Traversal on Tree

Unit-3 Lecture-9

Dr. Dillip Rout, Assistant Professor, Dept. of Computer Science and Engineering

Outline

- Fundamentals of Traversal
- Recursive Traversal
- Non-recursive Traversal
- Homework
- Conclusions

Fundamentals of Traversal on Binary Tree

Traversal

- Traversing is a process in which each element of a data structure is accessed.
- Accessing an element of data structure means visiting every element at least once.
- Traversing is performed to display every element of data structure or to perform any operation on its element.
- Several ways to traverse on a tree or graph since these are non-linear data structures.

Traversal on Binary Tree

Various ways of traversing a Binary Tree:

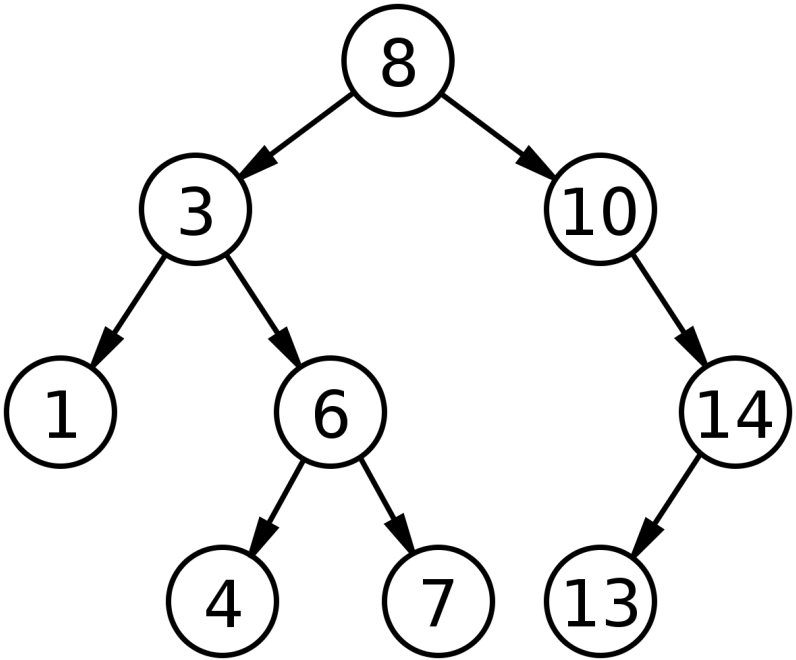
- Pre-order: N L R
  - Visit the root/node first
  - visit left sub-tree
  - visit right sub-tree
- In-order: L N R
  - visit left sub-tree first
  - Visit the root/node
  - Visit right sub-tree
- Post-order: L R N
  - visit left sub-tree
  - visit right sub-tree
  - visit the root/node

Traversal Example

- Pre-order: Node-Left-Right
  - 8, 3, 1, 6, 4, 7, 10, 14, 13
- In-order: Left-Node-Right
  - 1, 3, 4, 6, 7, 8, 10, 13, 14
- Post-order: Left-Right-Node
  - 1, 4, 7, 6, 3, 13, 14, 10, 8

Which one has the root at the beginning?

Which one has produced a sorted order?



Knowledge Check

Q1. Level order Traversal of a rooted tree starting from the root can be done by performing:

a) Depth First Search

b) Root Search

c) Deep Search

d) Breadth-First Search

Q2. A queue data structure is used in the following tree traversals?

a) Preorder

b) Inorder

c) Postorder

d) Level order

Example Problem-1

If the trees Tree-1 and Tree-2 are the ones listed aside:

Which traversals of Tree-1 and Tree-2 will yield the same sequence, respectively?

