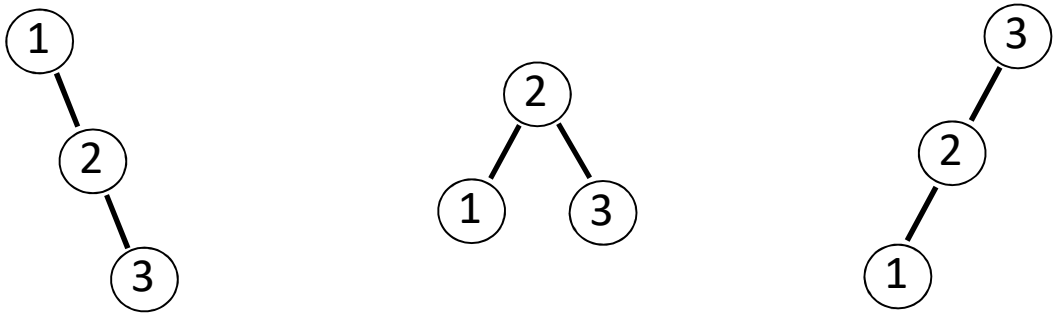
Outline

- Binary tree traversal
- Counting binary trees
- Threaded binary trees

Binary Tree for a Traversal Sequence

- If you are given an <u>inorder</u> sequence, can you define a binary tree uniquely?

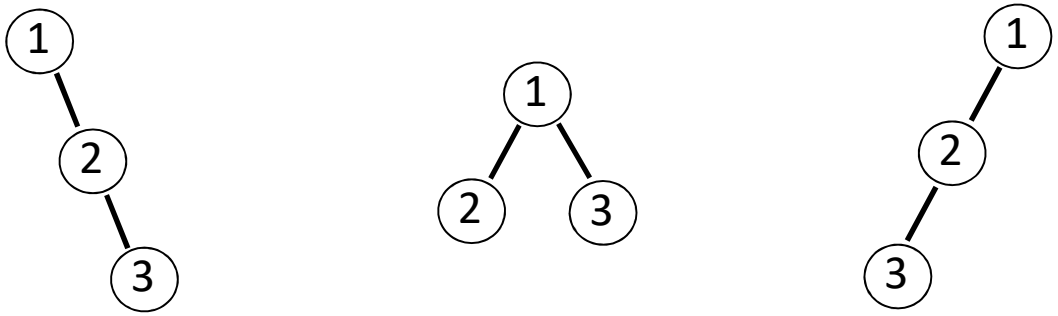
  1 2 3



Binary Tree for a Traversal Sequence

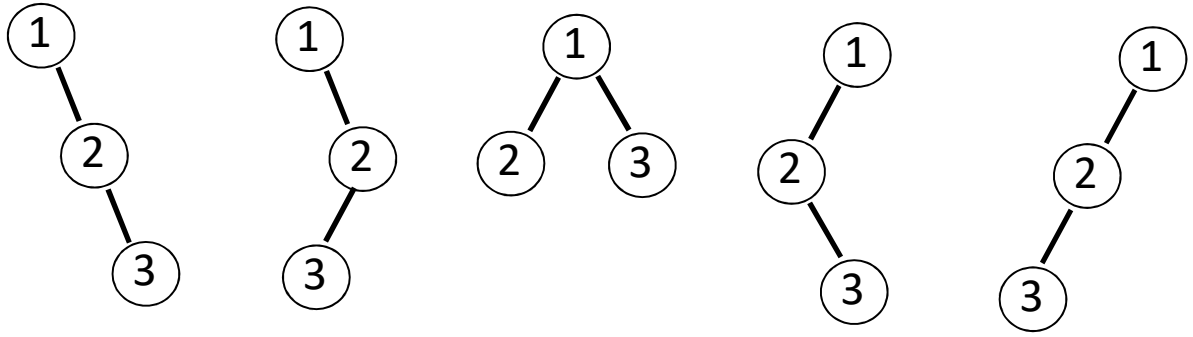
- If you are given a <u>preorder</u> sequence, can you define a binary tree uniquely?

  1 2 3



Counting Binary Trees

- Every binary tree has a unique pair of preorder and inorder sequences



Inorder? 123, 132, 213, 231, 321

Preorder? 123 for all

Binary Tree from Traversal

- Can we reconstruct a binary tree satisfying the given traversal orders?
  - Inorder : C B D A F G E
  - Preorder : A B C D E F G

Binary Tree from Traversal

- Inorder
  - If you pick a letter, all the letters on the left/right are in the left/right subtree of that letter
  - but now we cannot tell which one is the root

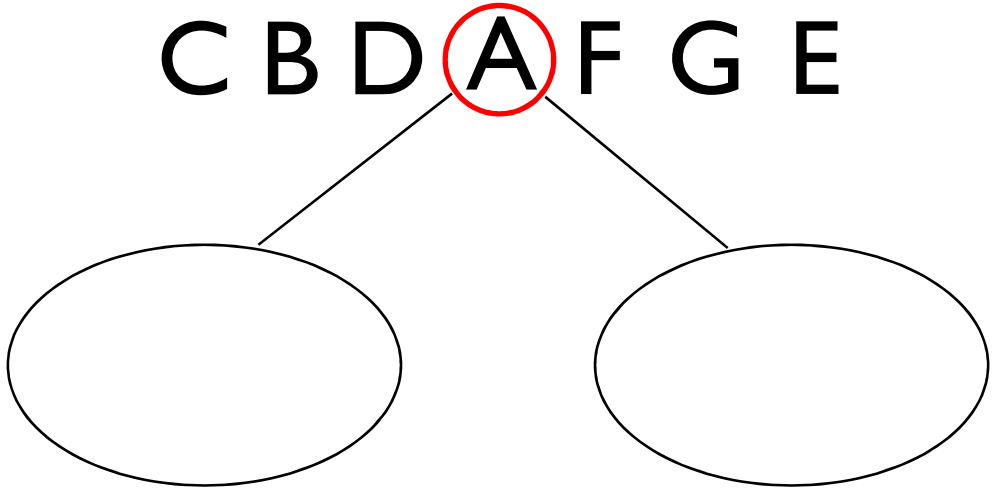
C B D A F G E

Binary Tree from Traversal

- Preorder
  - The front-most letter is the root node

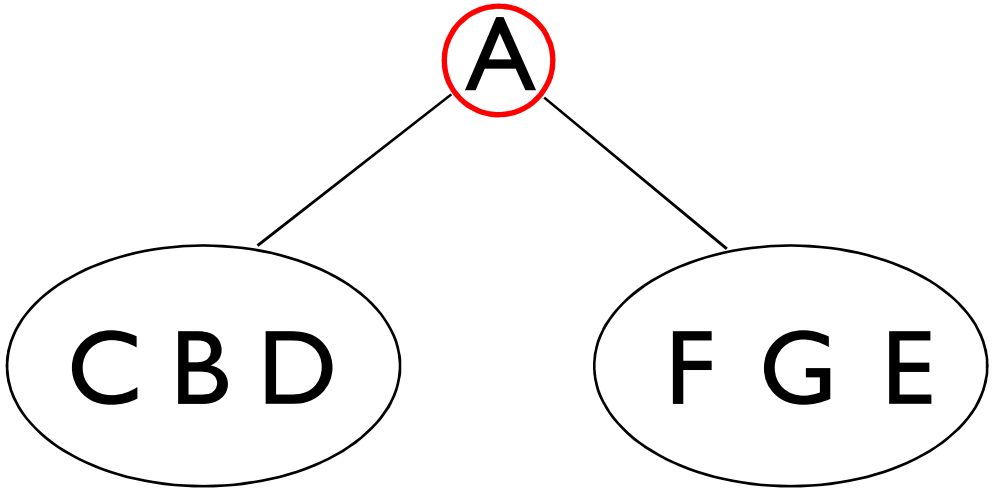
Binary Tree from Traversal

- Inorder
  - Because we now know the root, we can divide into left/right subtree



Binary Tree from Traversal

- Inorder breaks the tree into two subtrees
- Follow-up steps
  - –Preorder visits all left subtree elements
  - –Recursive construct left subtree
  - –Then recursively construct right subtree



A B C D E F G

Define a Unique Binary Tree

- Inorder – postorder
- Inorder – preorder
- Inorder – level order

- Other combination of traversal sequences can not define a unique binary tree

Outline

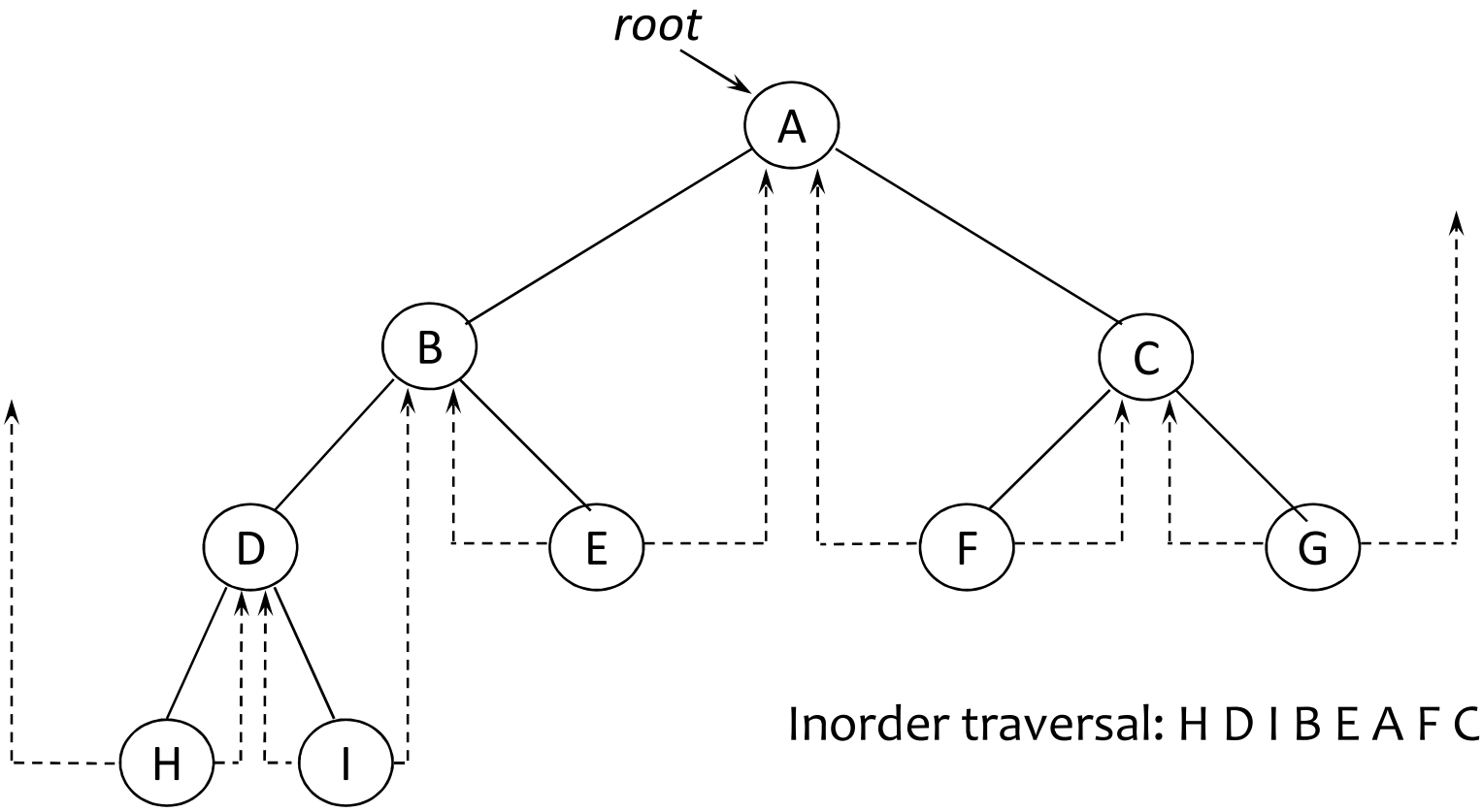
- Binary tree traversal
- Counting binary trees
- Threaded binary trees

Threaded Binary Trees

- Binary tree with n nodes
  - Total # of link pointers : 2n
  - Total # of null links : n+1 -> ~50% are wasted!
- Idea
  - Use null links to represent traversal order
  - Null in rightChild: next node for inorder traversal
  - Null in leftChild: previous node for inorder traversal
  - Benefit: traverse a tree inorder without a stack

Threaded Binary Trees (inorder)

