CSE-213
(Data Structure)

Lecture on
Traversal of Binary Tree

Md. Jalal Uddin

Lecturer

Department of CSE

City University

Email: jalalruice@gmail.com

No: 01717011128 (Emergency Call)

Department of Computer Science & Engineering (CSE)
City University, Khagan, Birulia, Savar, Dhaka-1216, Bangladesh

Lecture-12: Traversal of Binary Tree

Objectives of this Lecture:

- ❖ Traversing binary trees:
  - Preorder traversal
  - Inorder traversal
  - Postorder traversal
- ❖ Threads and threaded trees:
  - Convert a binary tree to threaded tree

Binary Tree Traversal

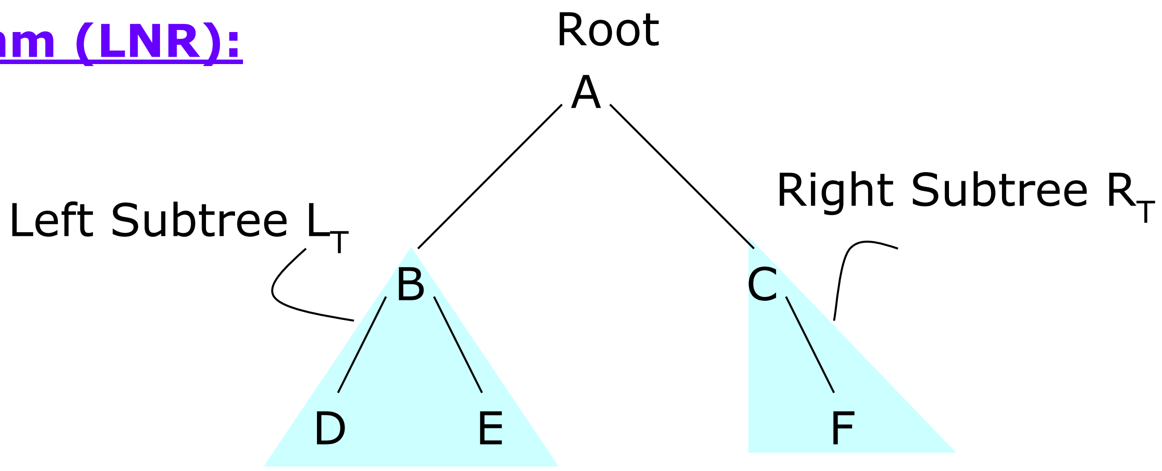
- In a traversal, each element of the binary tree is **visited** exactly once.
- Many binary tree operations are done by performing a **traversal** of the binary tree.

Ways of traversing a binary tree:

- There are three standard ways to traverse a binary tree. These are:
- ❑ In-order traversal or Left-Root-Right (LNR) traversal
- ❑ Pre-order traversal or Root-Left-Right (NLR) traversal
- ❑ Post-order traversal or Left-Right-Root (LRN) traversal

Binary Tree Traversal

In-order traversal Algorithm (LNR):



❖ To traverse a non-empty binary tree using In-order traversal, we need to perform the following operations:

(1) Traverse the left subtree of R in in-order

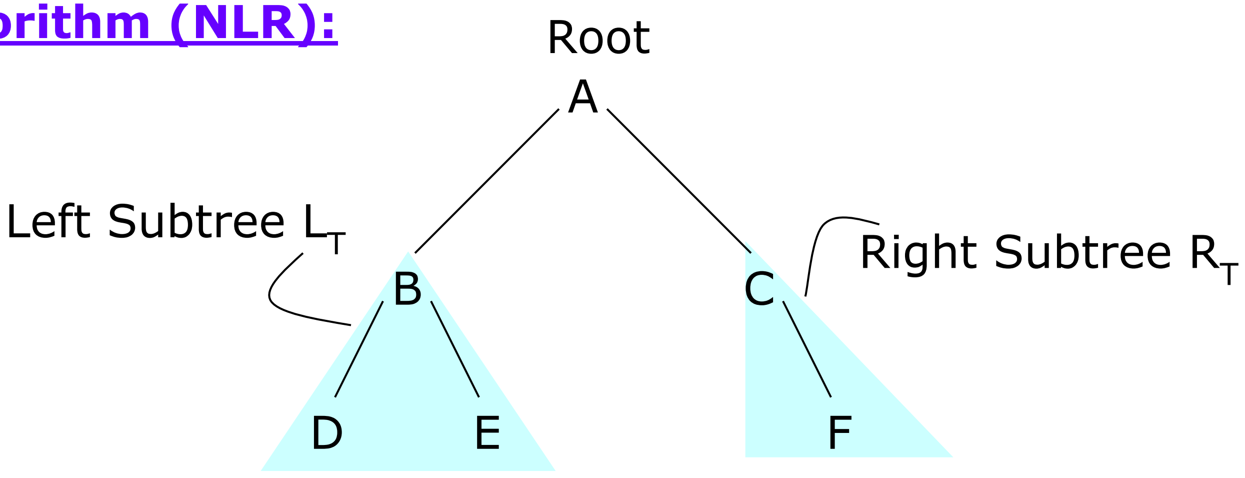
(2) Process the root R

(3) Traverse the right subtree of R in in-order

In-order traversal of the above tree is **DBEACF**

Binary Tree Traversal

Pre-order traversal Algorithm (NLR):



