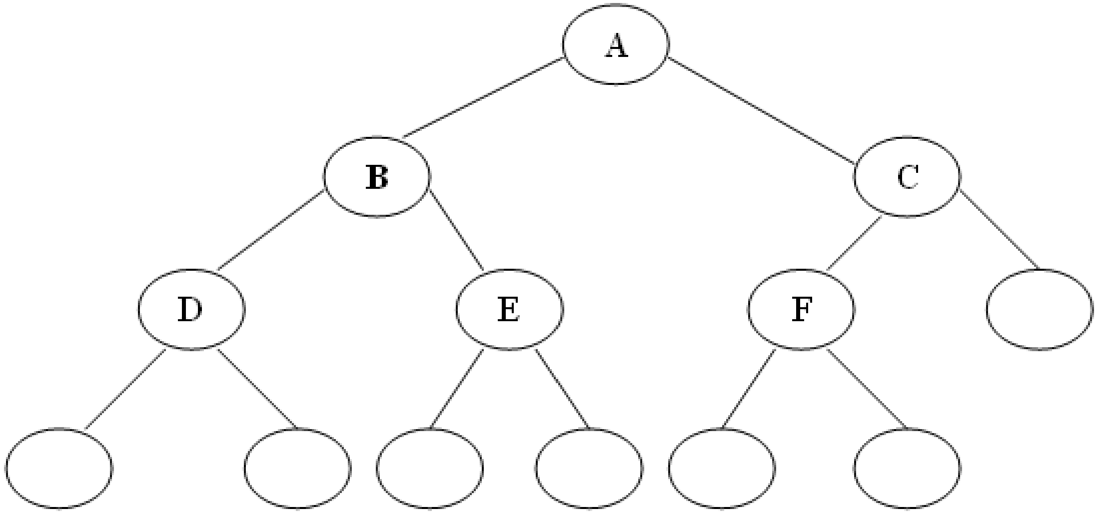
Threaded Binary Tree

Prof. Siddharth Shah

Threaded Binary Tree

- In a linked representation of a binary tree, the number of null links (null pointers) are actually more than non-null pointers. Consider the following binary tree:



A Binary tree with the null pointers

- In above binary tree, there are 7 null pointers & actual 5 pointers. In all there are 12 pointers.
- We can generalize it that for any binary tree with n nodes there will be (n+1) null pointers and 2n total pointers.
- The objective here to make effective use of these null pointers.

Threaded Binary Tree

- A. J. perils & C. Thornton jointly proposed idea to make effective use of these null pointers.
- According to this idea we are going to replace all the null pointers by the appropriate pointer values called threads.
- And binary tree with such pointers are called threaded tree.
- In the memory representation of a threaded binary tree, it is necessary to distinguish between a normal pointer and a thread.
- Therefore we have an alternate node representation for a threaded binary tree which contains five fields as show bellow:

LPTR	LTHREAD	Data	RTHREAD	RPTR

Alternate node structure for a threaded binary tree

LTHREAD = true (1): Denotes left thread link

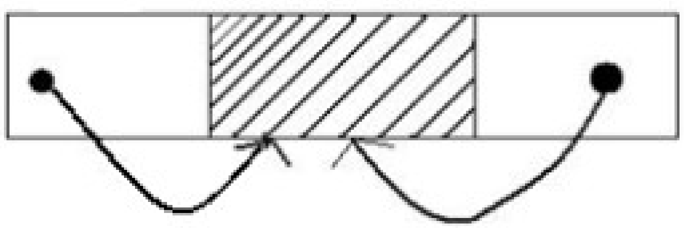
LTHREAD = false (0): Denotes left structural link

RTHREAD = true (1): Denotes right threaded link

RTHREAD = false (0): Denotes right structural link

Threaded Binary Tree

- A binary tree is threaded according to particular traversal order. e.g.: Threads for the inorder traversals of tree are pointers to its higher nodes, for this traversal order.
  - If left link of node P is null, then this link is replaced by the address of its predecessor.
  - If right link of node P is null, then it is replaced by the address of its successor
- Head node is simply another node which serves as the predecessor and successor of first and last tree nodes. Tree is attached to the left branch of the head node