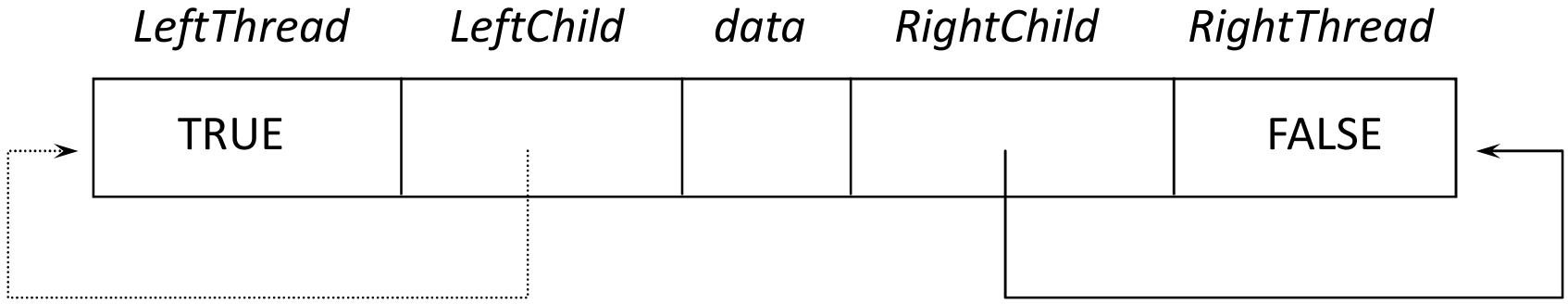
- Dotted line : thread



Inorder traversal: H D I B E A F C G

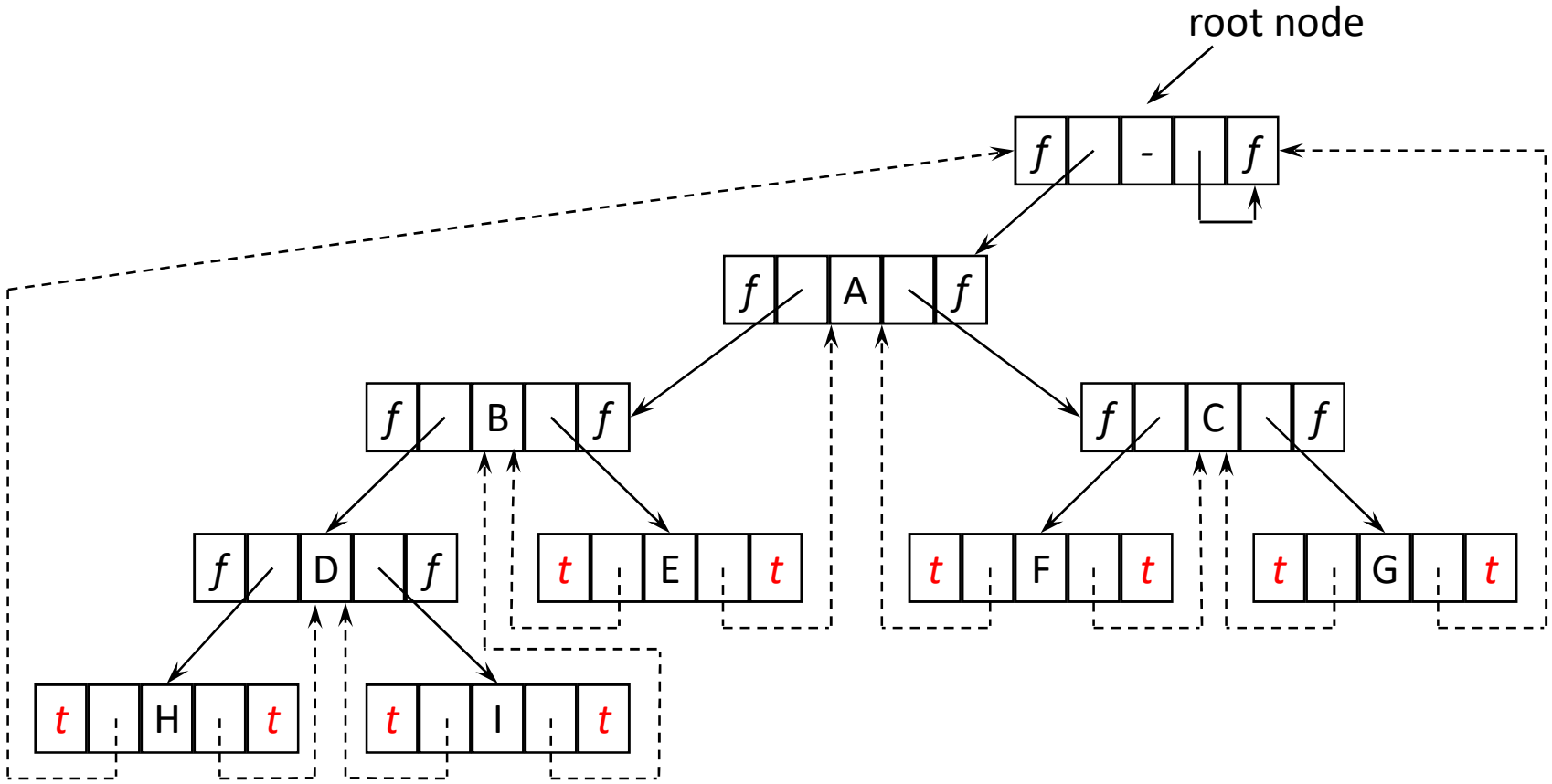
Threaded Binary Trees (inorder)

- Node representation
  - Need to distinguish child / thread pointer
    - One bit Boolean flag for each pointer
- Initializing tree
  - Root node pointing out itself (no null links at all)



Example of empty threaded binary tree

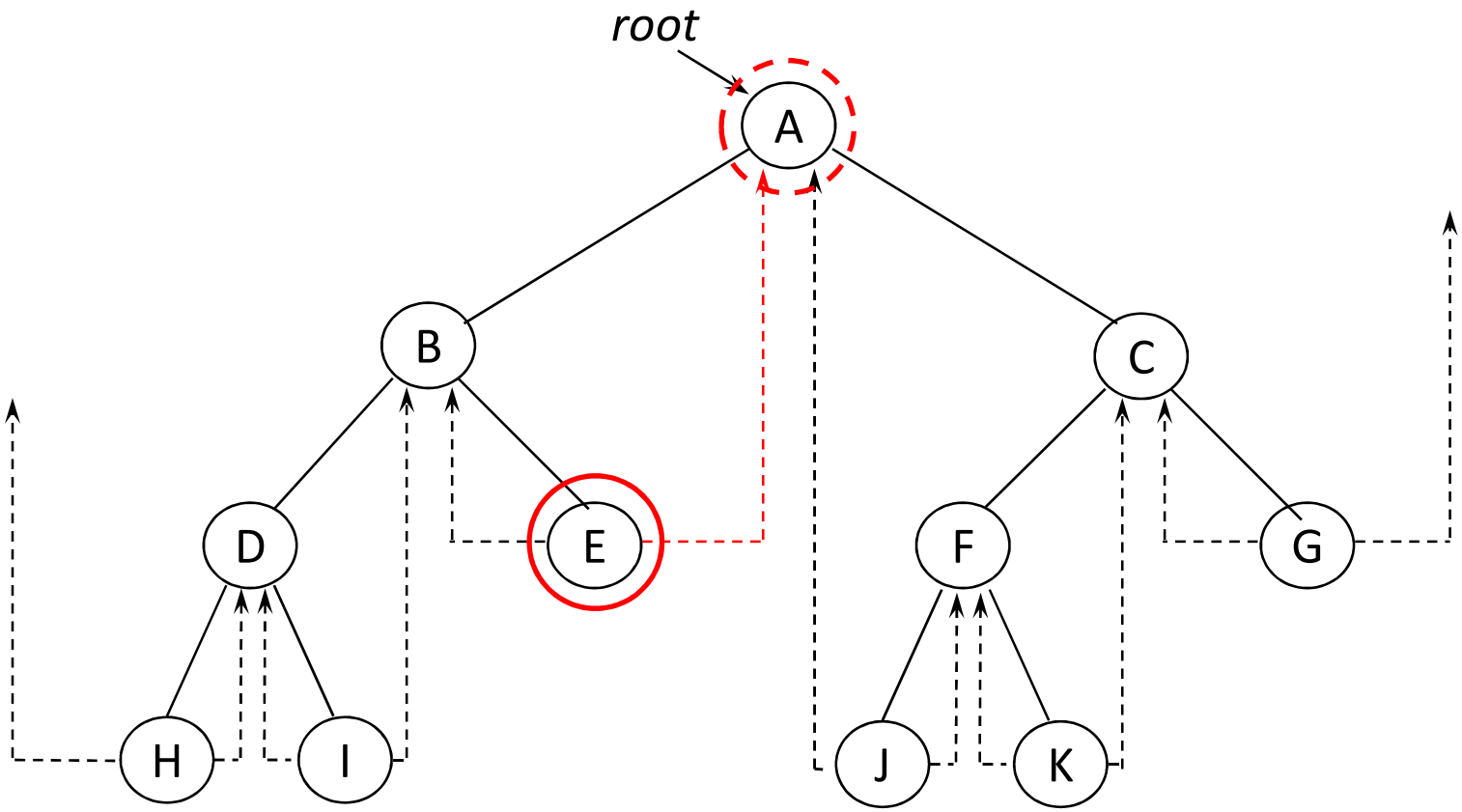
Threaded Binary Trees (inorder)