❖ To traverse a non-empty binary tree using Pre-order traversal, we need to perform the following operations:

(1) Process the root R

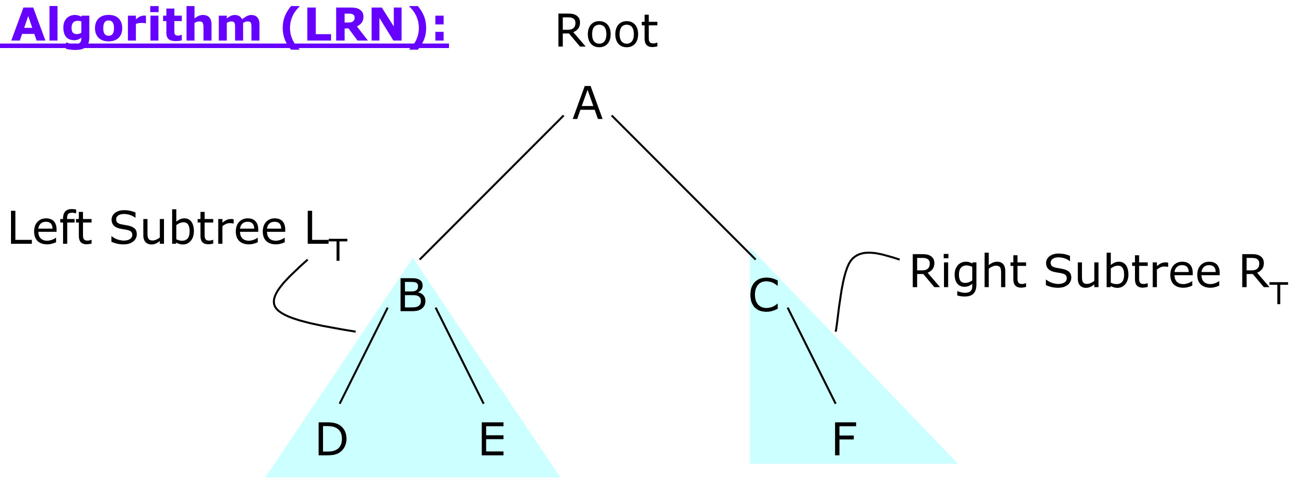
(2) Traverse the left subtree of R in pre-order

(3) Traverse the right subtree of R in pre-order

Pre-order traversal of the above tree is **ABDECF**

Binary Tree Traversal

Post-order traversal Algorithm (LRN):



❖ To traverse a non-empty binary tree using Post-order traversal, we need to perform the following operations:

(1) Traverse the left subtree of R in post-order

(2) Traverse the right subtree of R in post-order

(3) Process the root R

Post-order traversal of the above tree is **DEBFCA**

Binary Tree Traversal

Note:

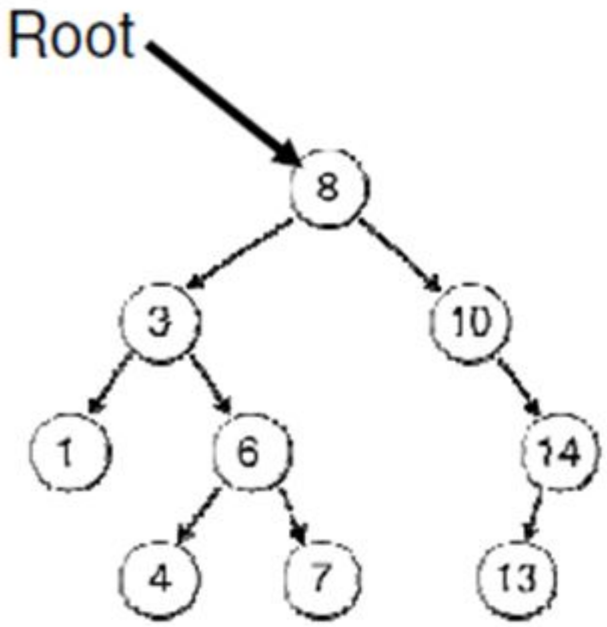
- Each traversal algorithm has the same three steps.
- In each algorithm, the left subtree is always traversed before the right subtree.
- Difference between the algorithms is the time at which the root R is processed.
- Each algorithm is defined recursively.
- When the algorithms are implemented on a computer, a stack will be used.