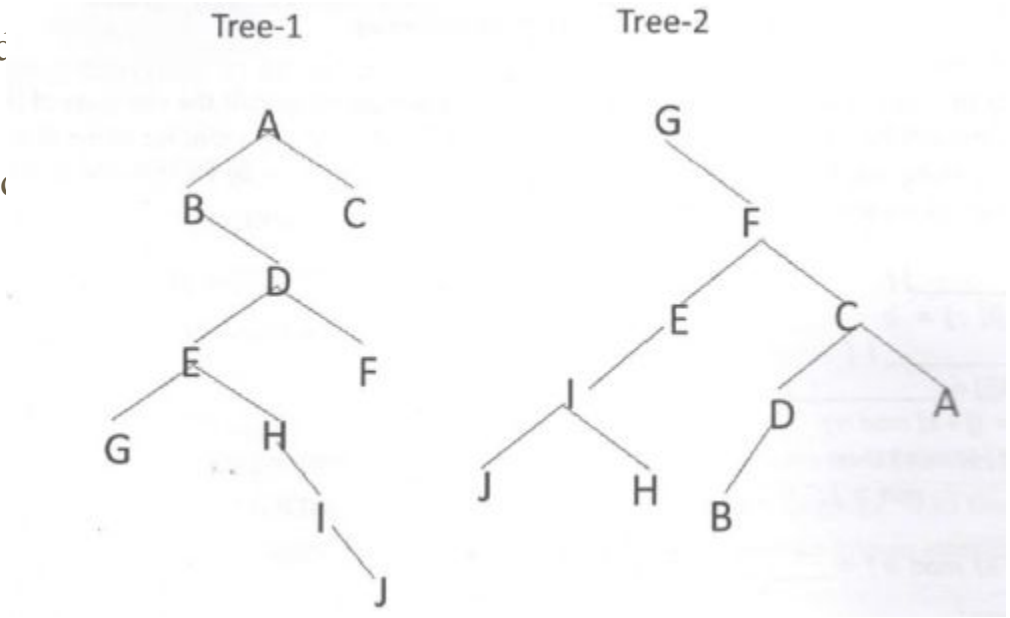
a) Postorder, inorder

b) Postorder, preorder

c) Inorder, preorder

d) Preorder, postorder

e) None of the above



Example Problem-2

What do the three types of traversals (Inorder, Preorder, and Postorder) have in common?

a) The root is visited before the right subtree

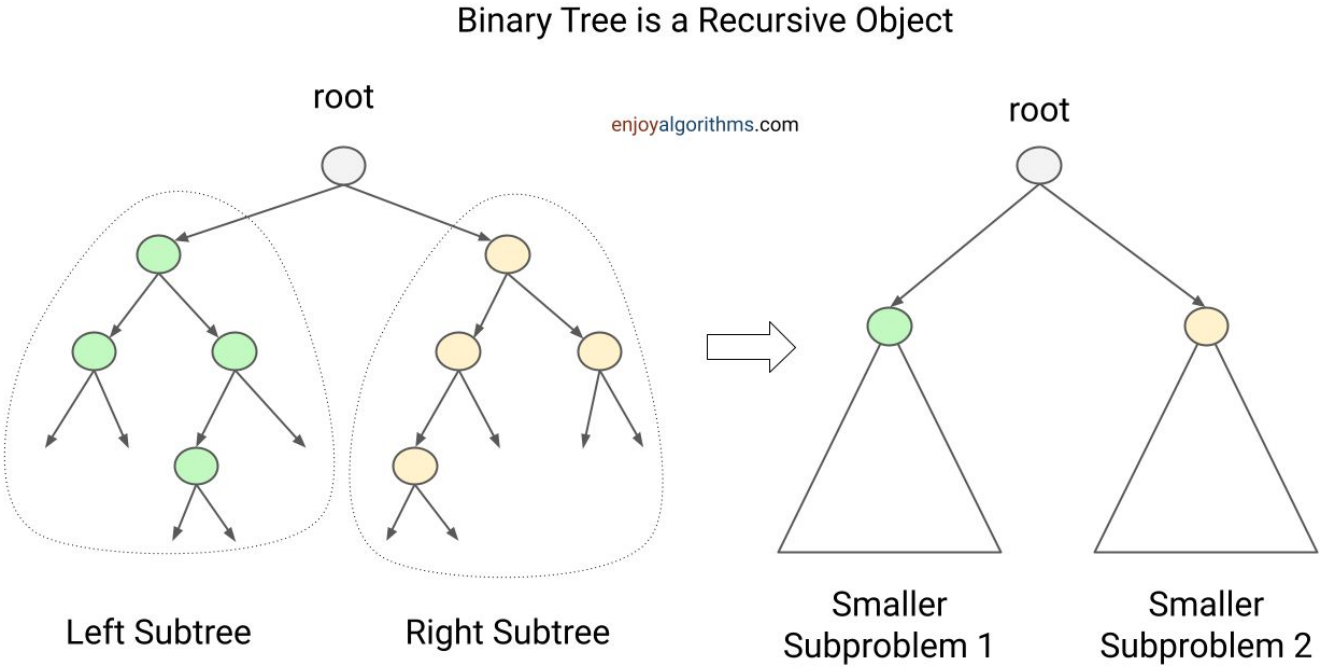
b) The left subtree is always visited before the right subtree