f = FALSE; t = TRUE

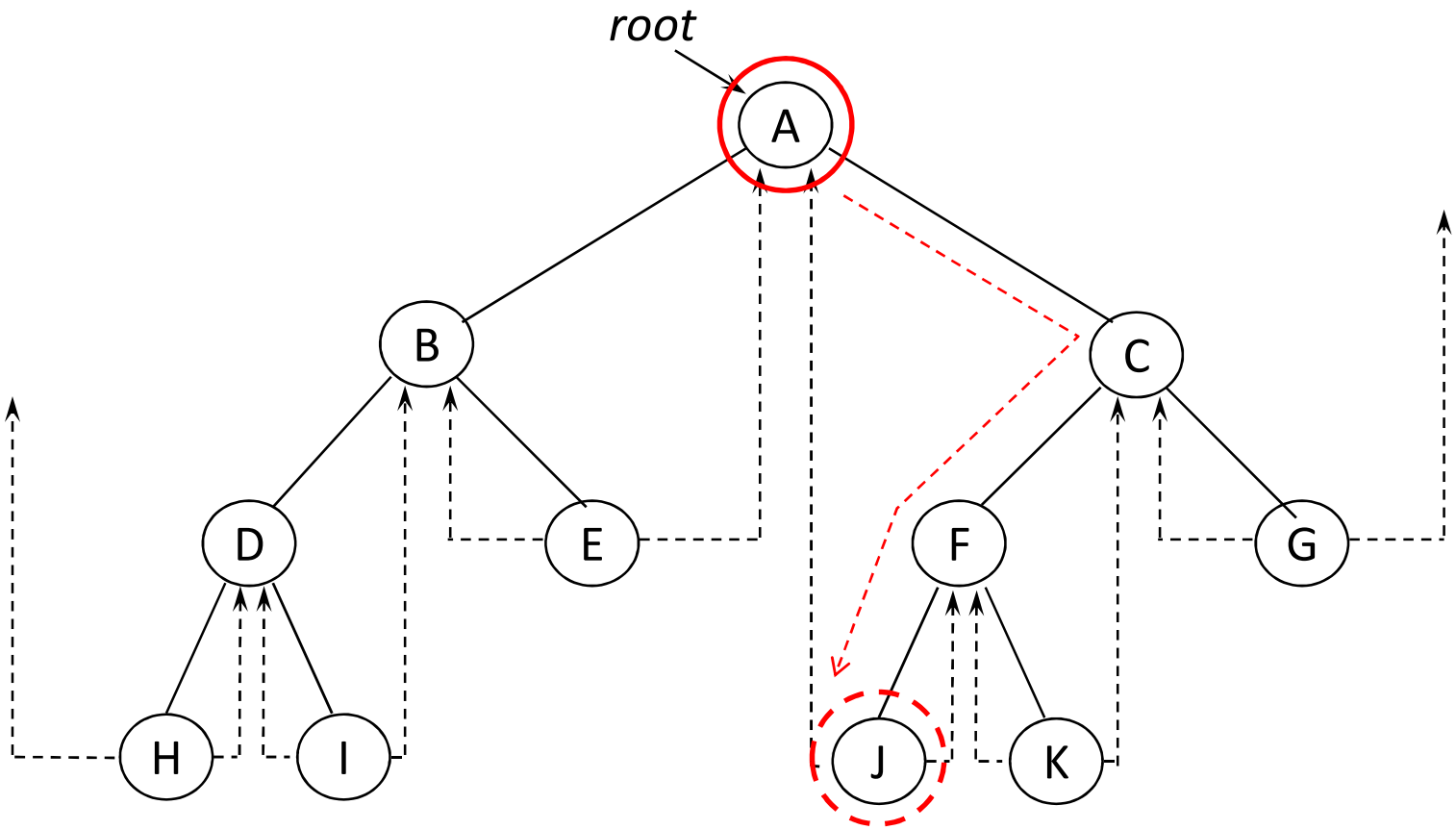
Current node : E

RightThread is True, so the next node for inorder traversal is E->RightChild = A



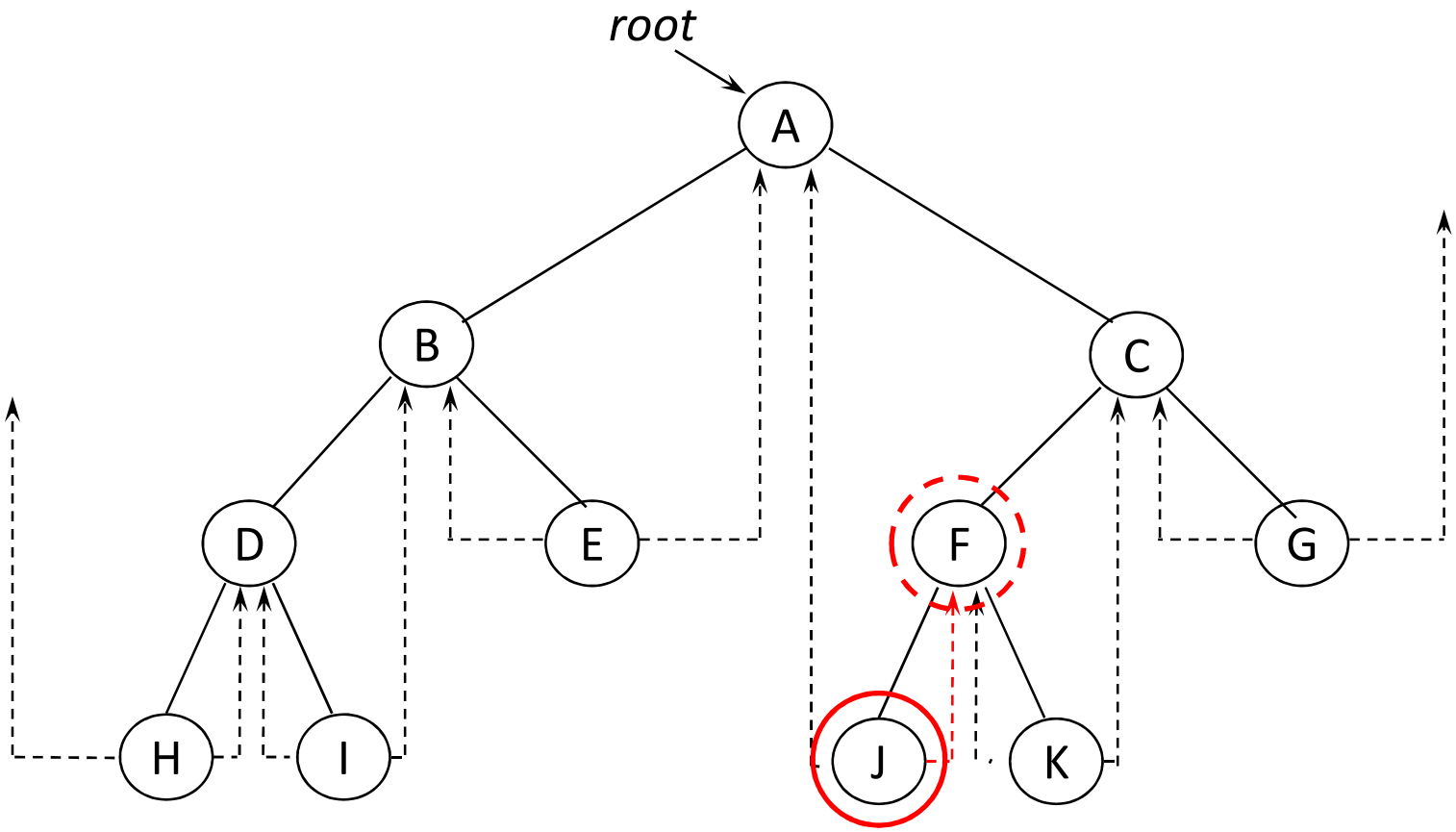
Current node : A

RightThread is False, so you follow the LeftChild link of RightChild node (C) until you reach the node with the LeftThread is True (which is J)



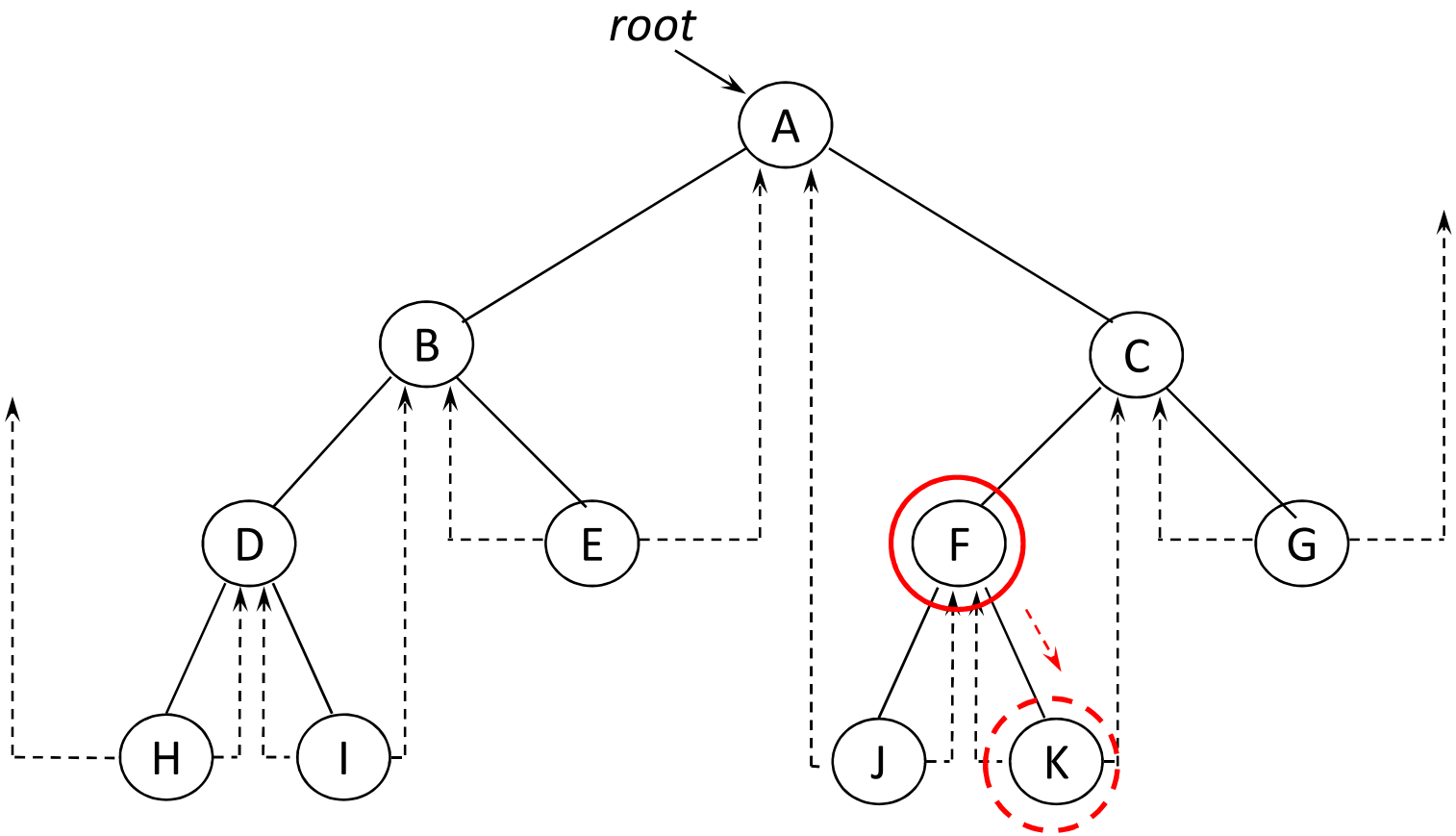
Current node : J

RightThread is True, so the next node for inorder traversal is J->RightChild = F



Current node : F

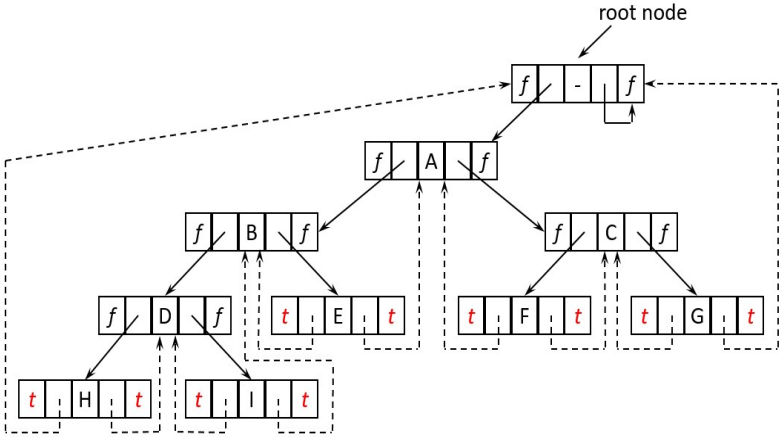
RightThread is False, so you follow the LeftChild link of RightChild node (K) until you reach the node with the LeftThread is True (which is K itself)



Traverse Threaded Binary Tree

- Traverse inorder without using a stack

```
Next() // return next inorder
{
     ThreadedNode *temp = CurrentNode->RightChild;
     if (!CurrentNode->RightThread)
          while (!temp->LeftThread) temp = temp->LeftChild;
     CurrentNode = temp;
}
```



1) Start from root node: RightThread is false, so all the way down to the left
2) If CurrentNode is root node, done with inorder traversal

Questions?

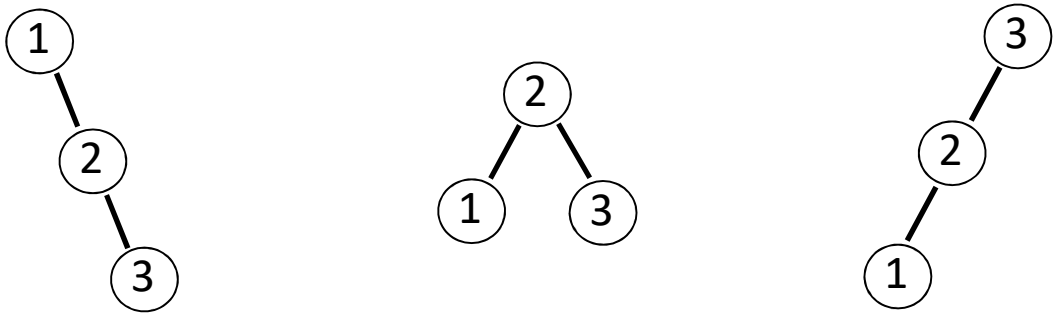
Outline

- Binary tree traversal
- Counting binary trees
- Threaded binary trees

Binary Tree for a Traversal Sequence

- If you are given an <u>inorder</u> sequence, can you define a binary tree uniquely?

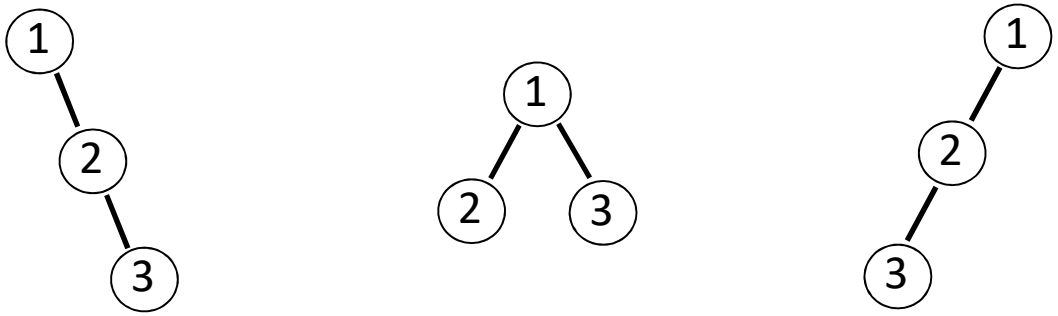
  1 2 3



Binary Tree for a Traversal Sequence

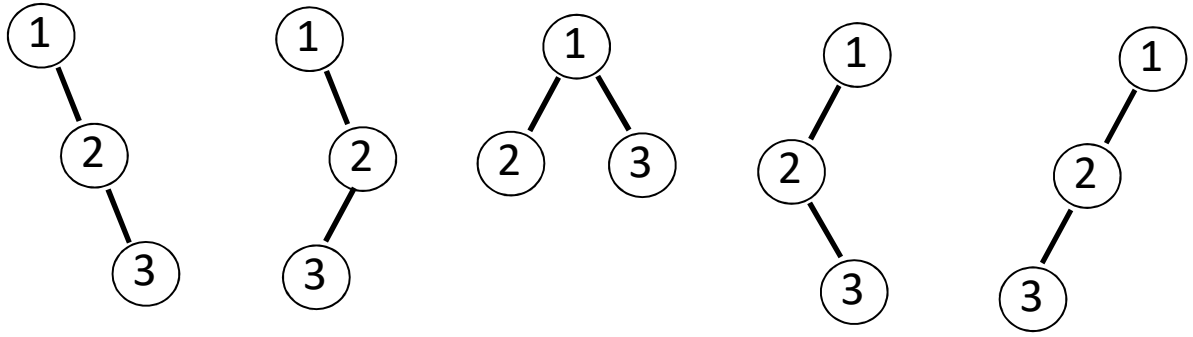
- If you are given a <u>preorder</u> sequence, can you define a binary tree uniquely?