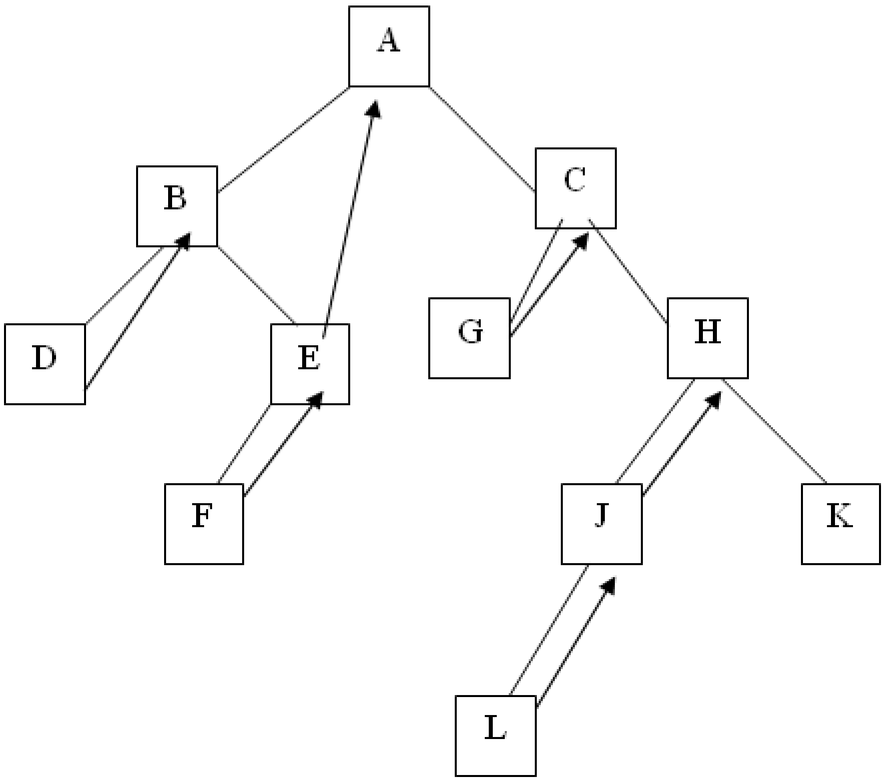


Head

- Also one may choose a one-way threading or a two-way threading.

One-way Threading

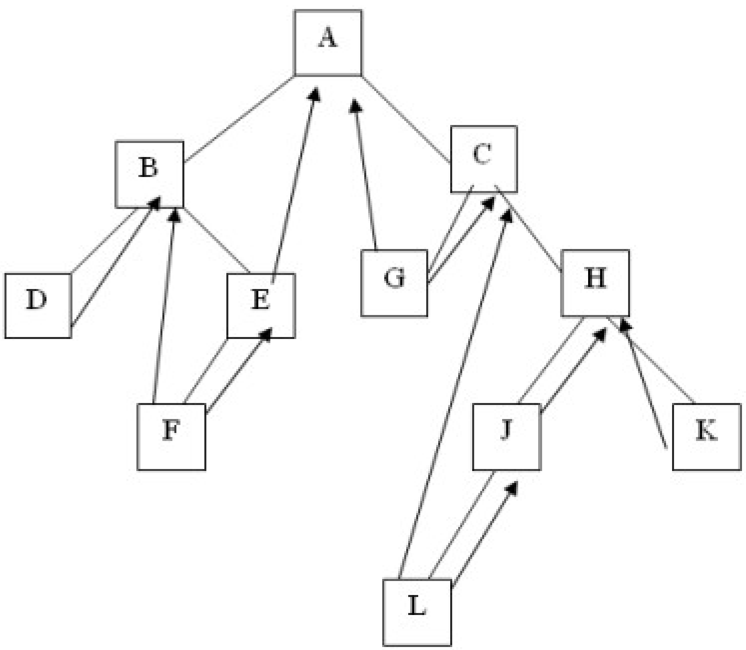
- If threading corresponds to the in order traversal of T, in the one way threading of T,
- a thread will appear in the right field of a node and will point to the successor node (right in-threaded)
- **or**
- a thread will appear in the left field of a node and will point to the predecessor node (left in-threaded) in the inorder traversal of T.