Binary Tree Traversal

Example-1: Consider the binary tree shown in the figure below. Print the value in each node after an:

(1) In-order traversal (LNR)
(2) Pre-order traversal (NLR)
(3) Post-order traversal (LRN)

Answer:



Inorder Traversal: 1, 3, 4, 6, 7, 8, 10, 13, 14

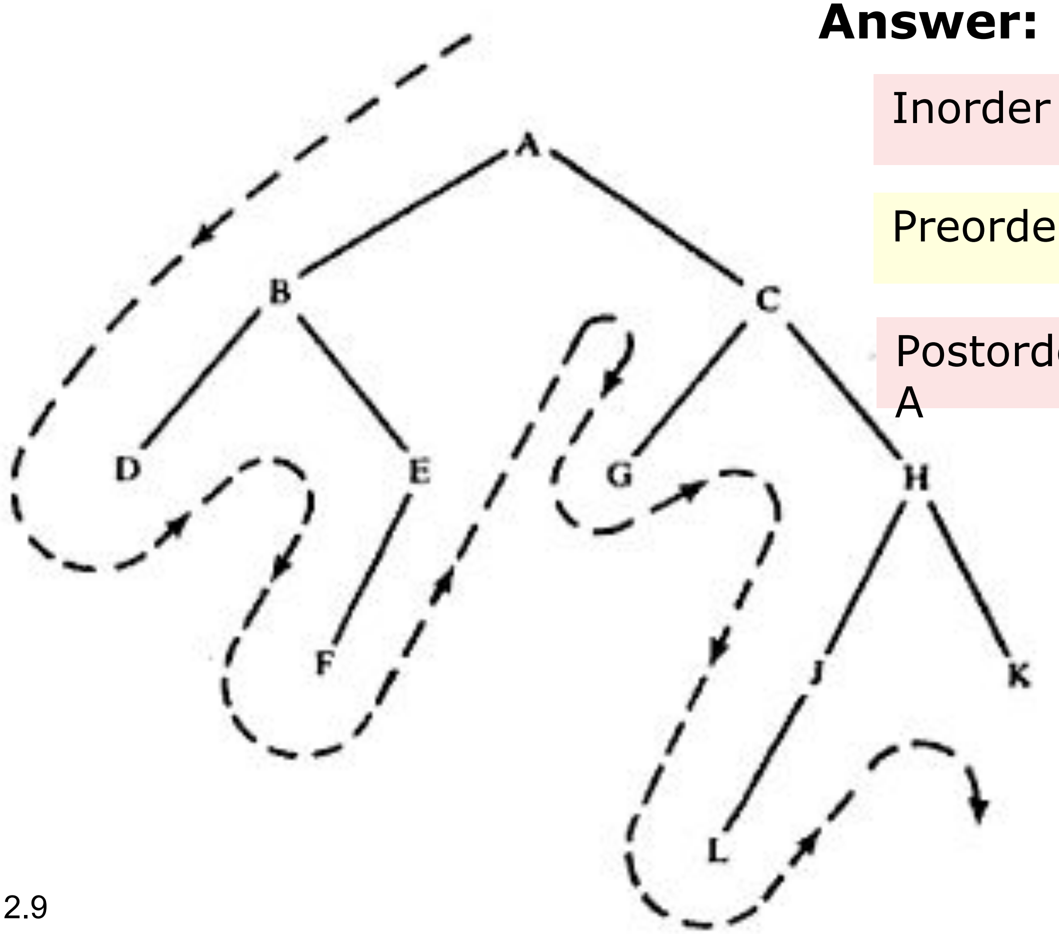
Preorder Traversal: 8, 3, 1, 6, 4, 7, 10, 14, 13

Postorder Traversal: 1, 4, 7, 6, 3, 13, 14, 10, 8

Binary Tree Traversal

Example-2: Consider the binary tree shown in the figure below. Print the value in each node after an:

(1) In-order traversal (LNR)
(2) Pre-order traversal (NLR)
(3) Post-order traversal (LRN)



Answer:

Inorder Traversal: D B F E A G C L J H K

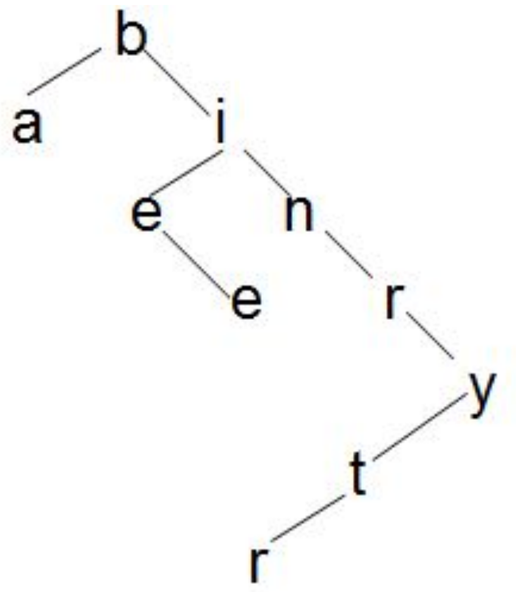
Preorder Traversal: A B D E F C G H J L K

Postorder Traversal: D F E B G L J K H C A

Binary Tree Traversal

Example-3: Consider the binary tree shown in the figure below. Find the traversal of the tree:

(1) In-order traversal (LNR)
(2) Pre-order traversal (NLR)
(3) Post-order traversal (LRN)



Answer:

Inorder Traversal: a b e e i n r r t y

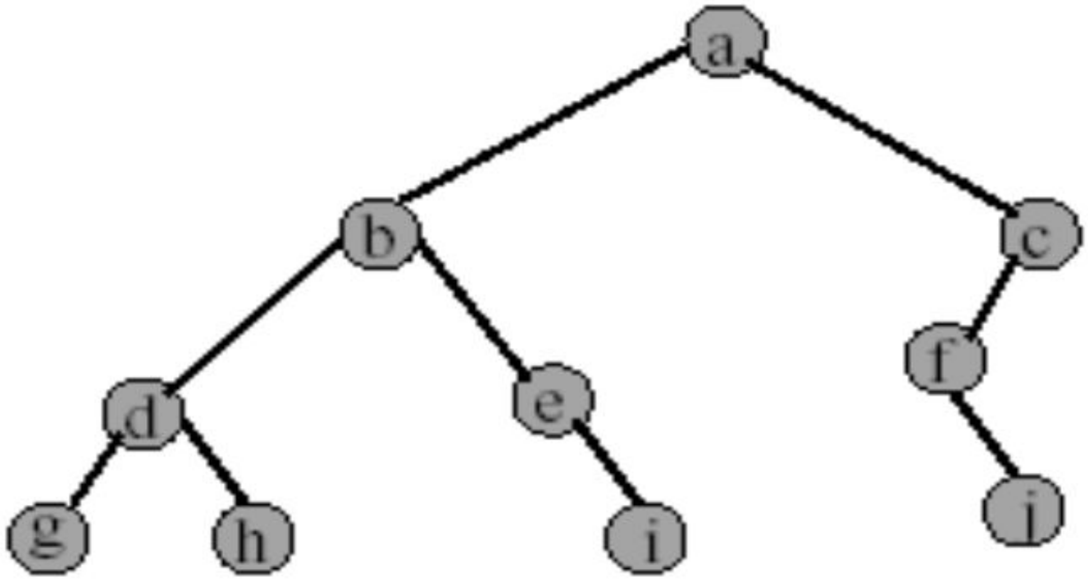
Preorder Traversal: b a i e e n r y t r

Postorder Traversal: a e e r t y r n i b

Binary Tree Traversal

Example-4: Find the following traversals for the binary tree shown below.

(1) In-order traversal
(2) Pre-order traversal
(3) Post-order traversal



Pre-order traversal of the tree is: a b d g h e i c f j

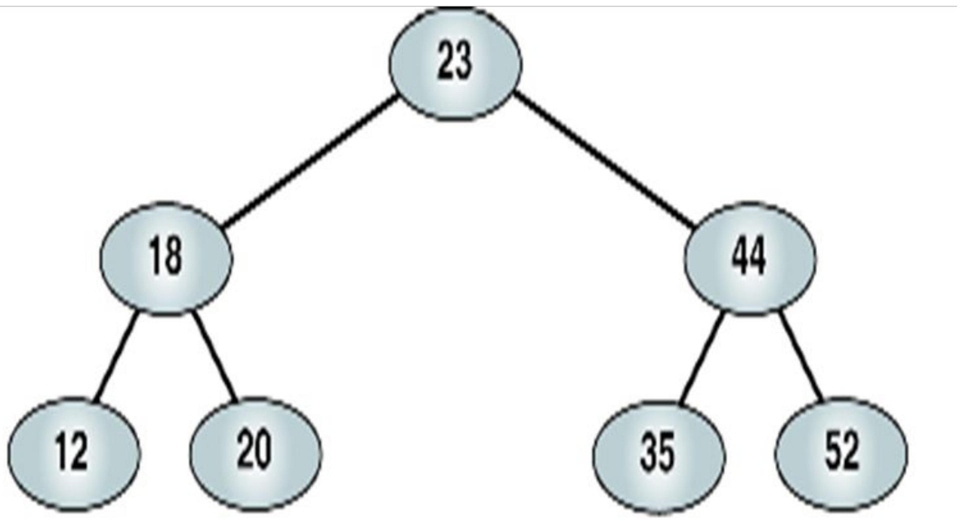
Post-order traversal of the tree is: g h d i e b j f c a

In-order traversal of the tree is: g d h b e i a f j c

Binary Tree Traversal

Example-5: Find the following traversals of the binary tree shown below.

(1) In-order traversal
(2) Pre-order traversal
(3) Post-order traversal



Pre-order traversal of the tree is: **23 18 12 20 44 35 52**

Post-order traversal of the tree is: **12 20 18 35 52 44 23**

In-order traversal of the tree is: **12 18 20 23 35 44 52**

Binary Tree Traversal

Traversal of Binary Tree Made from Algebraic Expression:

❖ In order to motivate this subject, we introduce the concept of Polish notation.