  1 2 3



Counting Binary Trees

- Every binary tree has a unique pair of preorder and inorder sequences



Inorder? 123, 132, 213, 231, 321

Preorder? 123 for all

Binary Tree from Traversal

- Can we reconstruct a binary tree satisfying the given traversal orders?
  - Inorder : C B D A F G E
  - Preorder : A B C D E F G

Binary Tree from Traversal

- Inorder
  - If you pick a letter, all the letters on the left/right are in the left/right subtree of that letter
  - but now we cannot tell which one is the root

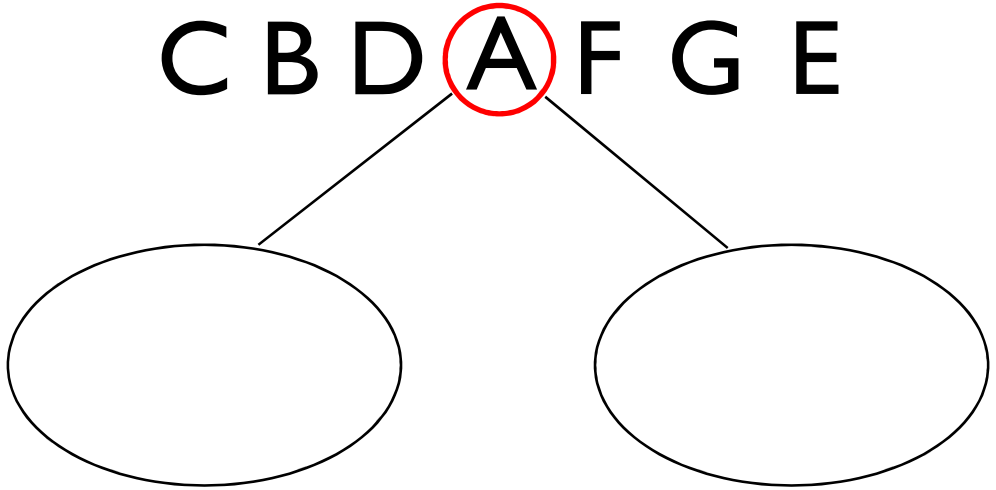
C B D A F G E

Binary Tree from Traversal

- Preorder
  - The front-most letter is the root node

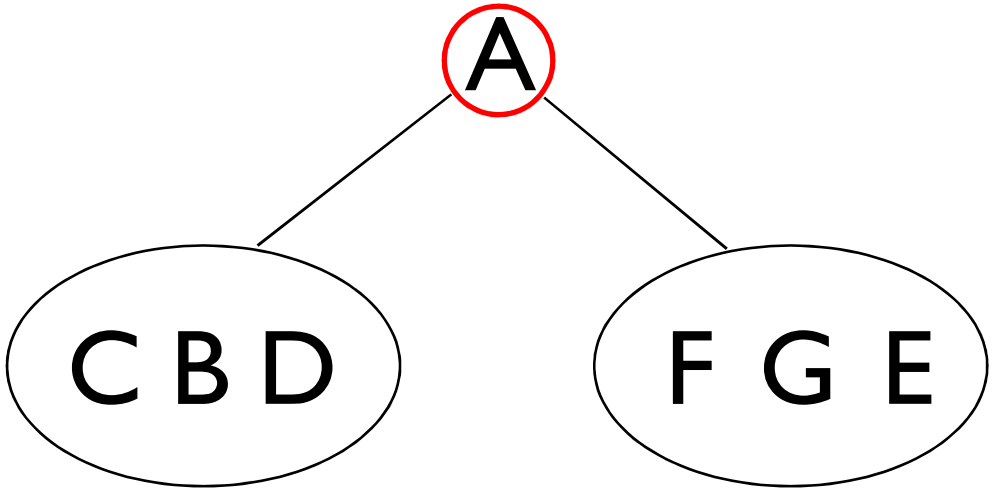
Binary Tree from Traversal

- Inorder
  - Because we now know the root, we can divide into left/right subtree



Binary Tree from Traversal

- Inorder breaks the tree into two subtrees
- Follow-up steps
  - –Preorder visits all left subtree elements
  - –Recursive construct left subtree
  - –Then recursively construct right subtree



A B C D E F G

Define a Unique Binary Tree

- Inorder – postorder
- Inorder – preorder
- Inorder – level order

- Other combination of traversal sequences can not define a unique binary tree

Outline

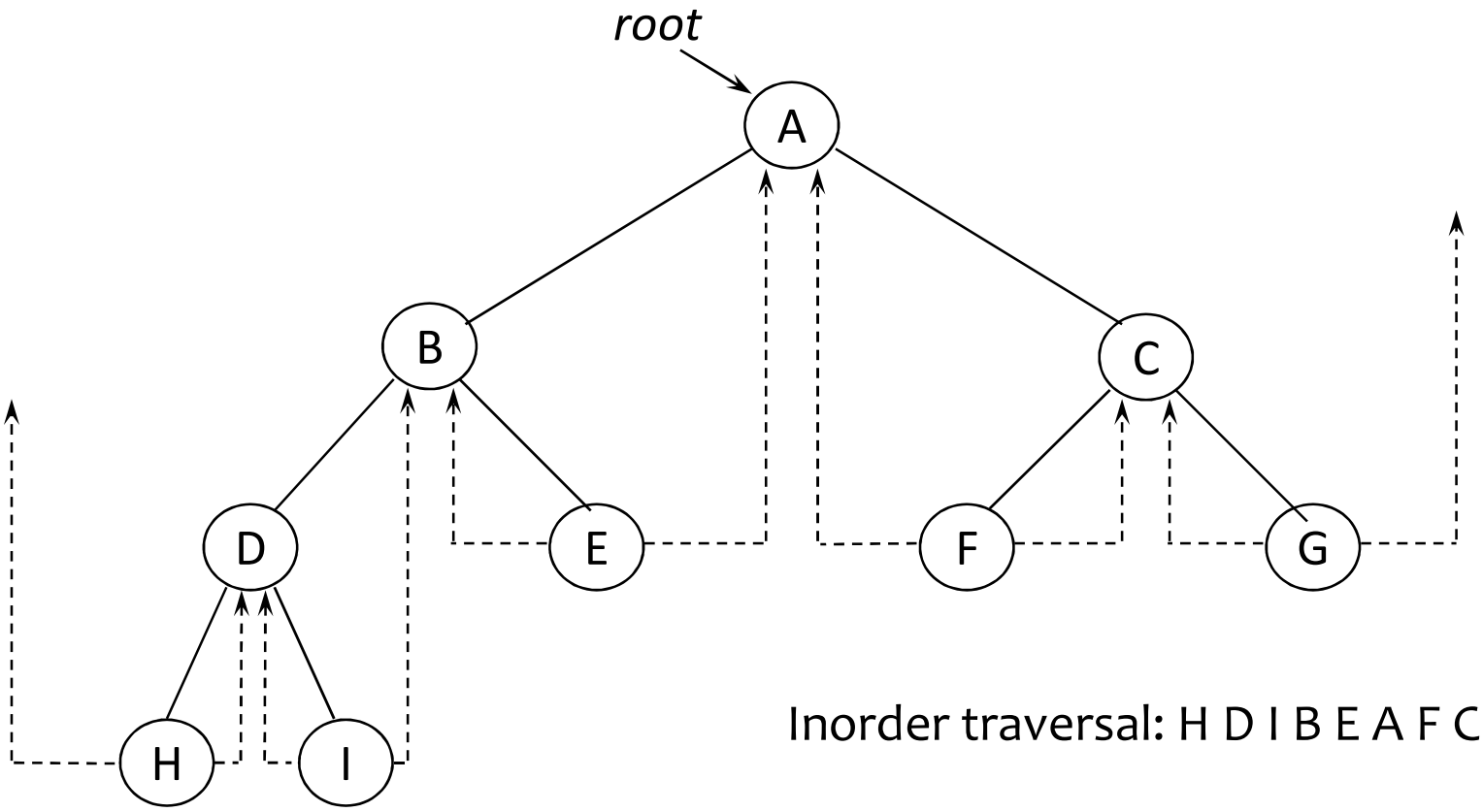
- Binary tree traversal
- Counting binary trees
- Threaded binary trees

Threaded Binary Trees

- Binary tree with n nodes
  - Total # of link pointers : 2n
  - Total # of null links : n+1 -> ~50% are wasted!
- Idea
  - Use null links to represent traversal order
  - Null in rightChild: next node for inorder traversal
  - Null in leftChild: previous node for inorder traversal
  - Benefit: traverse a tree inorder without a stack

Threaded Binary Trees (inorder)

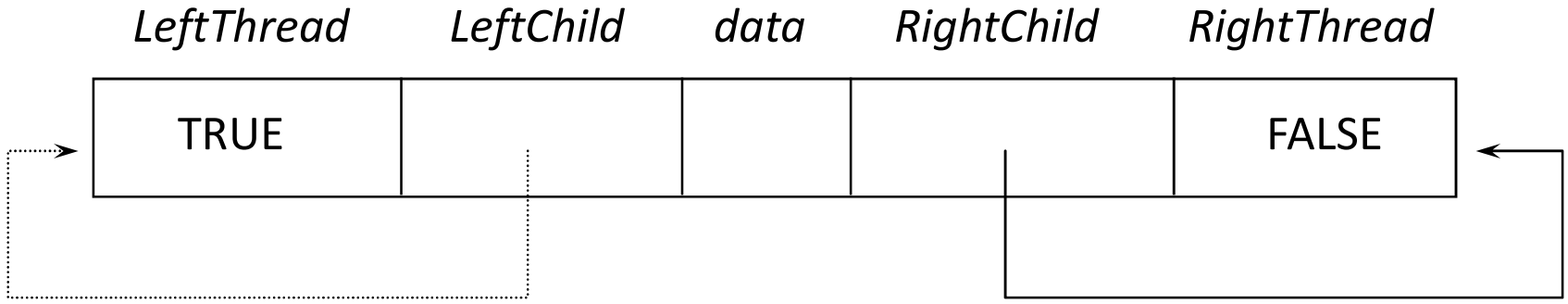
- Dotted line : thread



Inorder traversal: H D I B E A F C G

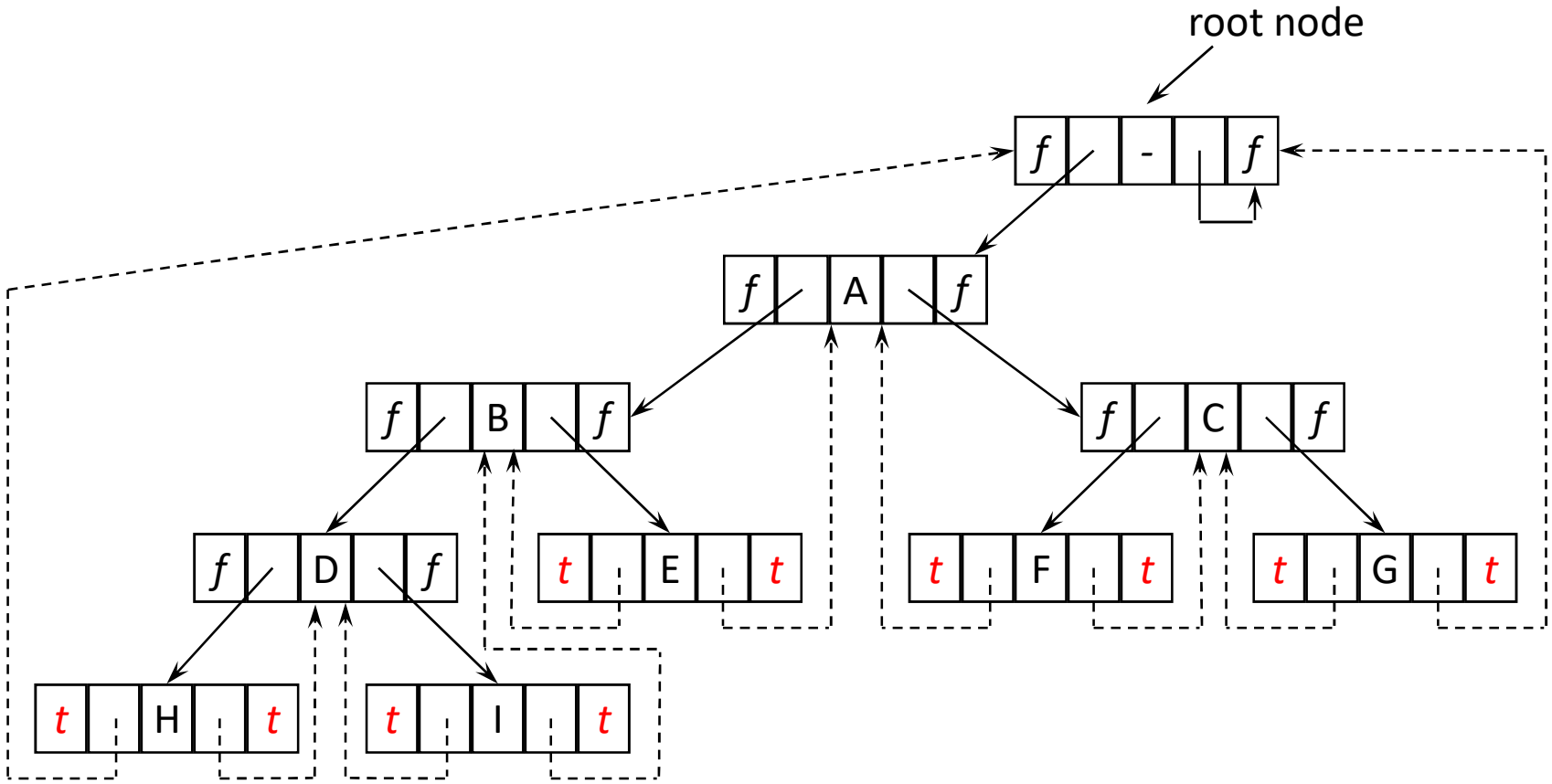
Threaded Binary Trees (inorder)