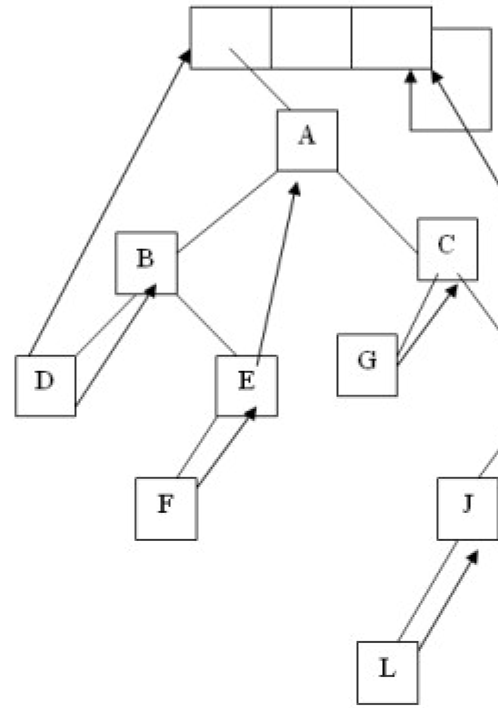
Inorder traversal of above tree is: D,B,F,E,A,G,C,L,J,H,K

Two-way Threading

- Here, a thread in the left field points to the predecessor node and a thread in right field points to a successor node, in the inorder traversal of tree T.
- Furthermore, the left pointer of the first node and the right pointer of the last node (in the inorder traversal of T) will contain the null value when T does not have a header node



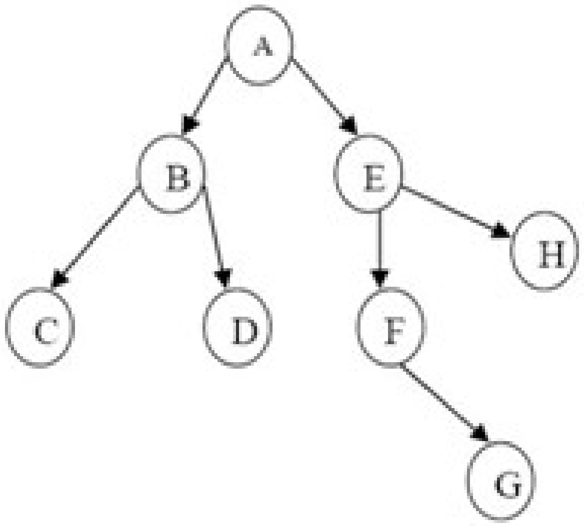
Inorder of above tree is: D,B,F,E,A,G,C,L,J,H,K



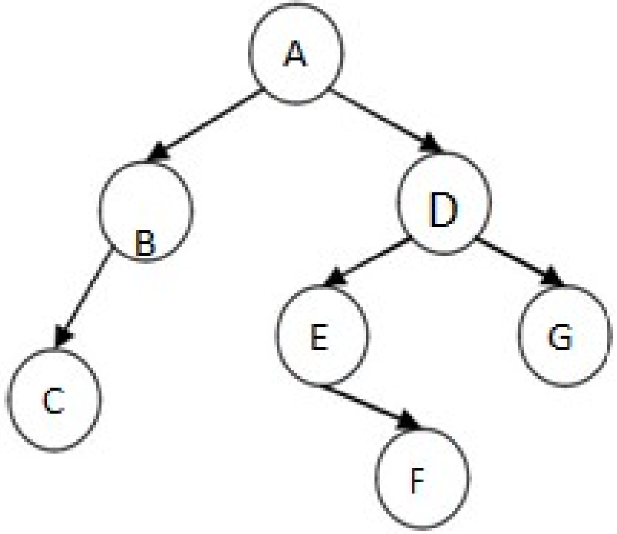
Threaded binary tree T, when T has a head node