❖ Given a binary operation ʘ, it is customary to represent the result of applying the operation to two elements *a*, *b* by placing the operation symbol in the middle:

a ʘ *b* *[e.g. a + b]*

This is called *infix* notation.

❖ The *Polish or Prefix* notation consists of placing the operator symbol to the left i.e. before the two operands:

ʘ a b *[e.g. + a b]*

❖ The *reverse Polish or Postfix or Suffix* notation consists of placing the operator symbol to the right:

a b ʘ *[e.g. a b +]*

Binary Tree Traversal

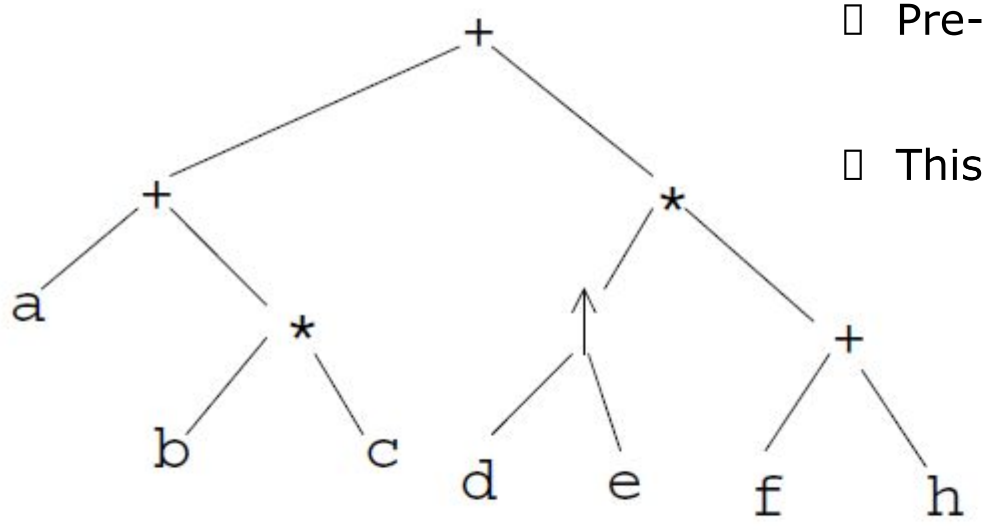
Example-6: Given an algebraic expression E = a + b * c + d ↑ e * (f+ h)

(1) Represent E by means of a binary tree T.

(2) Find the pre-order and post-order traversals of T.

Solution:

 The binary tree for this expression is given in figure below.



 Pre-order traversal of the tree is:

+ + a * b c * ↑ d e + f h

 This is analogous to Polish notation.

❖ Post-order traversal of the tree is

a b c * + d e ↑ f h + * +

❖ This is analogous to Reverse Polish notation.

Figure: Tree for E = a + b * c + d ↑ e * (f+ h)

Traversal Algorithm Using Stack

- Suppose a binary tree T is maintained in memory by some linked representation TREE(INFO, LEFT, RIGHT, ROOT).

Preorder Traversal Algorithm:

- This algorithm uses a variable PTR (pointer) which contains the location of the node N currently being scanned. The algorithm also uses an array STACK, which will hold the addresses of nodes for future processing.
- Initially push NULL onto STACK and then set PTR:=ROOT. Then repeat the following steps until PTR=NULL or, equivalently, while PTR≠NULL.

a) Proceed down the left-most path rooted at PTR, processing each node N on the path and pushing each right child R(N), if any, onto STACK.

❖ The traversing ends after a node N with no left child L(N) is processed.

❖ Thus PTR is updated using PTR:=LEFT[PTR], and the traversing stops when LEFT[PTR]=NULL.

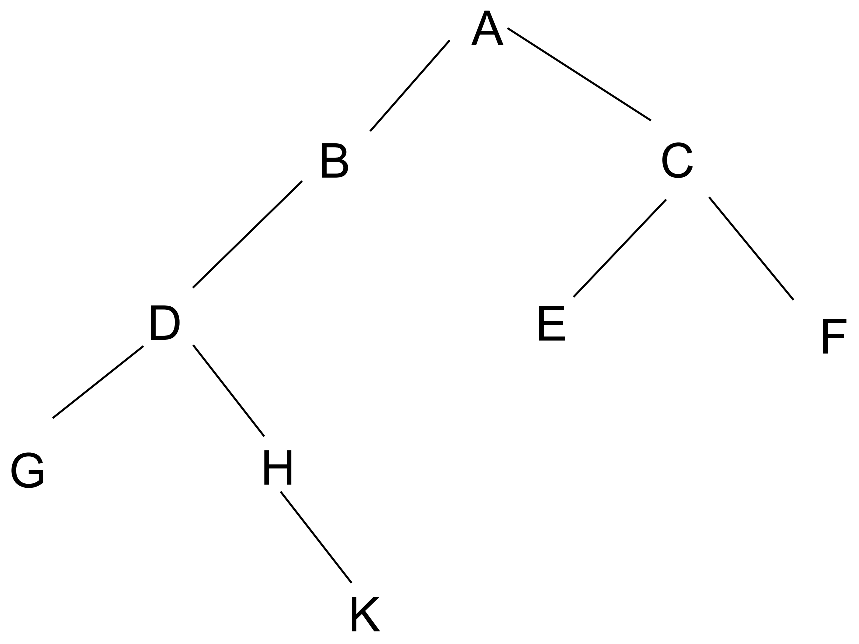
a) [Backtracking] Pop and assign to PTR the top element on STACK. IF PTR≠NULL, then return to Step (a); otherwise exit.n

Traversal Algorithm Using Stack

- We simulate the preorder traversal algorithm with an example.