- Node representation
  - Need to distinguish child / thread pointer
    - One bit Boolean flag for each pointer
- Initializing tree
  - Root node pointing out itself (no null links at all)



Example of empty threaded binary tree

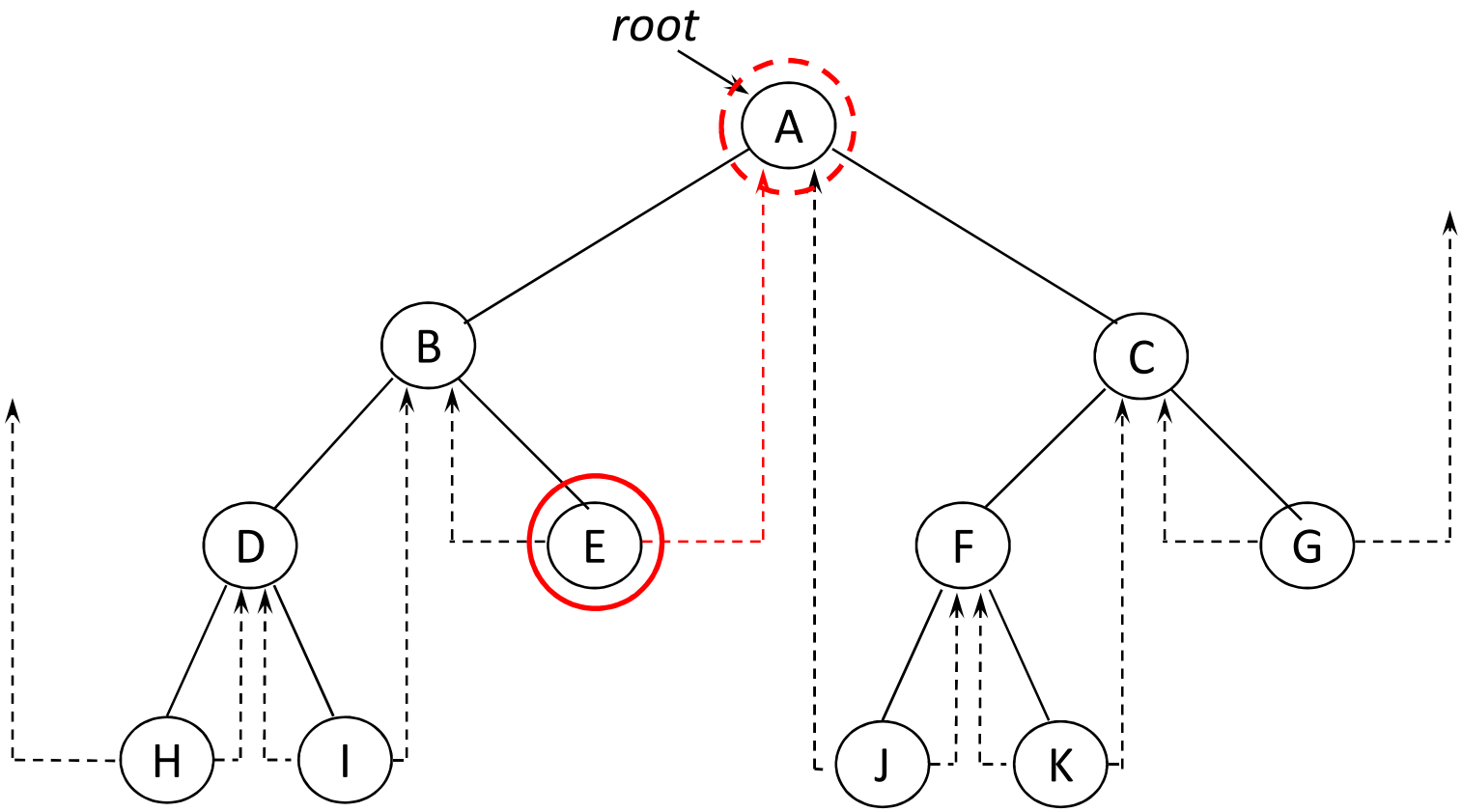
Threaded Binary Trees (inorder)



f = FALSE; t = TRUE

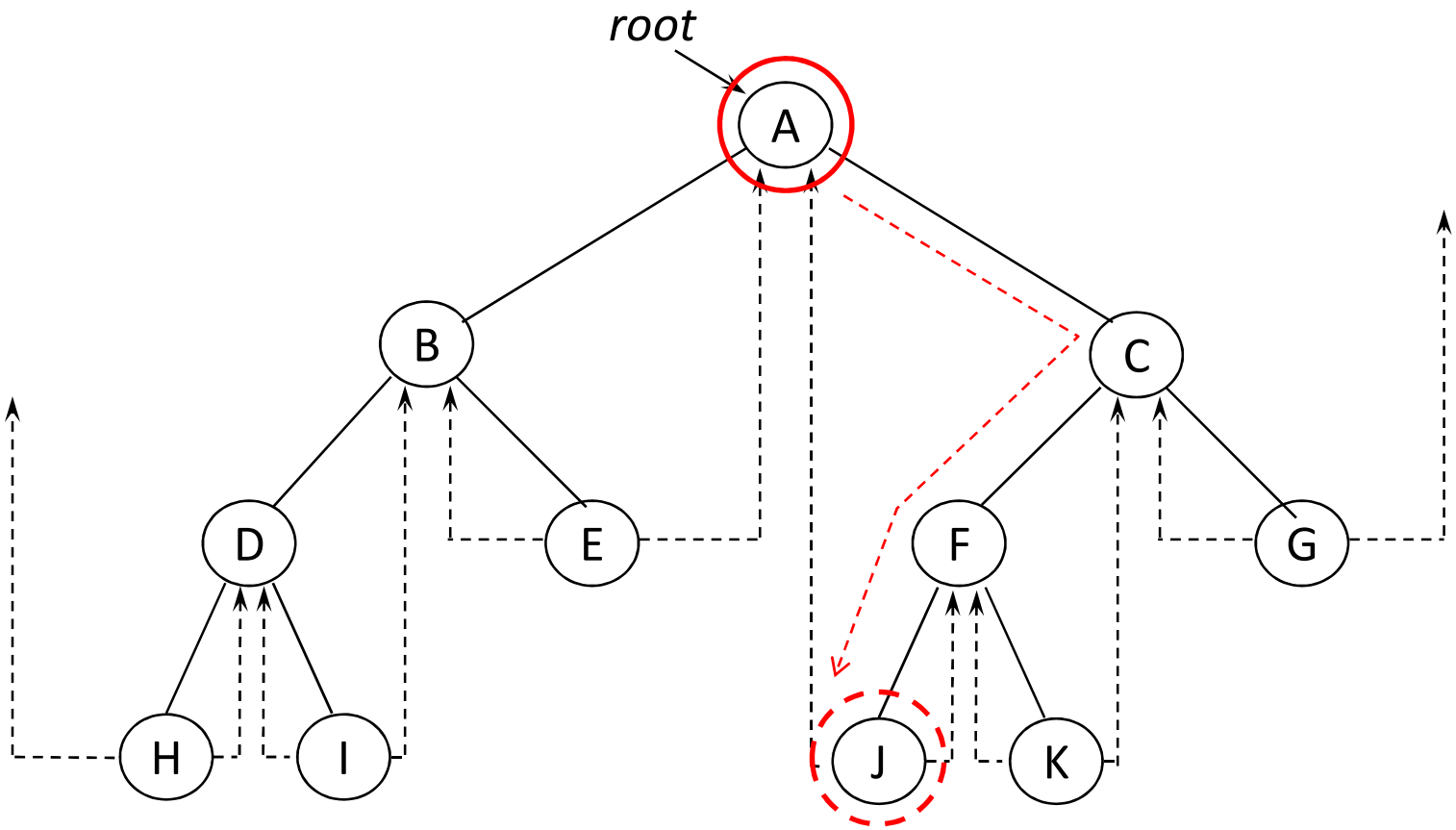
Current node : E

RightThread is True, so the next node for inorder traversal is E->RightChild = A



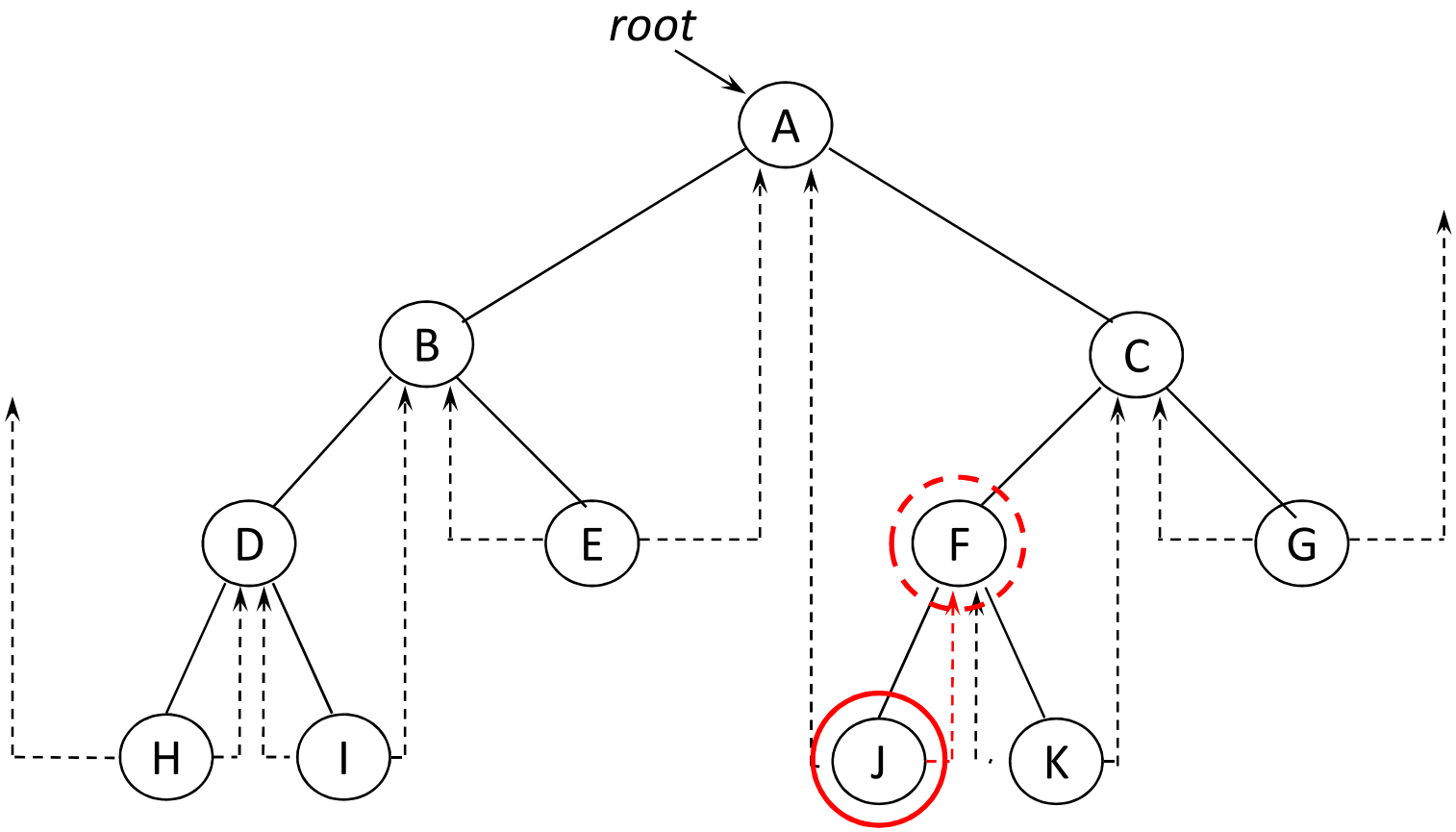
Current node : A

RightThread is False, so you follow the LeftChild link of RightChild node (C) until you reach the node with the LeftThread is True (which is J)