Examples

1. Draw a right in-threaded binary tree for the given tree.

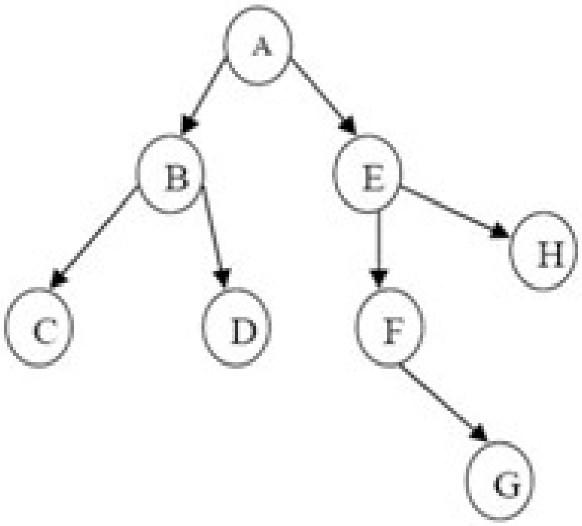


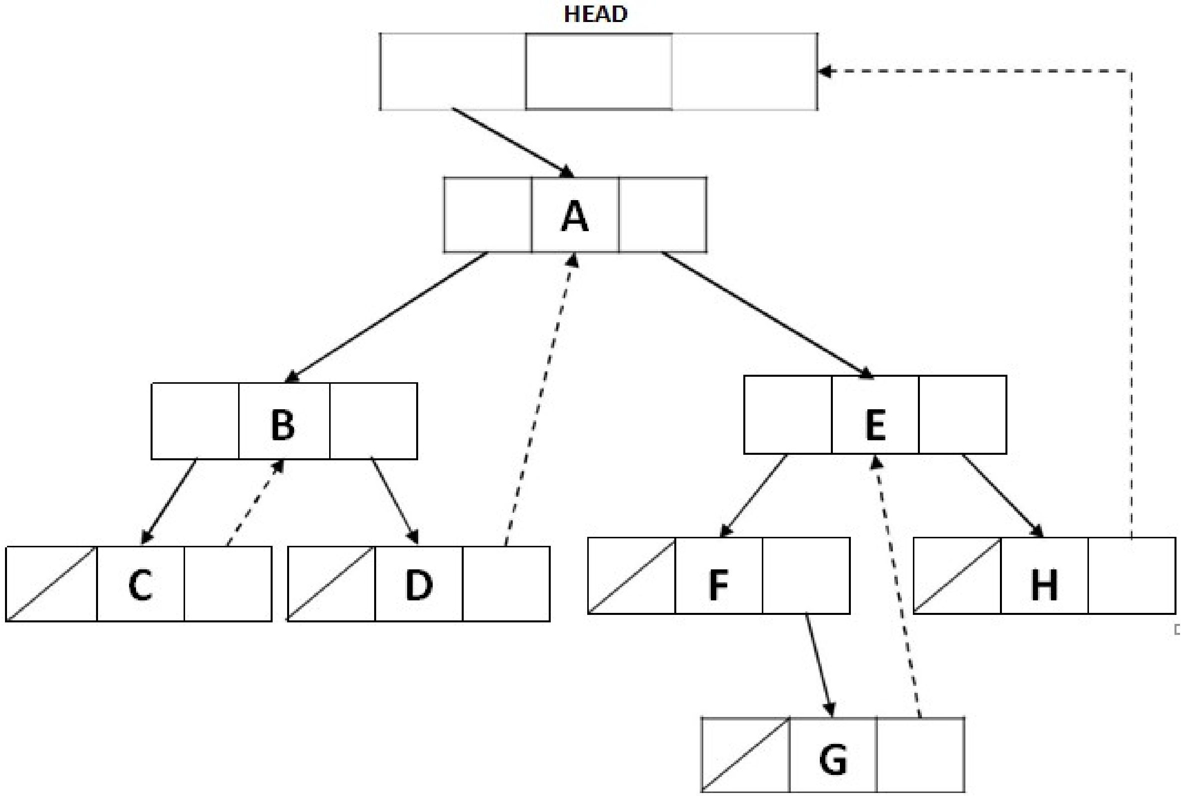
1. Draw a fully in-threaded binary tree for the given tree.