c) The root is visited after the left subtree

d) All of the above

Recursive Traversal

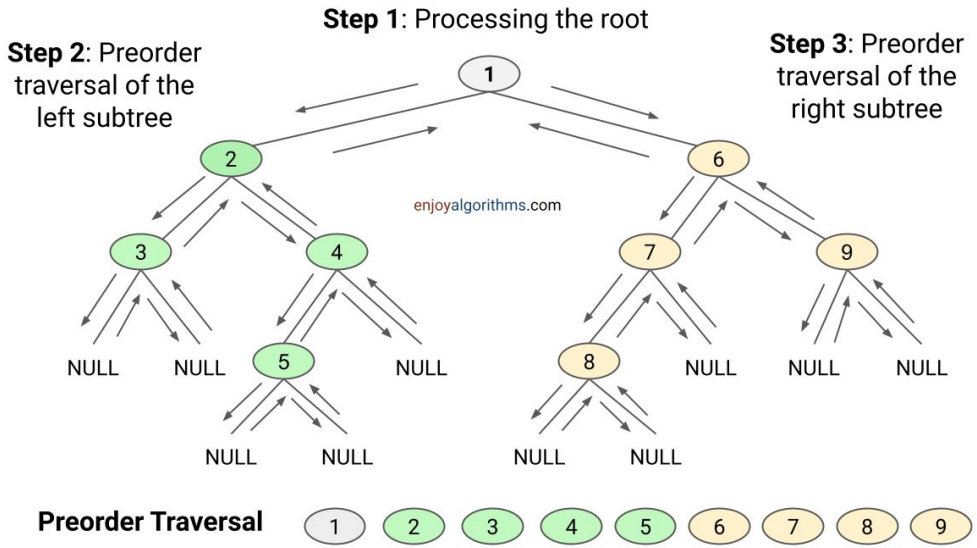
Recursion is Natural Choice



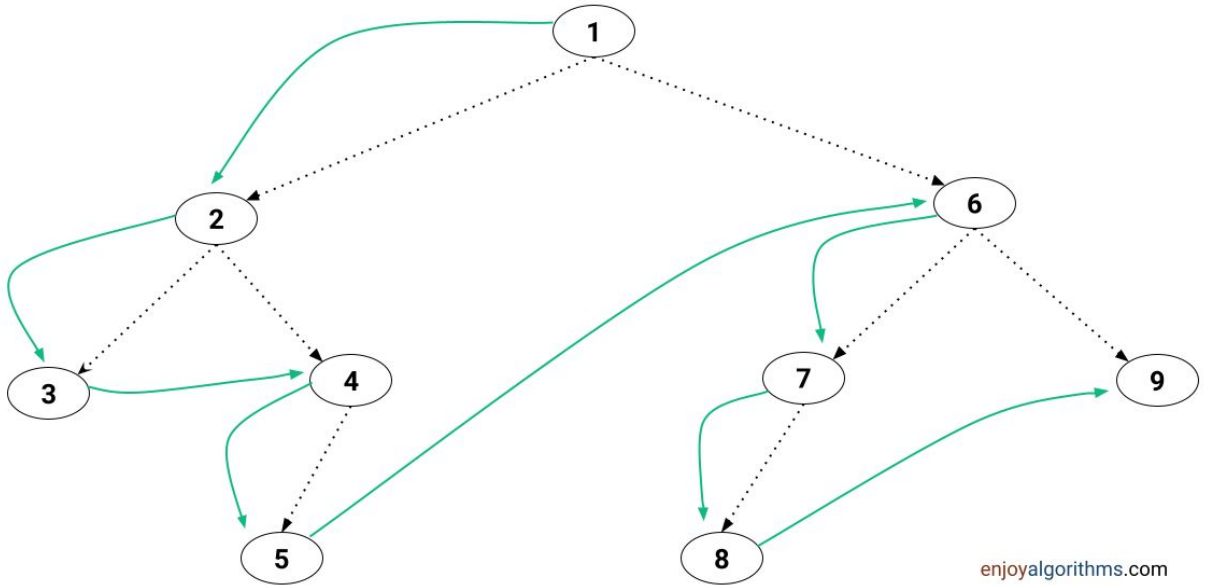
Pre-order: Root-Left-Right

Algorithm Preorder (TreeNode root)

1. If root == NULL then
2. Return
3. Process (root.data)
4. Preorder(root.left)
5. Preorder(root.right)



Pre-order: Root-Left-Right...