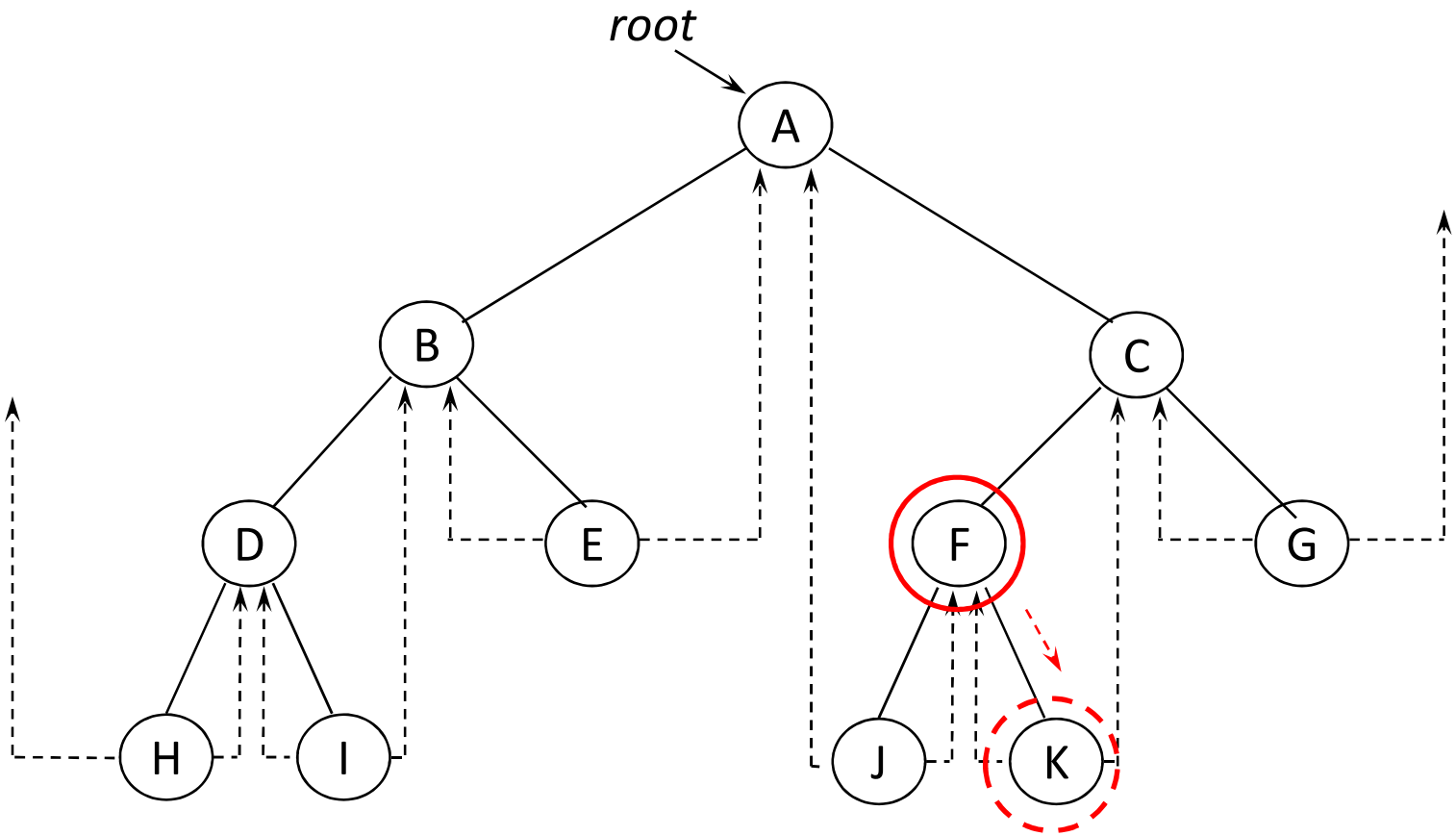
Current node : J

RightThread is True, so the next node for inorder traversal is J->RightChild = F



Current node : F

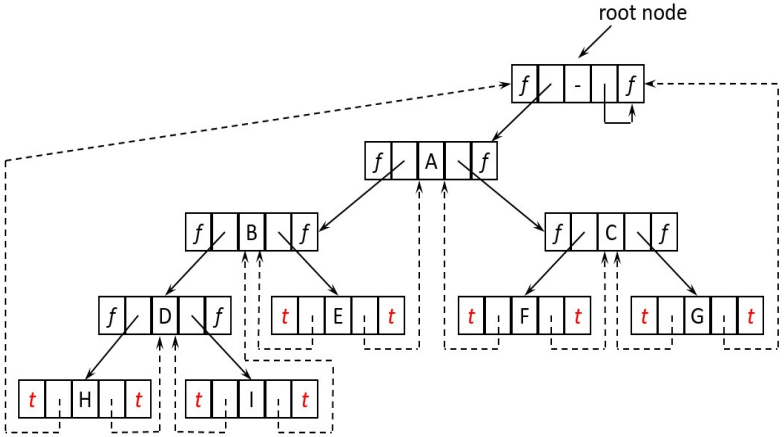
RightThread is False, so you follow the LeftChild link of RightChild node (K) until you reach the node with the LeftThread is True (which is K itself)



Traverse Threaded Binary Tree

- Traverse inorder without using a stack

```
Next() // return next inorder
{
     ThreadedNode *temp = CurrentNode->RightChild;
     if (!CurrentNode->RightThread)
          while (!temp->LeftThread) temp = temp->LeftChild;
     CurrentNode = temp;
}
```



1) Start from root node: RightThread is false, so all the way down to the left
2) If CurrentNode is root node, done with inorder traversal

Questions?

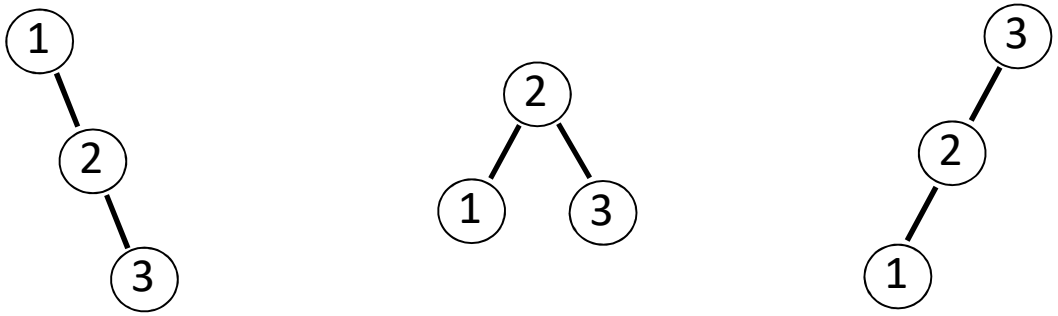
Outline

- Binary tree traversal
- Counting binary trees
- Threaded binary trees

Binary Tree for a Traversal Sequence

- If you are given an <u>inorder</u> sequence, can you define a binary tree uniquely?

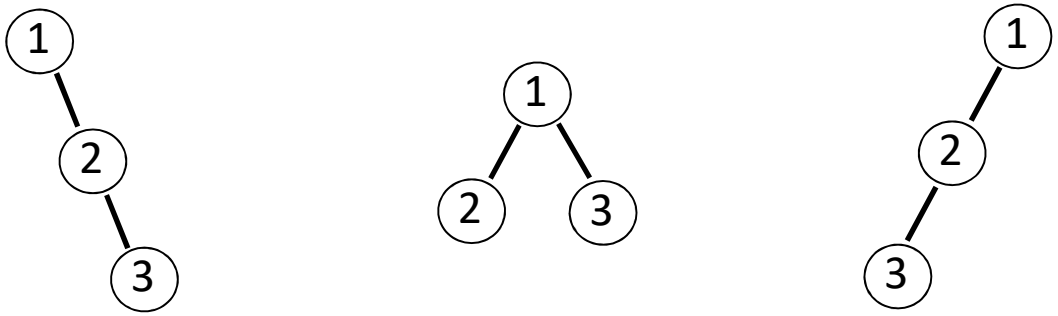
  1 2 3



Binary Tree for a Traversal Sequence

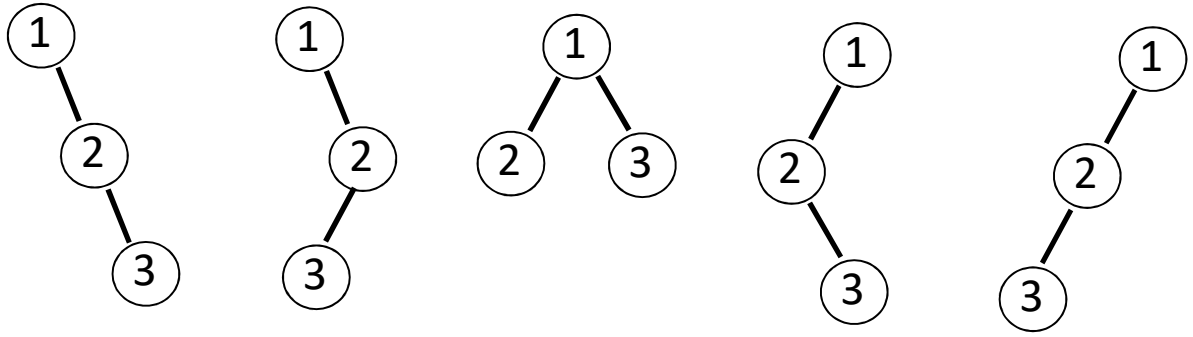
- If you are given a <u>preorder</u> sequence, can you define a binary tree uniquely?

  1 2 3



Counting Binary Trees

- Every binary tree has a unique pair of preorder and inorder sequences



Inorder? 123, 132, 213, 231, 321

Preorder? 123 for all

Binary Tree from Traversal

- Can we reconstruct a binary tree satisfying the given traversal orders?
  - Inorder : C B D A F G E
  - Preorder : A B C D E F G

Binary Tree from Traversal

- Inorder
  - If you pick a letter, all the letters on the left/right are in the left/right subtree of that letter
  - but now we cannot tell which one is the root

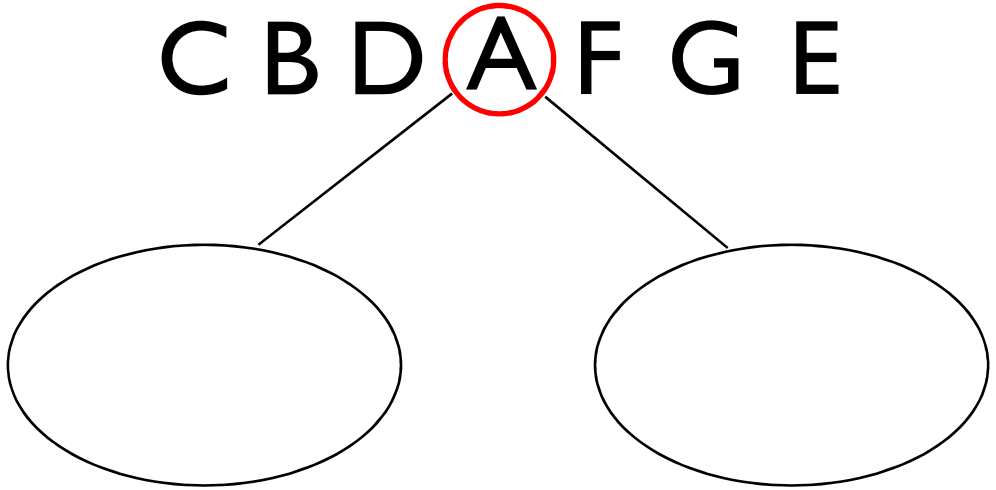
C B D A F G E

Binary Tree from Traversal

- Preorder
  - The front-most letter is the root node

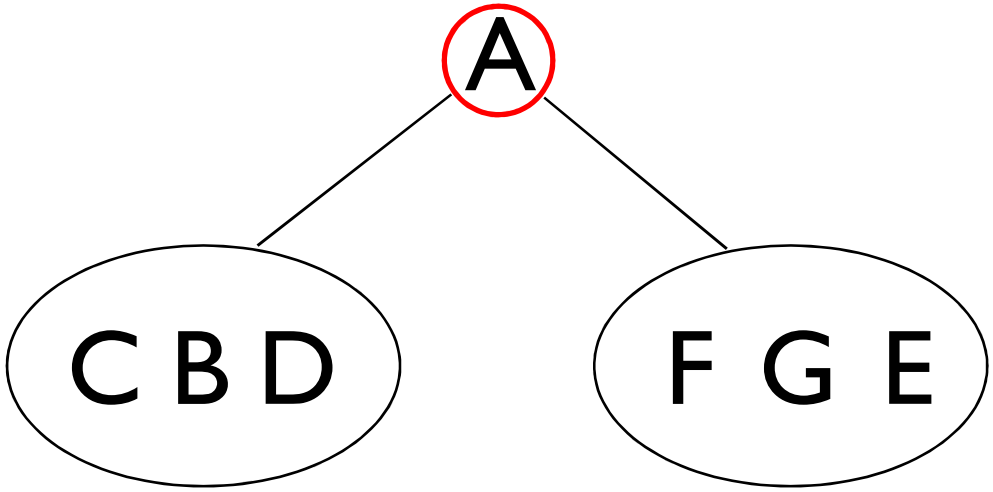
Binary Tree from Traversal

- Inorder
  - Because we now know the root, we can divide into left/right subtree



Binary Tree from Traversal

- Inorder breaks the tree into two subtrees
- Follow-up steps
  - –Preorder visits all left subtree elements
  - –Recursive construct left subtree
  - –Then recursively construct right subtree



