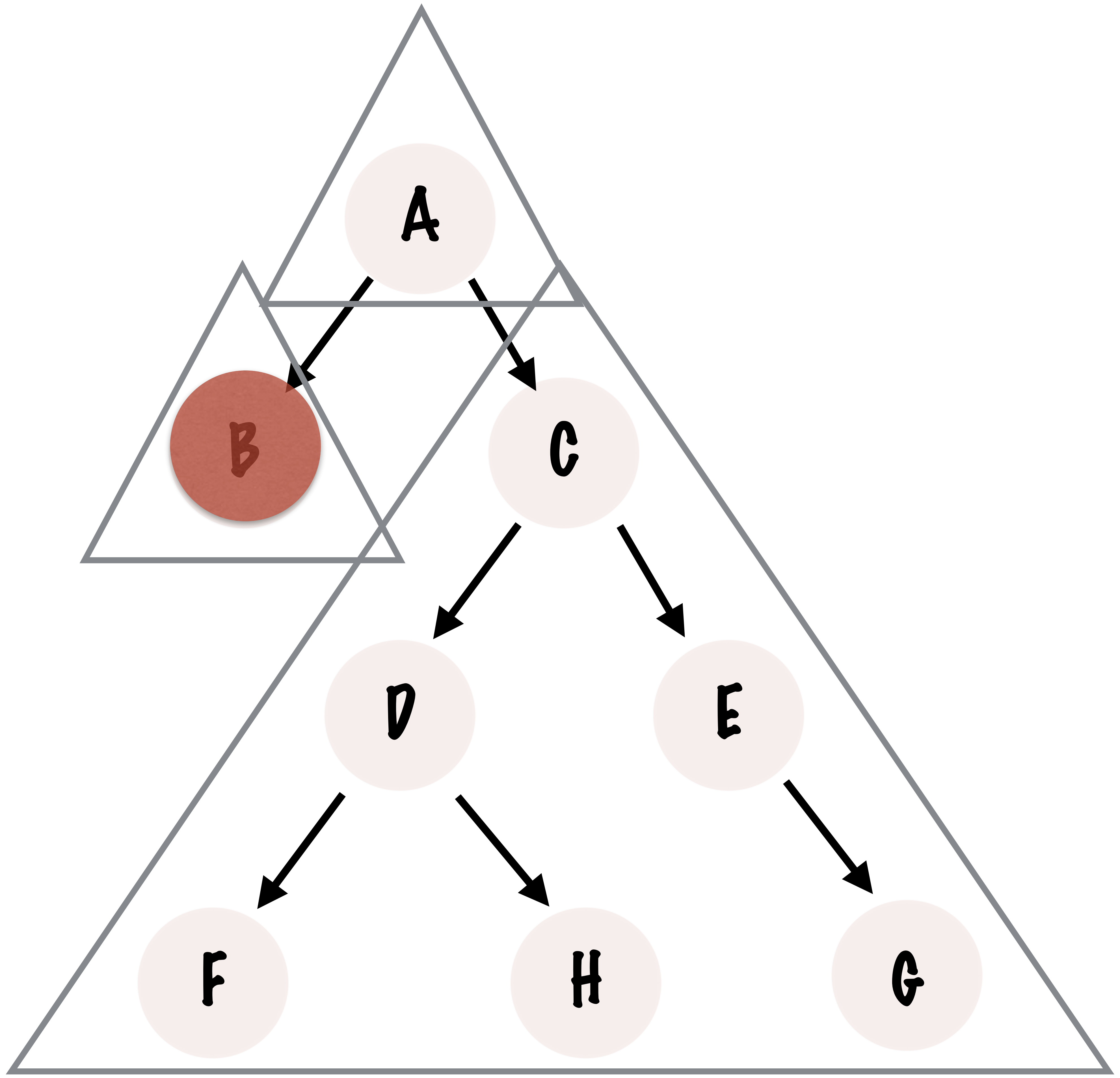
IN-ORDER TRAVERSAL

THE LEFT SUBTREE IS PROCESSED FIRST, THEN THE NODE, THEN THE RIGHT SUBTREE



LEFT SUBTREE

↓