Example: Preorder Traversal Algorithm

- Consider the binary tree shown in the figure below. Determine the preorder traversal of the tree by simulating the above algorithm.



Traversal Algorithm Using Stack

Solution:

1. Initially push NULL onto STACK:

   STACK: NULL

   Then set PTR:= A, the root of the tree.

2. Proceed down the left-most path rooted at PTR=A as follows:

   (i) Process A and push its right child C onto STACK:

   STACK: NULL, C.

   (ii) Process B (There is no right child).

   (iii) Process D and push its right child H onto STACK:

   STACK: NULL, C, H.

   (iv) Process G (there is no right child).

   No other node is processed, since G has no left child.

3. [Backtracking] Pop the top element H from STACK, and set PTR:=H, This leaves:

   STACK: NULL, C.

   Since PTR≠NULL, return to Step (a) of the algorithm.

Traversal Algorithm Using Stack

4. Proceed down the left-most path rooted at PTR=H as follows:

   (v) Process H and push its right child K onto STACK:

   STACK: NULL, C, K.

   No other node is processed, since H has no left child.

5. [Backtracking] Pop the top element K from STACK, and set PTR:=K, This leaves:

   STACK: NULL, C.

   Since PTR≠NULL, return to step (a) of the algorithm.

6. Proceed down the left-most path rooted at PTR=K as follows:

   (vi) Process K (There is no right child).

   No other node is processed, since K has no left child.

7. [Backtracking] Pop the top element C from STACK, and set PTR:=C, This leaves:

   STACK: NULL.

   Since PTR≠NULL, return to step (a) of the algorithm.

Traversal Algorithm Using Stack

8. Proceed down the left-most path rooted at PTR=C as follows:

    (vii) Process C and push its right child F onto STACK:

        STACK: NULL, F.

    (viii) Process E (There is no right child).

9. [Backtracking] Pop the top element F from STACK, and set PTR:=F, This leaves:

        STACK: NULL.

    Since PTR≠NULL, return to step (a) of the algorithm.

10. Proceed down the left-most path rooted at PTR=F as follows:

    (ix) Process F (There is no right child).

    No other node is processed, since F has no left child.

11. [Backtracking] Pop the top element NULL from STACK, and set PTR:=NULL.

    Since PTR=NULL, the algorithm is completed.

- As seen from Steps 2, 4, 6, 8 and 10, the nodes are processed in the order A, B, D, G, H, K, C, E, F. This is the required preorder traversal of the given tree.

Threaded Representation of Binary Tree/ Threaded Tree

- In linked representation of a binary tree, three important facts are:

(a) the traversal algorithms studied so far spend most of their time manipulating a stack;

(b) the storage space required for the stack is potentially large;

(c) Approximately half of the entries in the pointer fields (LLINK and RLINK) will contain null elements.

- The above loss of space and time may be more efficiently handled by replacing the null entries by special pointers called threads which point to nodes higher in the tree.
- Binary trees with thread pointers are called threaded trees.
- Threaded representation of a binary tree removes the need for a stack, making use of pointer fields which would otherwise have the value nil.
- The most common convention for threaded representation is:

❑ if the left subtree is empty, LLINK points to the in-order predecessor

❑ if the right subtree is empty, RLINK points to the in-order successor.

- These special pointers are known as threads.

Threaded Representation of Binary Tree/ Threaded Tree

- Threads (thread pointers) in a threaded tree must be distinguished in some way from ordinary pointers (LLINK and RLINK).
- In the diagram of a threaded tree, threads are usually indicated by dotted lines and ordinary pointers are indicated by solid lines.
- While representing a threaded tree in the memory of a computer, an extra 1-bit TAG field may be used to distinguish threads from ordinary pointers, or, alternatively, threads may be denoted by negative integers when ordinary pointers are denoted by positive integers.

Threaded Representation of Binary Tree/ Threaded Tree

Types of Threaded Tree/ Ways to Thread a Binary Tree:

- There are many ways to thread a binary tree, but each threading will correspond to a particular traversal of the tree.
- One may choose a one-way threading or a two-way threading.

❑ Single Threaded Tree or One-way Threaded Tree:

❖ Here, a thread will usually appear in the right filed (RLINK) of a node and will point to the next node in the inorder traversal of the tree.