Examples

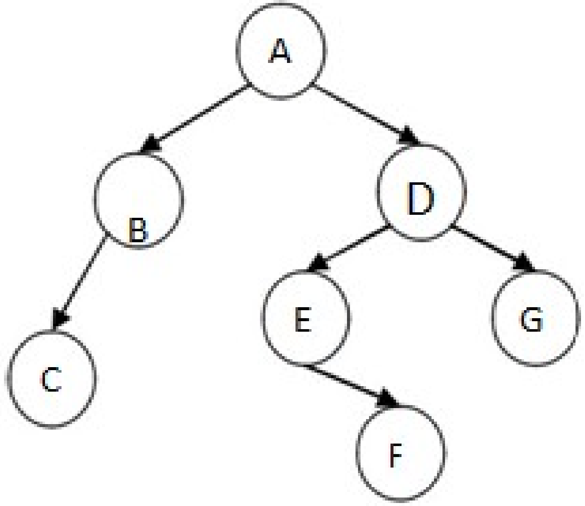
1.Draw a right in-threaded binary tree for the given tree.

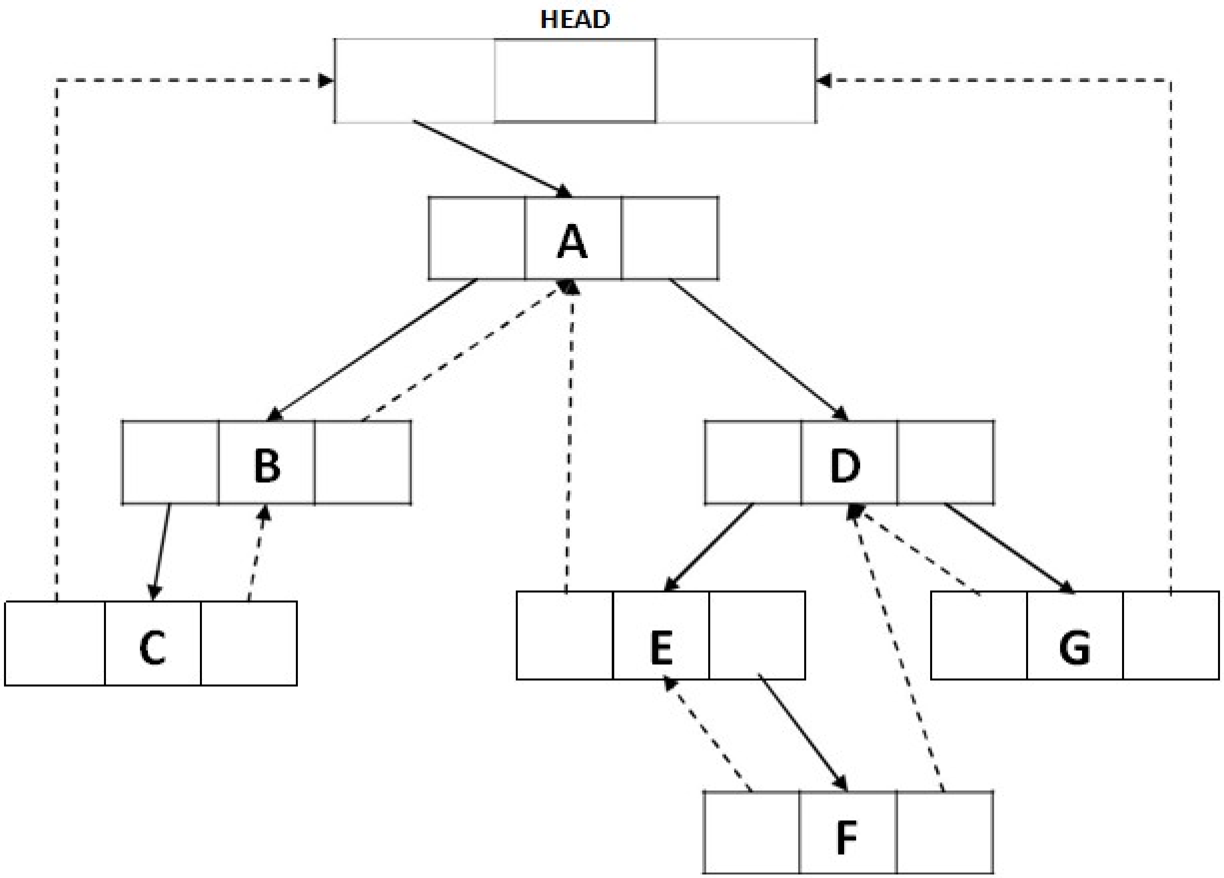




Examples

1.Draw a fully in-threaded binary tree for the given tree.





Threaded Binary Tree

- **Advantages:**
- Inorder traversal is faster than unthreaded version as stack is not required.
- It is possible to generate successor or predecessor of any node without having over head of stack with the help of threading.

- **Disadvantages:**
- Two additional fields are required.
- Insertion into and deletion from threaded binary tree are more time consuming because both thread and structural link must be maintained.

Algorithms for In-Threaded Binary Tree

Procedure: INS(X) :

Given X, the address of a node in a threaded binary tree, this function returns the address of its inorder successor. P is a temporary pointer variable.

1. **[Return the right pointer of the given node if a thread]**

   $P \leftarrow |RPTR(X)|$

   If RPTR (X) < 0

   Then Return (P)

2. **[Branch left repeatedly until a left thread]**

   Repeat while LPTR (P) > 0

   $P \leftarrow LPTR(P)$

3. **[Return address of successor]**

   Return (P)

Algorithms for In-Threaded Binary Tree

Procedure: INP(X) :

Given X, the address of a node in a threaded binary tree, this function returns the address of its inorder predecessor. P is a temporary pointer variable.