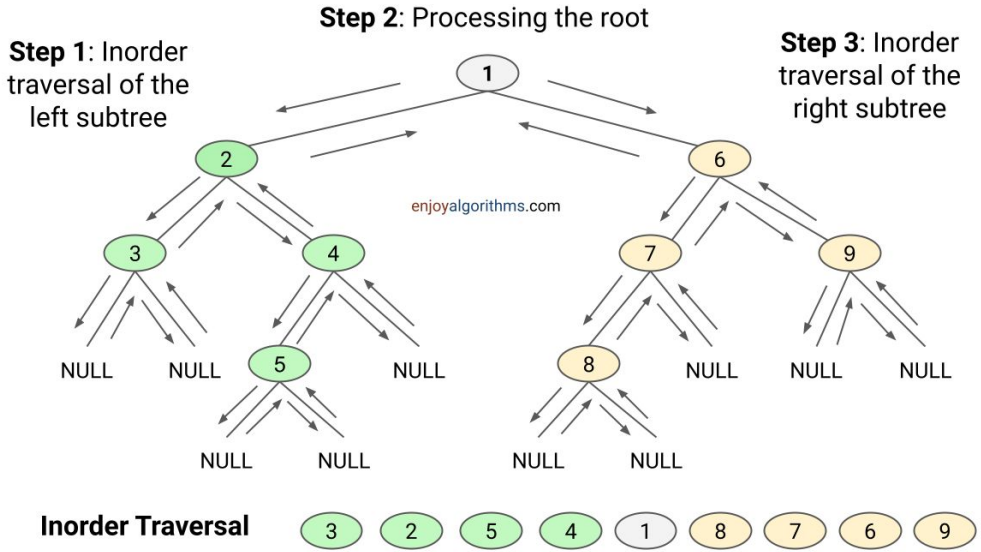


Preorder Traversal 1 2 3 4 5 6 7 8 9

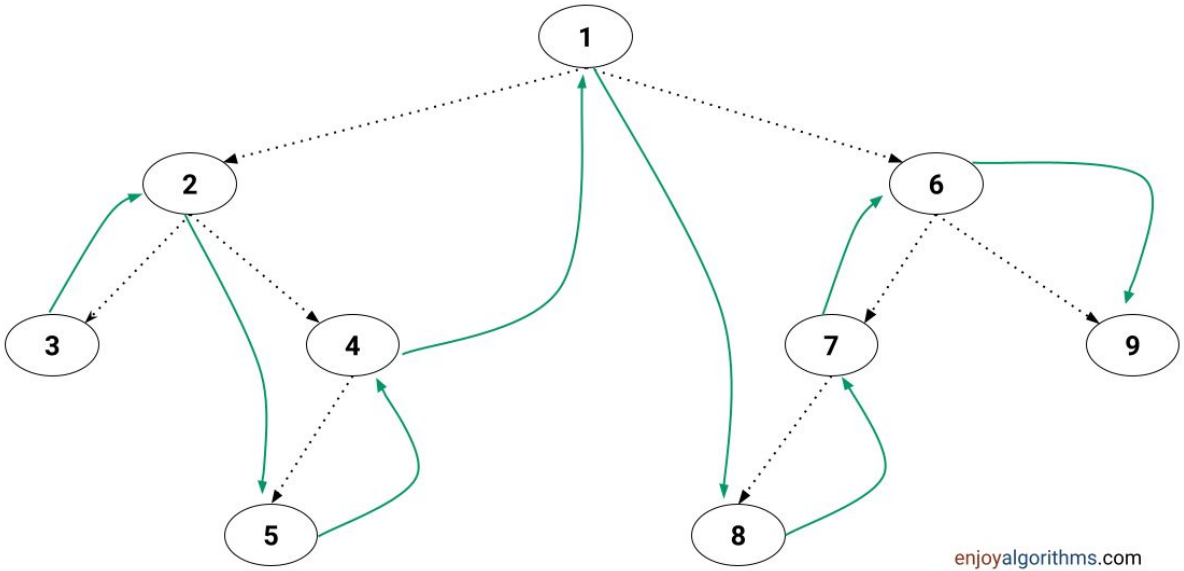
In-order: Left-Root-Right

Algorithm In-order (TreeNode root)

1. If root == NULL then
2. Return
3. In-order(root.left)
4. Process (root.data)
5. In-order(root.right)



In-order: Left-Root-Right...