❑ Double threaded Tree or Two-way Threaded Tree:

❖ Here, a thread will appear both in the right field (RLINK) and left filed (LLINK) of a node.

❖ The left field thread will point to the next node in the inorder traversal of the tree and the right field thread will point to the preceding node in the inorder traversal of the tree.

❖ In the inorder traversal of the tree, the left pointer field of the first node and the right pointer field of the last node will contain the null value when the tree does not have a header node. But, if the tree has a header node, they will point to the header node.

Threaded Representation of Binary Tree/ Threaded Tree

Example: Binary Tree to Threaded Tree

- Consider the binary tree shown in the figure below. Find out the in-order traversal of the tree and make the respective threaded binary tree.

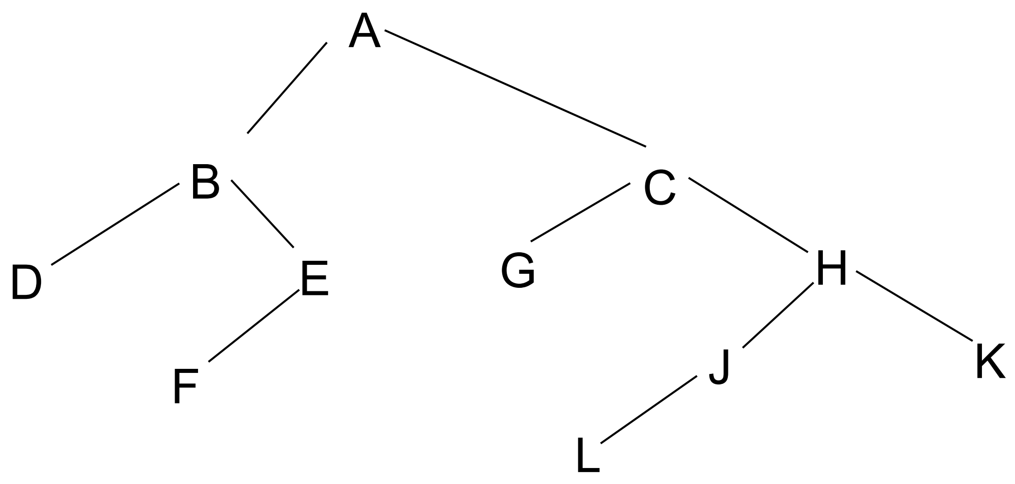


Figure: A binary tree

Threaded Representation of Binary Tree/ Threaded Tree

- One-way inorder thread tree for the given binary tree is shown in the figure below.
- There is a thread from node E to node A, since A is accessed after E in the inorder traversal of the tree.
- Observe that every null right pointer has been replaced by a thread except for the node K, which is the last node in the inorder traversal of the tree.

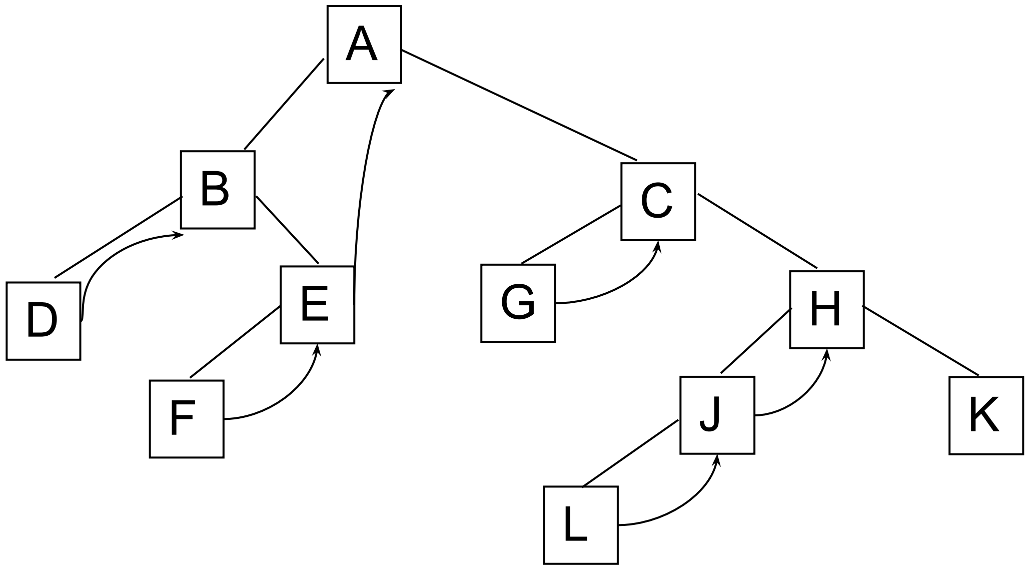


Figure: One-way inorder thread tree

Threaded Representation of Binary Tree/ Threaded Tree

- Two-way inorder threaded tree for the given binary tree is shown in the figure below.
- ❑ There is a left thread from node L to node C, since L is accessed after C in the inorder traversal of the tree.
- ❑ Observe that every null left pointer has been replaced by a thread except for the node D, which is the first node in the inorder traversal of the tree.
- ❑ All the right threads are the same as the one-way threads shown in the previous figure.