A B C D E F G

Define a Unique Binary Tree

- Inorder – postorder
- Inorder – preorder
- Inorder – level order

- Other combination of traversal sequences can not define a unique binary tree

Outline

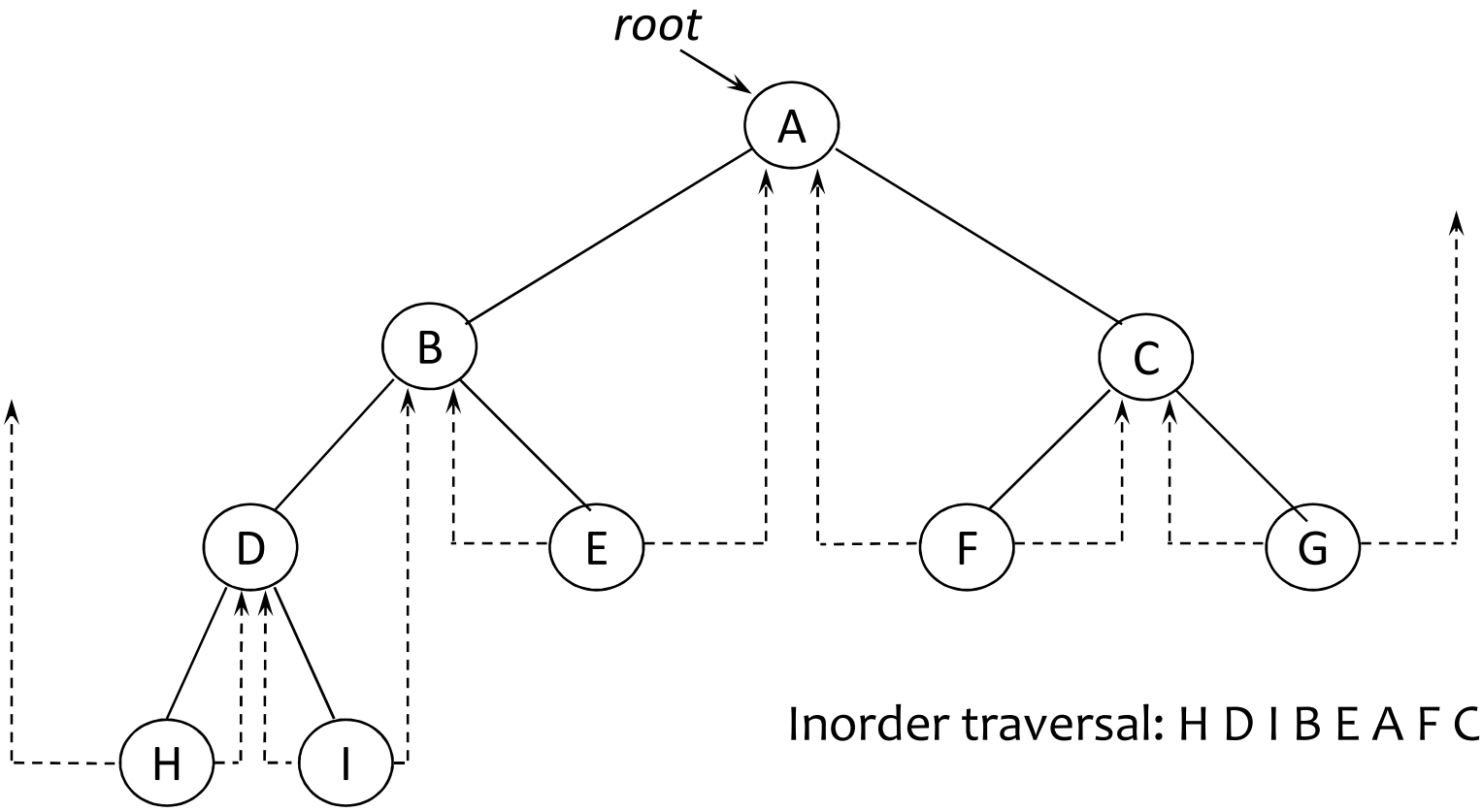
- Binary tree traversal
- Counting binary trees
- Threaded binary trees

Threaded Binary Trees

- Binary tree with n nodes
  - Total # of link pointers : 2n
  - Total # of null links : n+1 -> ~50% are wasted!
- Idea
  - Use null links to represent traversal order
  - Null in rightChild: next node for inorder traversal
  - Null in leftChild: previous node for inorder traversal
  - Benefit: traverse a tree inorder without a stack

Threaded Binary Trees (inorder)

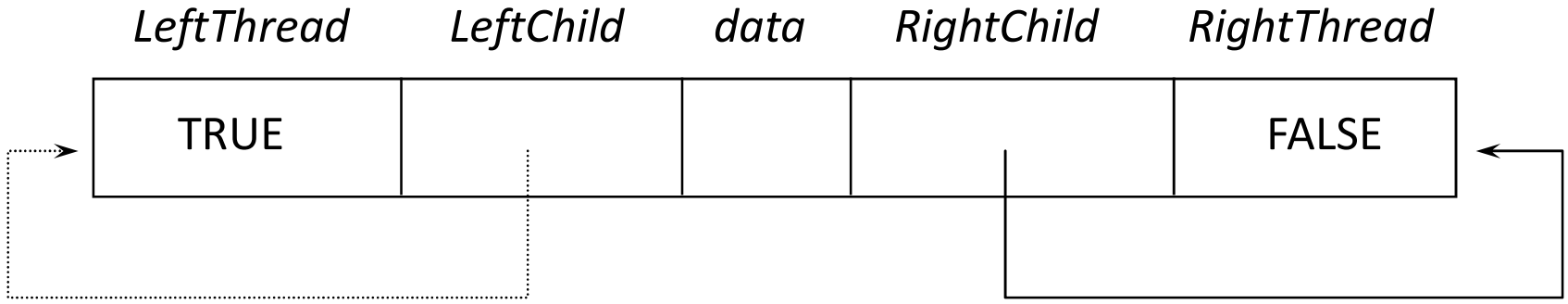
- Dotted line : thread



Inorder traversal: H D I B E A F C G

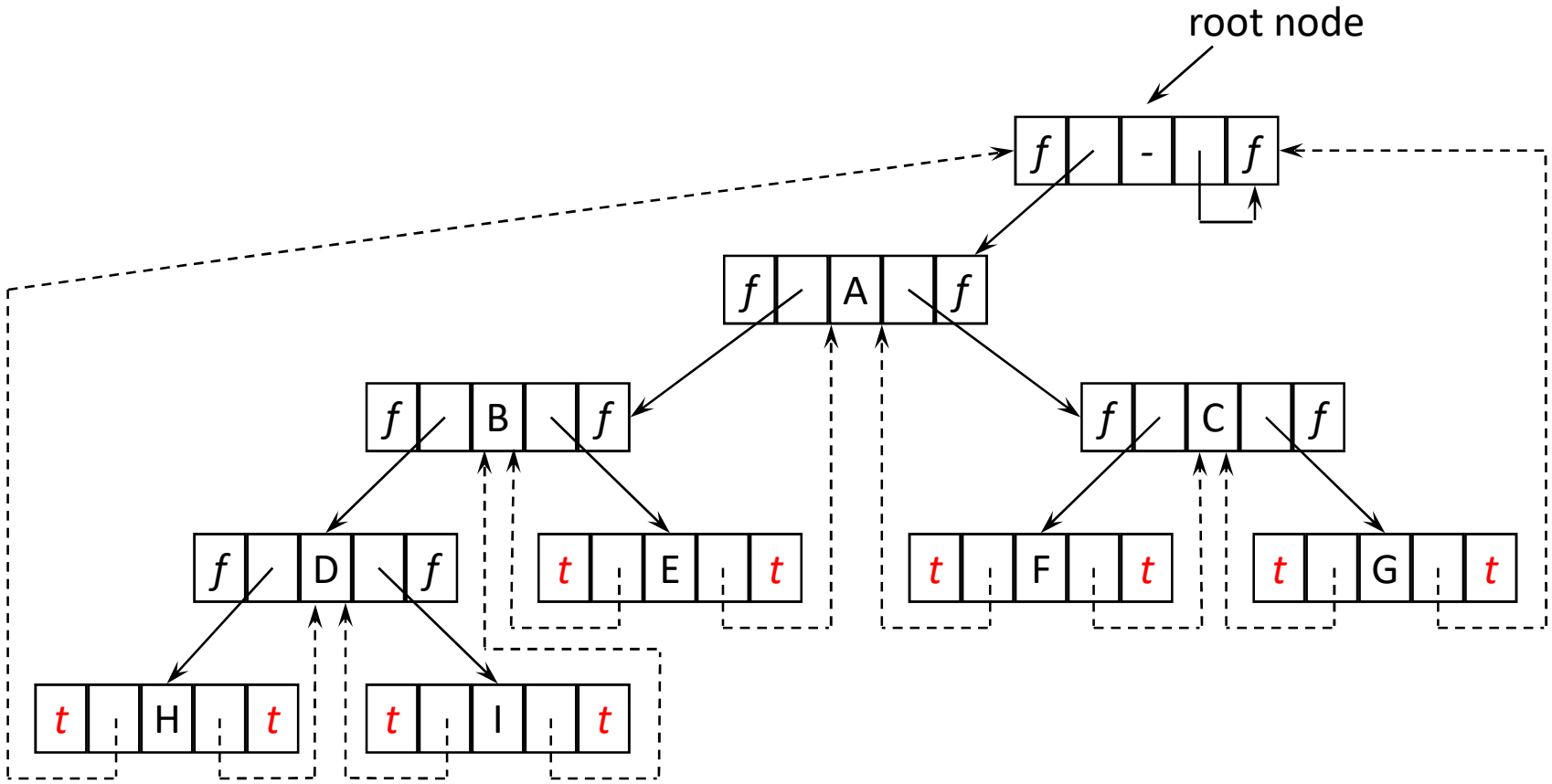
Threaded Binary Trees (inorder)

- Node representation
  - Need to distinguish child / thread pointer
    - One bit Boolean flag for each pointer
- Initializing tree
  - Root node pointing out itself (no null links at all)



Example of empty threaded binary tree

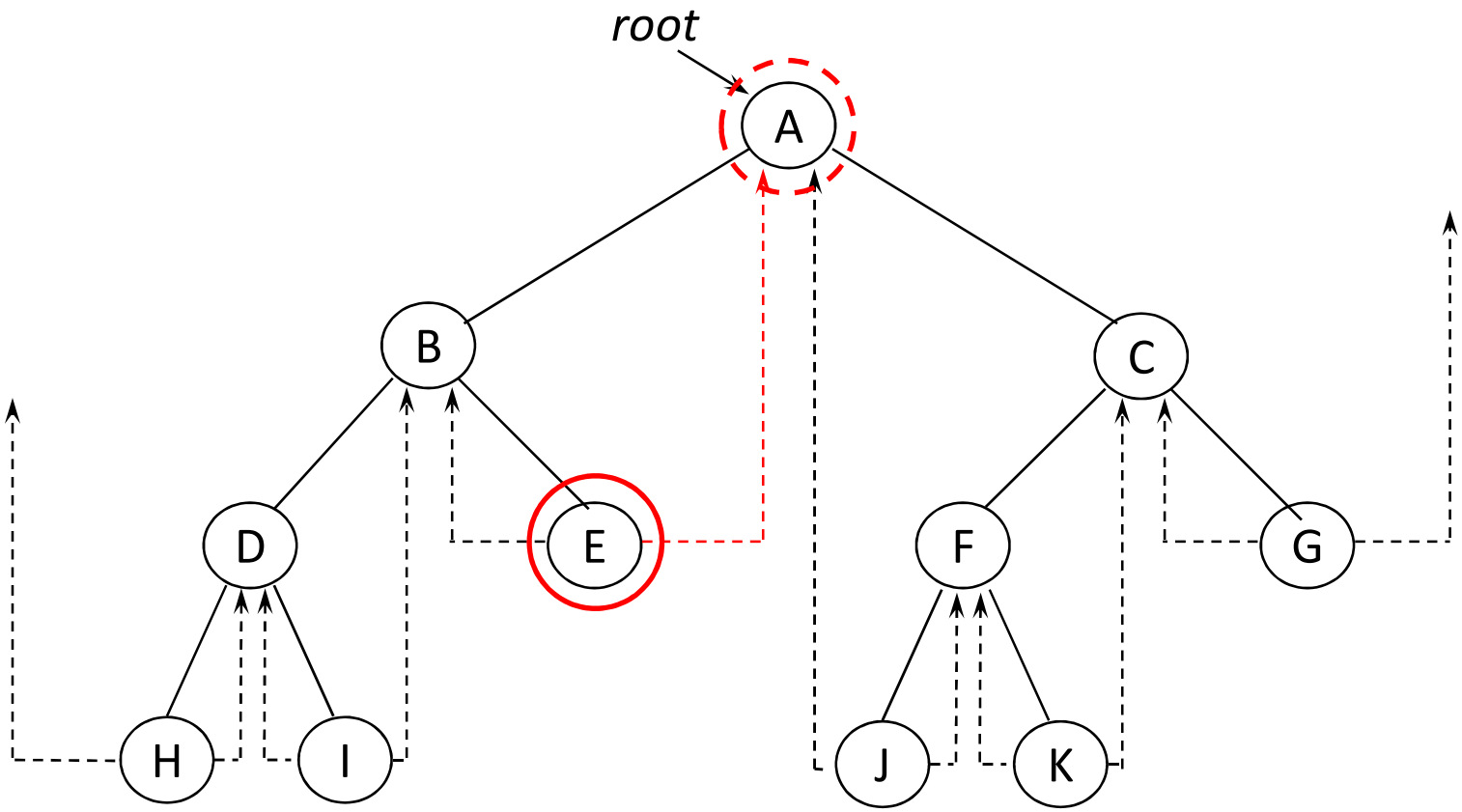
Threaded Binary Trees (inorder)