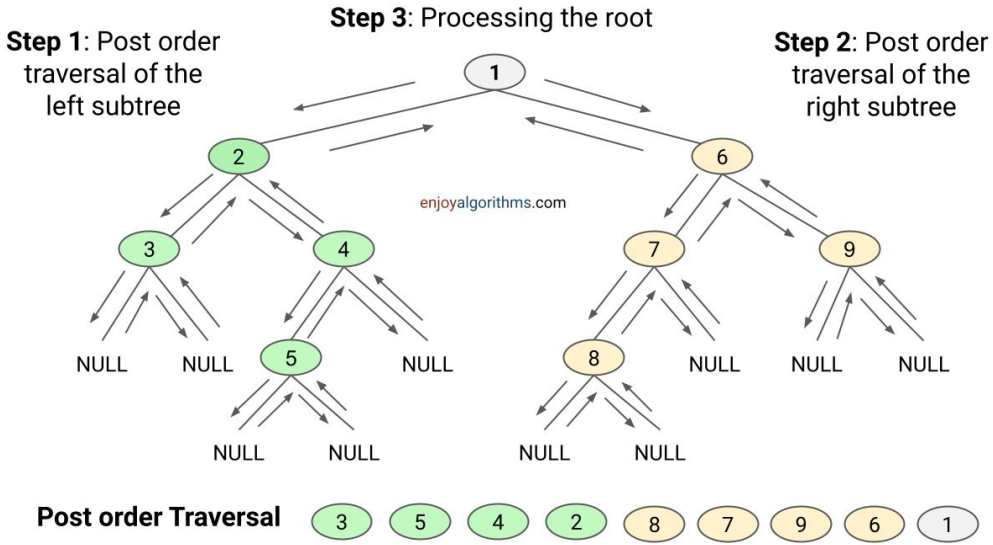


Inorder Traversal 3 2 5 4 1 8 7 6 9

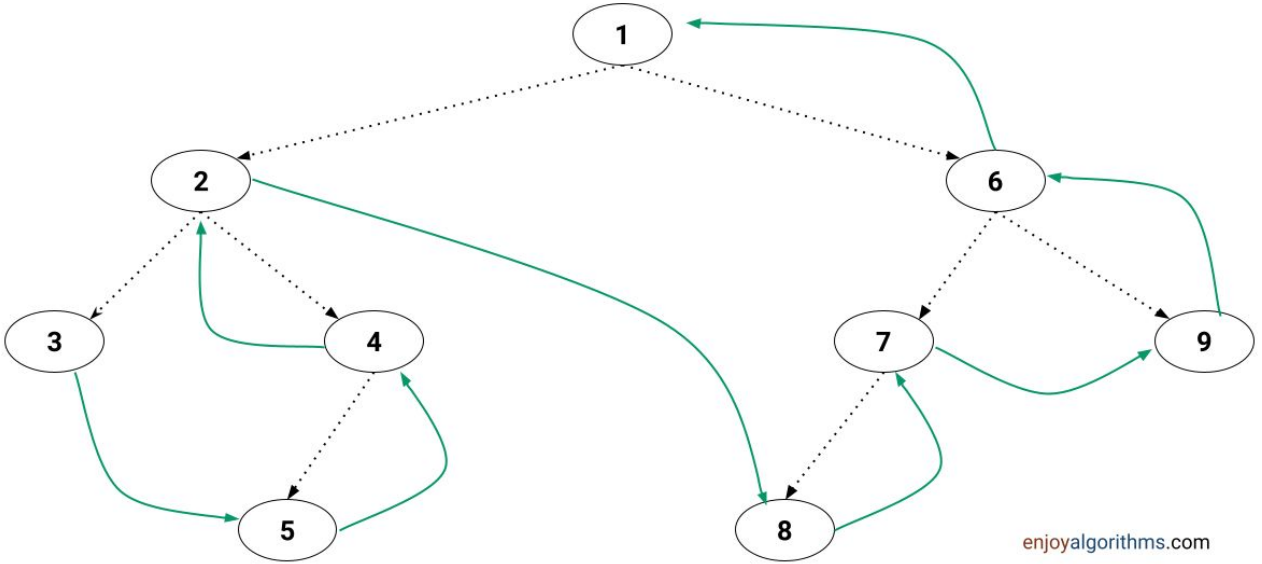
Post-order: Left-Right-Root

Algorithm Postorder (TreeNode root)

1. If root == NULL then
2. Return
3. Postorder(root.left)
4. Postorder(root.right)
5. Process (root.data)



Post-order: Left-Right-Root...



Example Problem-1

If the trees Tree-1 and Tree-2 are the ones listed below:

Which traversals of Tree-1 and Tree-2 will yield the same sequence, respectively?

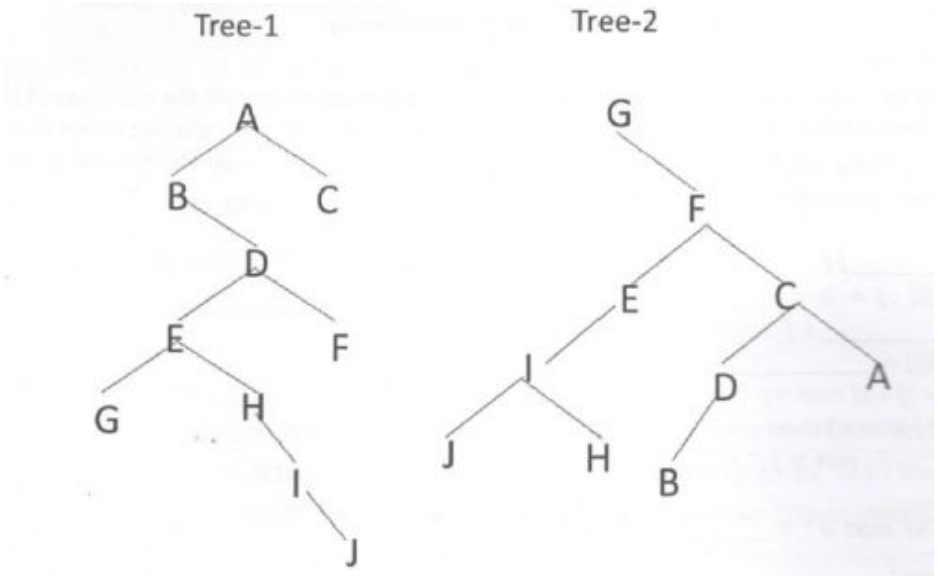
a) Postorder, inorder

b) Postorder, preorder

c) Inorder, preorder

d) Preorder, postorder

e) None of the above



Example Problem-2

dbeafcg and abdecfg are the inorder and pre-order Traversals of a binary Tree, respectively. The postorder traversal is

a) debfagc

b) dbefcga

c) debfgca

d) dbefacg

Example Problem-3

Consider the following statements:

(A) A rooted tree can be traversed using a Depth-first search.

(B) To list the vertices of an ordered rooted tree, pre-order, postorder, and inorder are performed.

(C) Huffman's approach is utilized to find an optimal binary tree with specified weights.

(D) Topological sorting assigns labels to parents greater than those assigned to their children.

The topological sort algorithm takes a directed graph and returns an array of the nodes where each node appears before all the nodes it points to. Used for scheduling.

Huffman coding is a method of data compression that is independent of the data type, that is, the data could represent an image, audio or spreadsheet. This compression scheme is used in JPEG and MPEG-2.

Which of the following statements is correct?

a) (A) and (B)

b) (C) and (D)

c) (A), (B) and (C)

d) (A), (B), (C) and (D)

Non-recursive Traversal

Non-recursive Traversal

- Recursion can also be implemented using stack.
- Non-recursive traversal can be performed using a stack.
- Nodes are pushed onto the stack instead of calling the same function in recursion for the nodes.
- The size of the stack should be defined by taking the height into consideration.

Pre-order: Root-Left-Right

Algorithm Preorder (TreeNode root)

```
1.  If root == NULL then
2.      Return
3.  Process (root.data)
4.  Preorder(root.left)
5.  Preorder(root.right)
```

How different non-recursive approach?