1. **[Return the left pointer of the given node if a thread]**

   $P \leftarrow |LPTR(X)|$

   If LPTR (X) < 0

   Then Return (P)

2. **[Branch right repeatedly until a right thread]**

   Repeat while RPTR (P) > 0

   $P \leftarrow$ RPTR (P)

3. **[Return address of predecessor]**

   Return (P)

Algorithms for In-Threaded Binary Tree

Procedure: TINORDER (HEAD) :

Given the address of the list head (HEAD) of a binary tree which has been threaded for inorder traversal and sub algorithm INS previously discussed, this procedure traverses the tree in inorder. P is a temporary pointer variable.

1. **[Initialize]**

   If LPTR(HEAD) = HEAD
   Then Exit
   Else $P \leftarrow HEAD$

2. **[Traverse threaded tree in inorder]**

   Repeat while true
   $P \leftarrow INS(P)$
   If P = HEAD
   Then Exit
   Else Write (DATA(P))

Algorithms for In-Threaded Binary Tree

Procedure: LEFT (X, INFO) :

Given the address of a designated node (X) in an inorder threaded binary tree and the information associated with a new node (INFO), this procedure inserts a new node to the left of the designated node. P is a temporary pointer variable which denotes the address of the node to be inserted.

1. [Create new node]

P <= NODE

DATA(P) ← INFO

2. [Adjust pointer fields]

LPTR(P) ← LPTR(X)

LPTR(X) ← P

RPTR(P) ← -X

3. [Reset predecessor thread if required]

If LPTR(P) > 0

Then RPTR(INP(P)) ← -P

Return