f = FALSE; t = TRUE

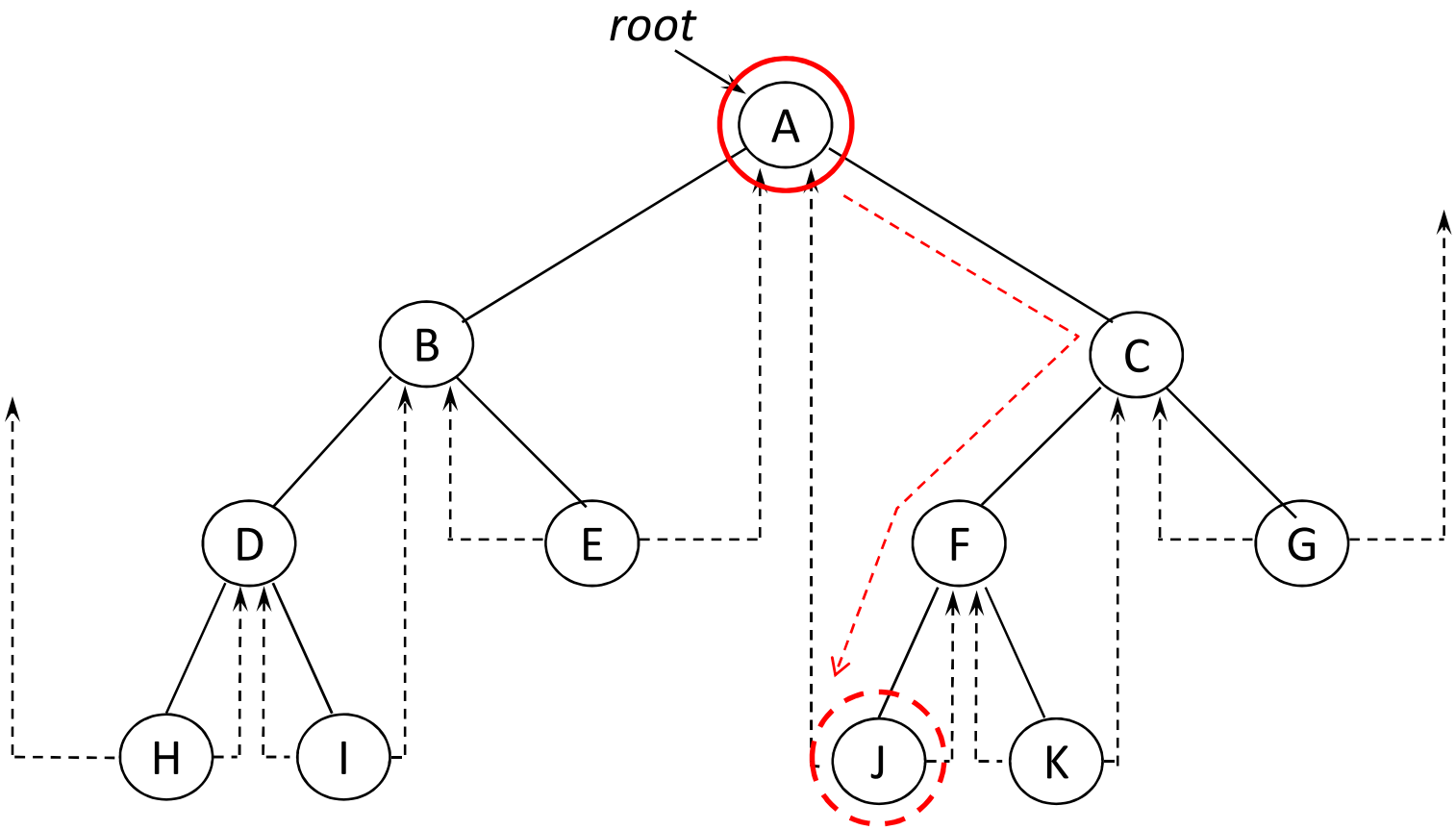
Current node : E

RightThread is True, so the next node for inorder traversal is E->RightChild = A



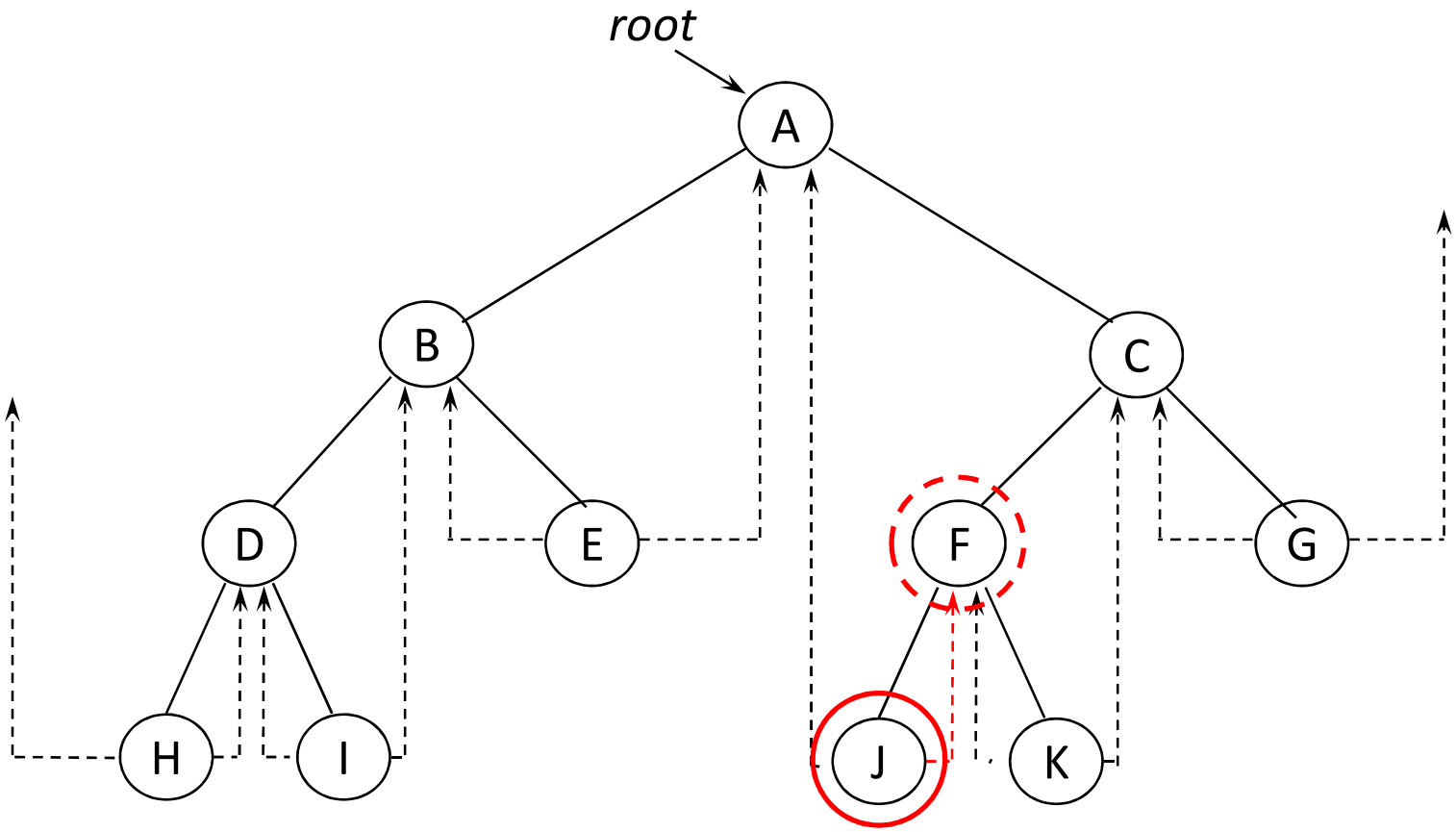
Current node : A

RightThread is False, so you follow the LeftChild link of RightChild node (C) until you reach the node with the LeftThread is True (which is J)



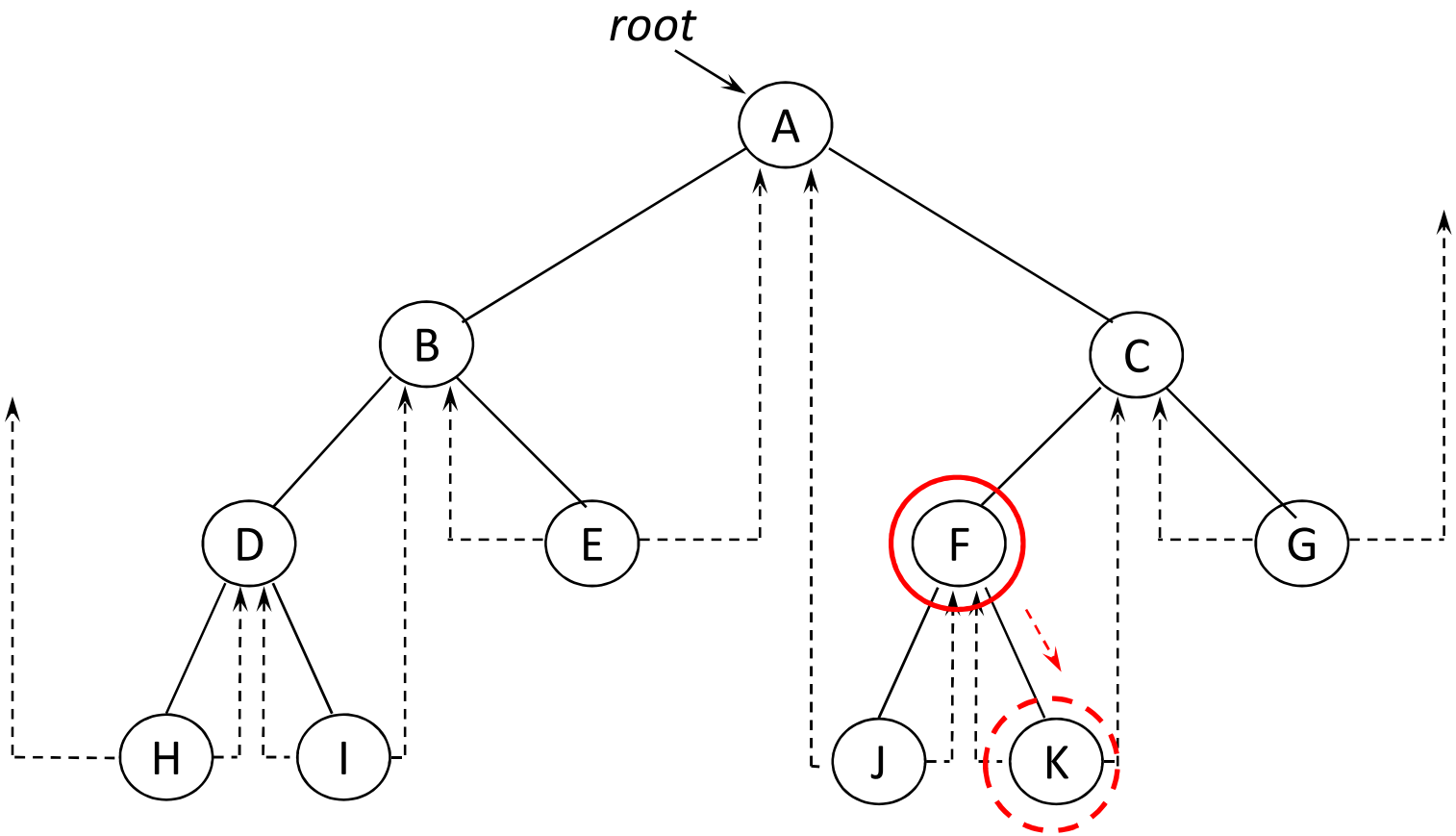
Current node : J

RightThread is True, so the next node for inorder traversal is J->RightChild = F



Current node : F

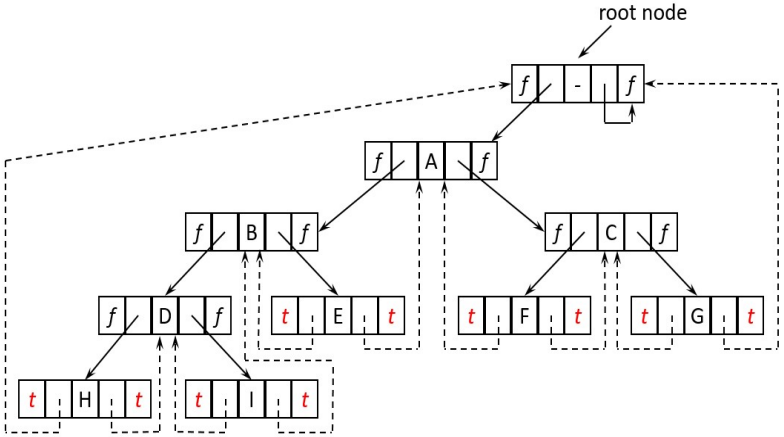
RightThread is False, so you follow the LeftChild link of RightChild node (K) until you reach the node with the LeftThread is True (which is K itself)



Traverse Threaded Binary Tree

- Traverse inorder without using a stack

```
Next() // return next inorder
{
     ThreadedNode *temp = CurrentNode->RightChild;
     if (!CurrentNode->RightThread)
          while (!temp->LeftThread) temp = temp->LeftChild;
     CurrentNode = temp;
}
```



1) Start from root node: RightThread is false, so all the way down to the left
2) If CurrentNode is root node, done with inorder traversal

Questions?