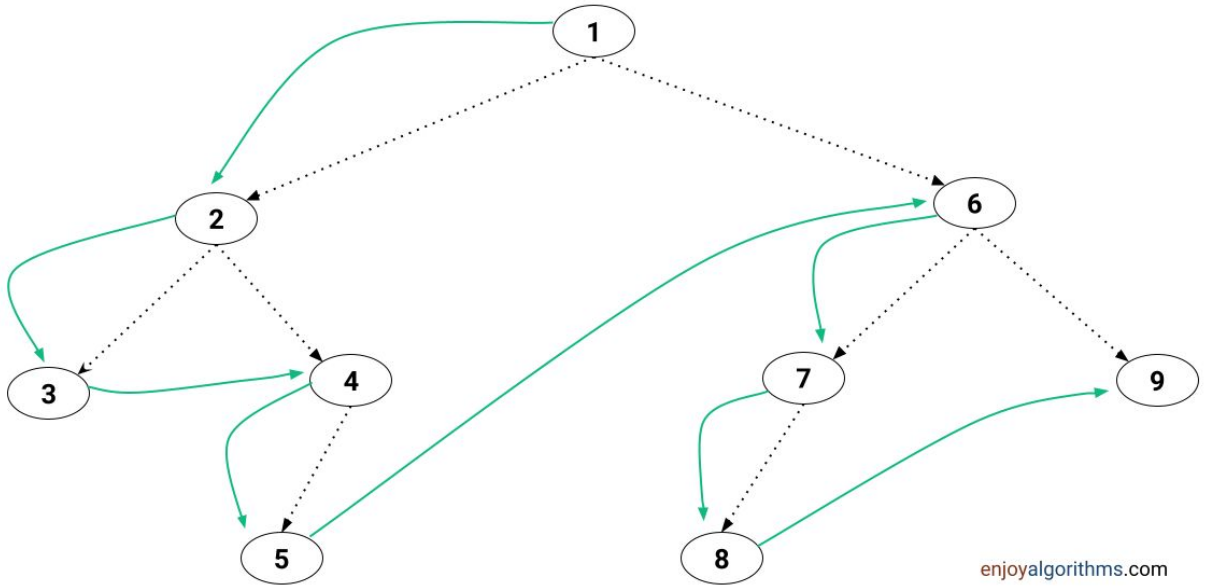
How similar non-recursive approach?

Algorithm Preorder (TreeNode root, Stack S)

```
1.  If root == NULL then
2.      Return
3.  S.Push (root)
4.  while S != ϕ do
5.    node = S.Pop ()
6.    If node != NULL then
7.       Process (node.data)
8.       S.Push(node.right) // may test NULL
9.       S.Push(node.left)    // may test NULL
```

What does Line 3 mean?

Pre-order: Root-Left-Right...



Preorder Traversal 1 2 3 4 5 6 7 8 9

In-order: Left-Root-Right

Algorithm In-order (TreeNode root)

```
1.  If root == NULL then
2.      Return
3.  In-order(root.left)
4.  Process (root.data)
5.  In-order(root.right)
```

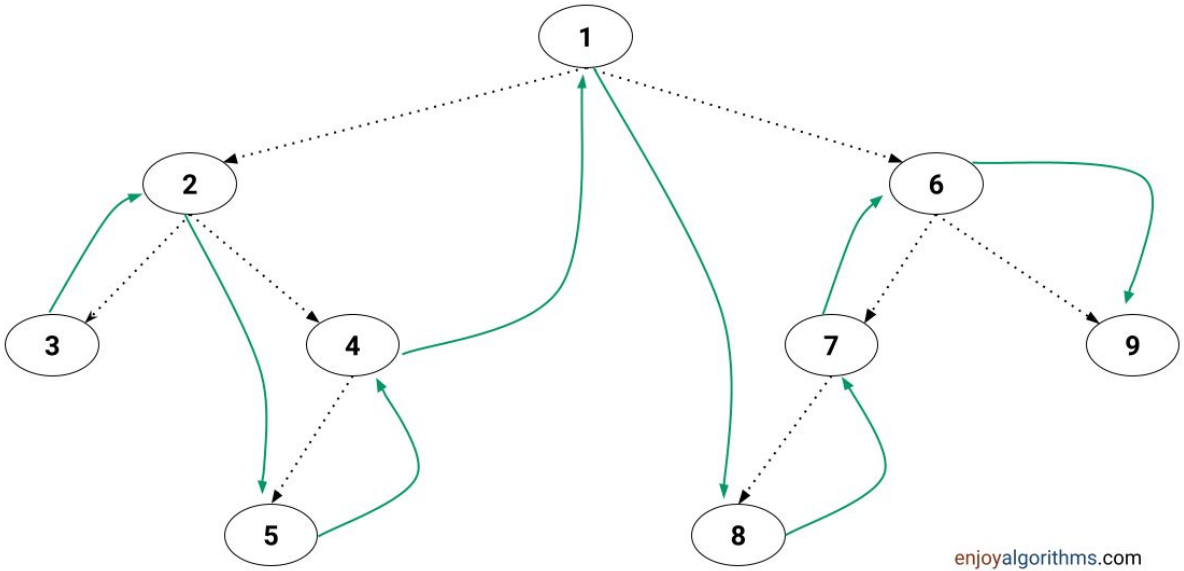
Algorithm Preorder (TreeNode root, Stack S)

```
1.  If root == NULL then
2.      Return
3.  node = root
4.  while S != ϕ or node != NULL do
5.      while node != NULL do
6.          S.push(node)
7.          node = node.left
8.      node = S.pop()
9.      Process (node.data)
10.     node = node.right
```

What is line 5-7?