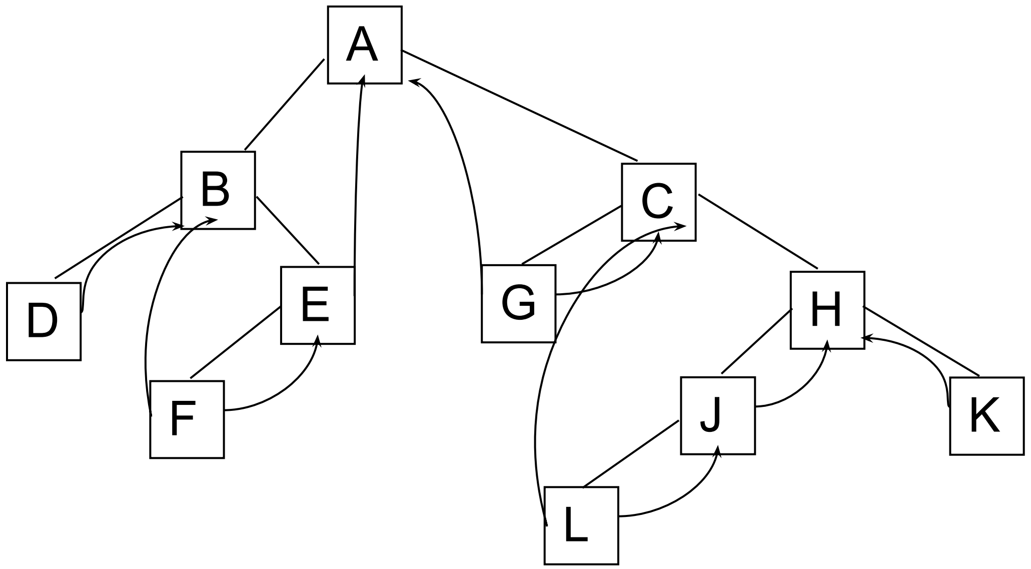


Figure: Two-way inorder threaded tree

Threaded Representation of Binary Tree/ Threaded Tree

- Two-way threaded tree with header node for the given binary tree is shown in the figure below.
- ❑ Here the left thread of D and the right thread of K point to the header node.

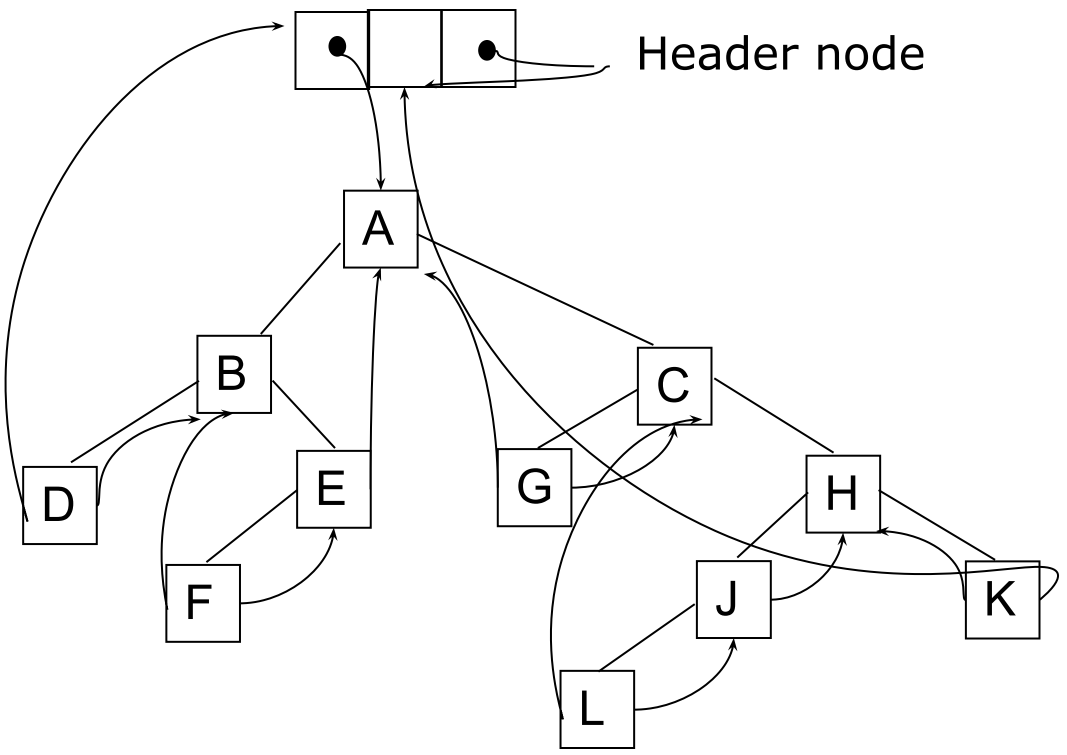


Figure: Two-way threaded tree with header node