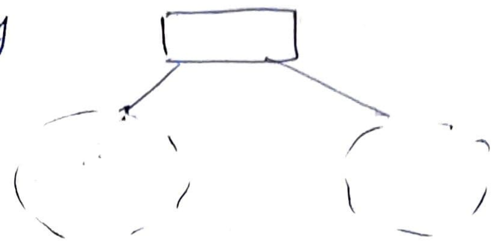
→ Algorithm

preOrder(v):
'visit' node v
for each child w of v do
recursively perform preOrder(w).

→ ex. is reading a document / paper from beginning to end.

Postorder traversal. → First visit children and then visit node.

let us say we need to calculate space occupied by a tree structure.



→ postorder.(v)

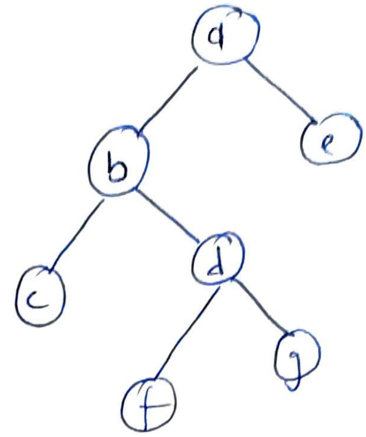
for each child w of v do.
recursively perform postorder(w).
(visit) node v afterwards.

Traversals of Binary trees

```
preorder (v):
   if (v == null), then return.
   else:   visit (v)
           preorder ( v.leftchild())
           preorder (v.rightchild())
```

```
postorder (v) :
   if (v == null) , then return.
   else:  postorder ( v.leftchild())
          postorder ( v.rightchild())
          visit (v).
```

→ assume, we are just printing the content on visit.



Preorder.

a b c d f g e

Postorder.

c f g d b e a.

→ evaluating arithmetic expression can be done using post order traversal. Because we need to calculate the value of left subtree and right subtree before visiting the node.

algorithm evaluate(v):

if v is a leaf:

return the value stored in v

else:

let o be the operator stored at v

x → evaluate (v.leftchild())

y → evaluate (v.rightchild())

return x o y.

Inorder traversal

→ Besides preorder and postorder, a third possibility arises when v is visited between the visit to left and right subtree.

algorithm.

```
inOrder (v) :
if (v == null), then return.
else:
    inOrder( v.leftChild())
    visit(v)
    inOrder (v.rightChild())
```

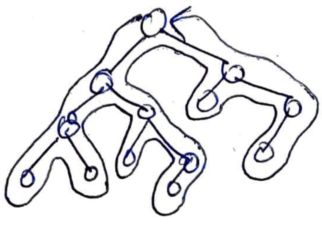
ex. same as prev.

c b f d g a e.

Euler Tour Traversal

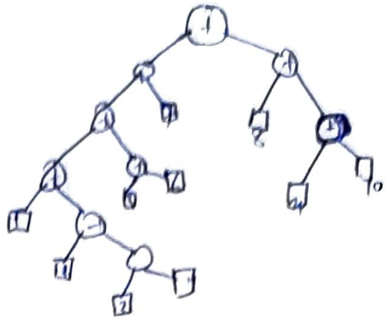
→ generic traversal of a binary tree
→ preorder, postorder, inorder. traversals are special cases of Euler tour traversal.

→ in ETT, walk around the tree and visit each node three times → on left
on right
from below.



Printing an arithmetic expression.

→ Print "(" before traversing the left subtree, print the value of node when visiting from bottom, print ")" when visiting after traversing right subtree



$$((((1 + (2 + (3+4))) + (5+6)) + 7) +$$
$$(8 + (9 + 10)))$$

Building a tree from pre and in order.

→ Given the preorder and inorder traversals of a binary tree, we can uniquely determine the tree.

Preorder

$\underset{\text{root}}{a}$ $\underline{b\ c\ d\ f\ g}$ e

In order

$\underbrace{c\ b\ f\ d\ g}_{\text{left subtree}}$ $\boxed{a}$ $\underset{\text{right subtree}}{e}$.

So, we know the root

now, for the left elements in inorder (no),
we have that many no. of elements in preorder
as the left subtree.

Now, we have a subproblem. we can then use recursion.

Programming assignment 1 : Given a preorder and inorder, determine the tree. (12)

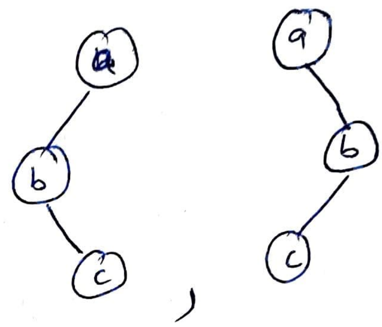
Similarly, given the postorder and inorder traversal, a tree can be uniquely determined.

This is not the case if we are given preorder and postorder traversal.

Counter example.

preorder : a b c

postorder. c b a

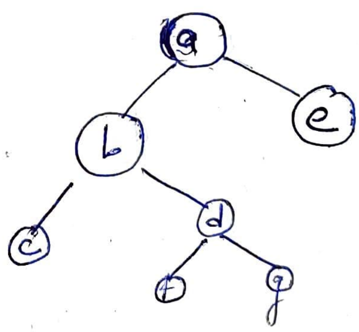


special case

If each internal node of the binary tree has atleast two children, then the tree can be determined from the pre and post order traversals.

Preorder Postorder

a b c d f g $\boxed{e}$ c f g d b $\boxed{e}$ a.



Now, left subtree has

b c $\boxed{d}$ f g as preorder traversal

and c f g d b as postorder.

d f g

f g d