Is line 4 always correct?

In-order: Left-Root-Right...



Inorder Traversal 3 2 5 4 1 8 7 6 9

Post-order Left-Right-Root

1.1 Create an empty stack

2.1 Do following while root is not NULL

a) Push root's right child and then root to stack.

b) Set root as root's left child.

2.2 Pop an item from stack and set it as root.

a) If the popped item has a right child and the right child is at top of stack, then remove the right child from stack, push the root back and set root as root's right child.

b) Else print root's data and set root as NULL.

2.3 Repeat steps 2.1 and 2.2 while stack is not empty.

Post-order: Left-Right-Root

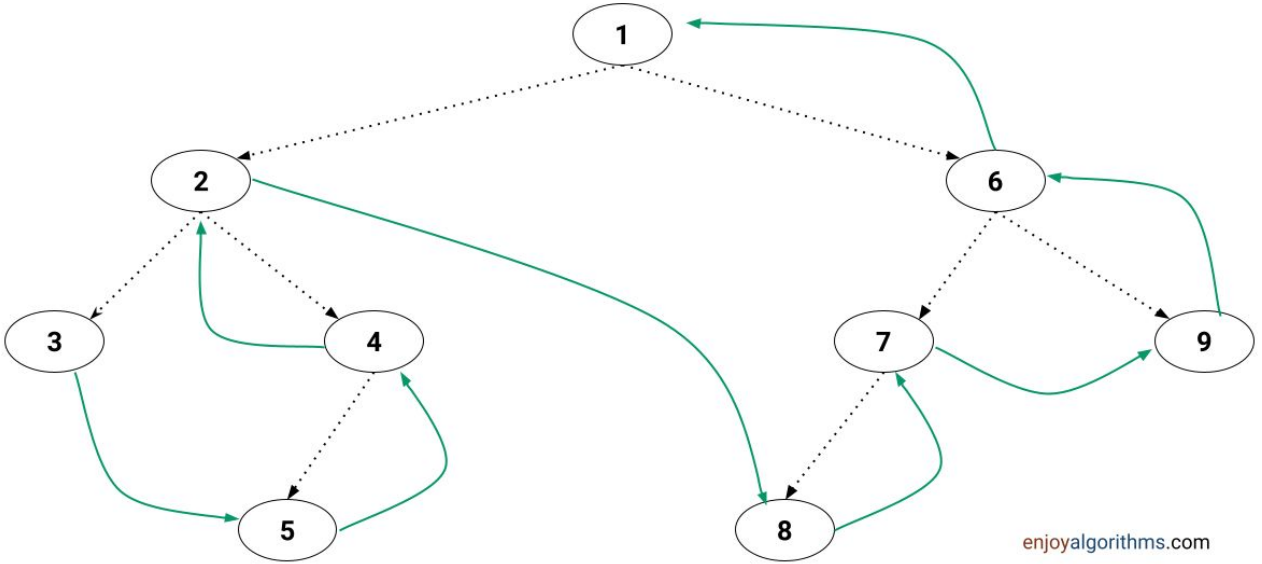
Algorithm Postorder (TreeNode root)

```
1.  If root == NULL then
2.      Return
3.  Postorder(root.left)
4.  Postorder(root.right)
5.  Process (root.data)
```

Algorithm Preorder (TreeNode root, Stack S)

```
1.  If root == NULL then return
2.  node = root
3.  do
4.     while node != NULL do
5.        If node.right != NULL
6.           S.push(node.right)
7.        S.push(node)
8.        node = node.left
9.     node = S.pop()
10.    If node.right == S.top()
11.       tmp = S.pop()
12.       S.push(node)
13.       node = tmp
14.    Else
15.       Process (node.data)
16.       node == NULL
17. while S != ϕ
```

Post-order: Left-Right-Root...



Knowledge Check

Consider the rooted tree below, with P as the root vertex.

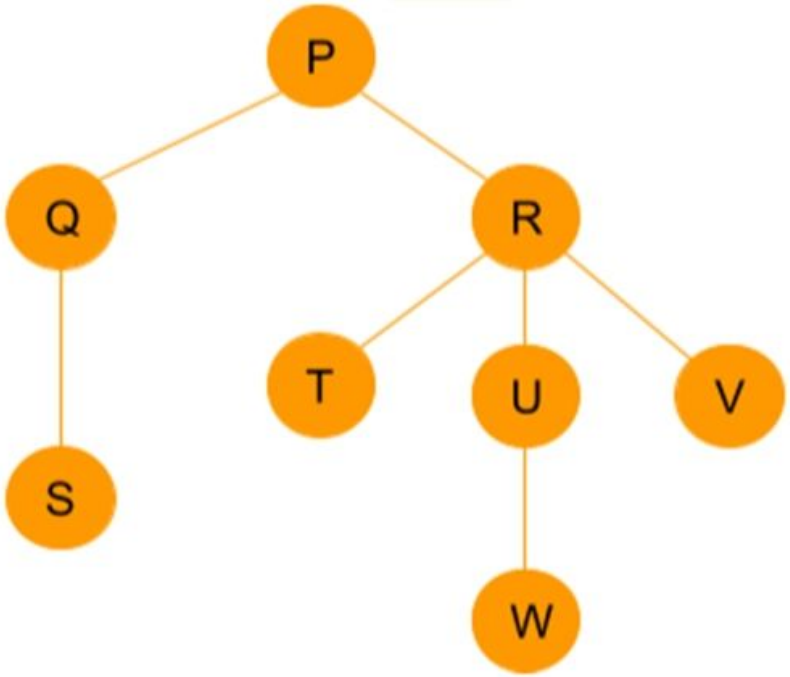
During in-order traversal, the nodes are visited in the following order:

a) SQPTRUWV

b) SQPTRWUV

c) SQPTURWV

d) SQPTWUVR



Example Problem-1

Assume that +,-, x and are left-associative, and that ^ is right-associative. The precedence order is ^, ×, +, −. The postfix expression corresponding to the infix expression "a + b × c − d ^ e ^ f " is

a) abc x + de ^ f ^ −

b) ab + c × d − e^f^

c) abc x + def ^ ^ −

d) − + a × b c^^ def

Hints: put brackets appropriately and try.

Is binary tree traversal helpful in this scenario?

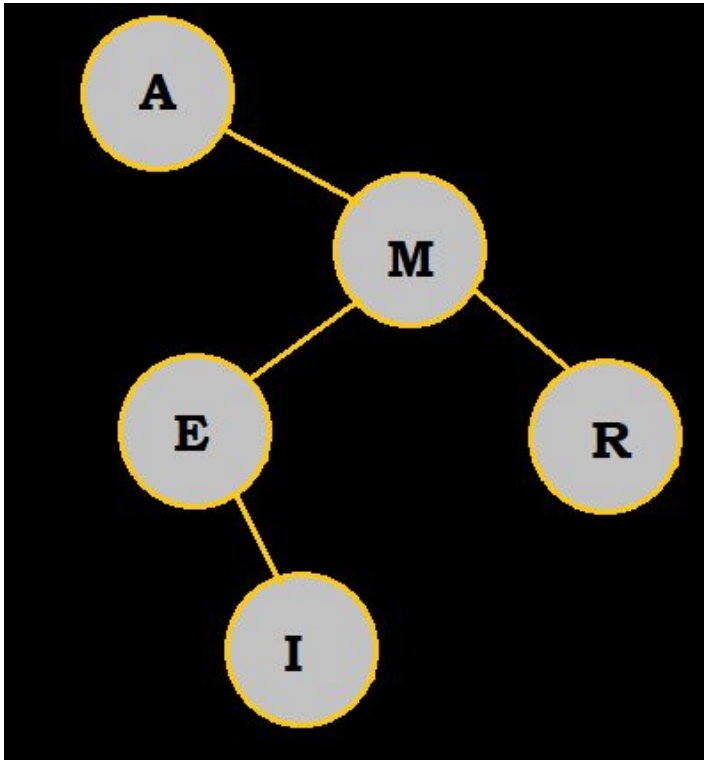
What will be the approach in the competitive examination?

Which is the operator to be performed at the end.

Example Problem-2

Suppose your are implementing post-order traversal for the given binary tree. The traversal needs to be implemented in non-recursive manner. Then, which elements will be pushed more than once onto the stack in this scenario.

I. M only
II. A, M only
III. All the nodes
IV. None of the nodes



Homework

- Theory
- Practice

Homework

- Compare between pre-order and post-order traversal of binary trees. Illustrate the differences with suitable example.
- Compare between pre-order and post-order traversal of binary trees when implemented using non-recursive method. Illustrate the differences with suitable example. Which one is actually faster? How many times a single node is pushed and popped from the stack?
- Design a binary tree which will produce the same result for pre-order as well as post-order traversal. The tree must have 10 nodes at least.
- Design a binary tree which will produce the same result for pre-order as well as post-order traversal. Is it possible to draw such a tree if it has left and right subtree? Support your statement with examples.
- Write a recursive algorithm for traversal reverse post-order traversal. In this traversal, the nodes are visited like Right-Root(Node)-Left.
- Write a non-recursive algorithm for traversal reverse post-order traversal. In this traversal, the nodes are visited like Right-Root(Node)-Left.
- Write a recursive algorithm for traversal reverse pre-order traversal. In this traversal, the nodes are visited like Root(Node)-Right-Left.
- Write a non-recursive algorithm for traversal reverse pre-order traversal. In this traversal, the nodes are visited like Root(Node)-Right-Left.
- Write a non-recursive algorithm for post-order traversal in a binary tree. You need to use two stacks for it. Compare it with the one stack version of the algorithm taught in the class.

Conclusion