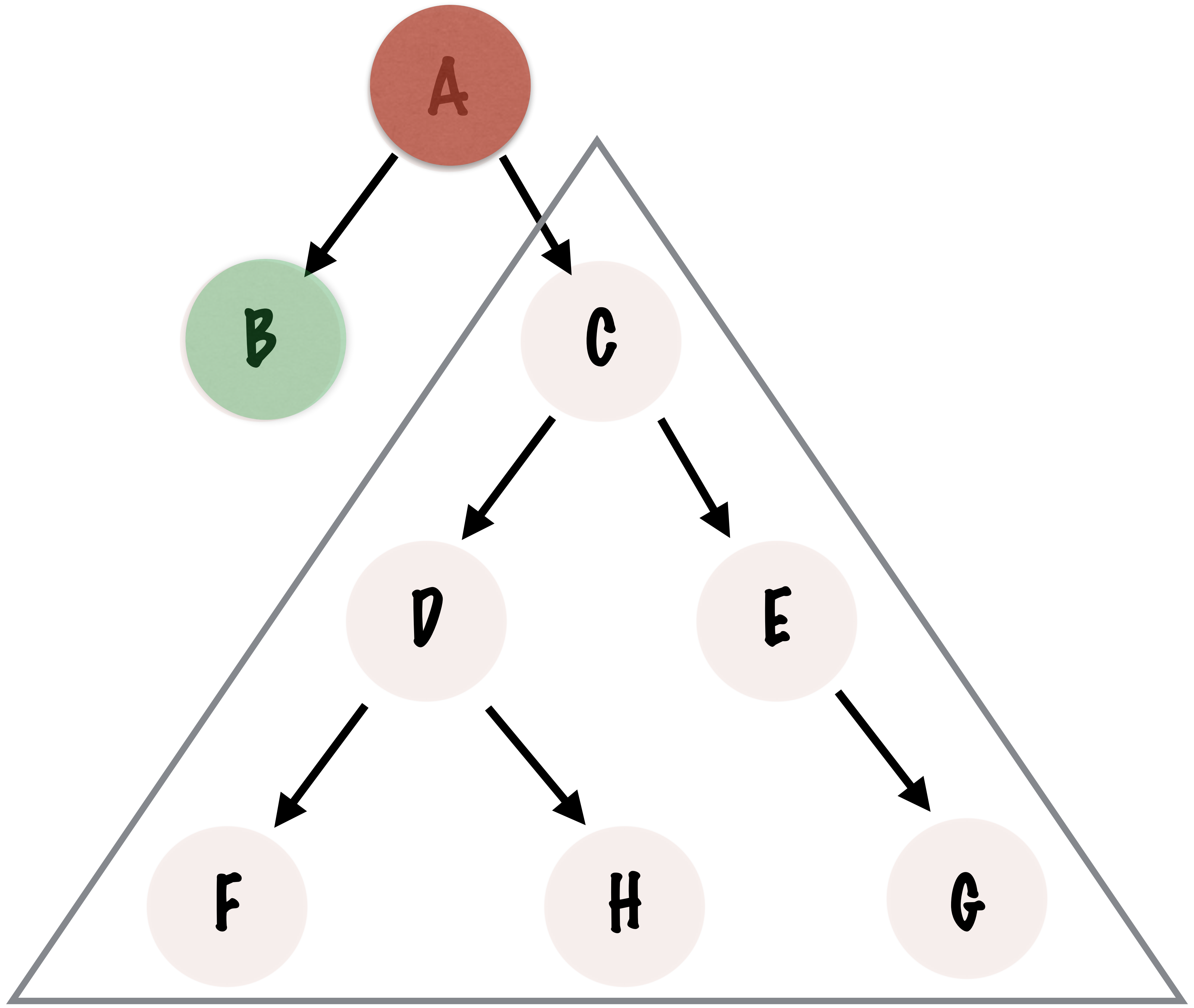
NODE

↓

RIGHT SUBTREE

THE SUBTREE ROOTED AT B IS PROCESSED BEFORE A AND THE SUBTREE ROOTED AT C

IN-ORDER TRAVERSAL



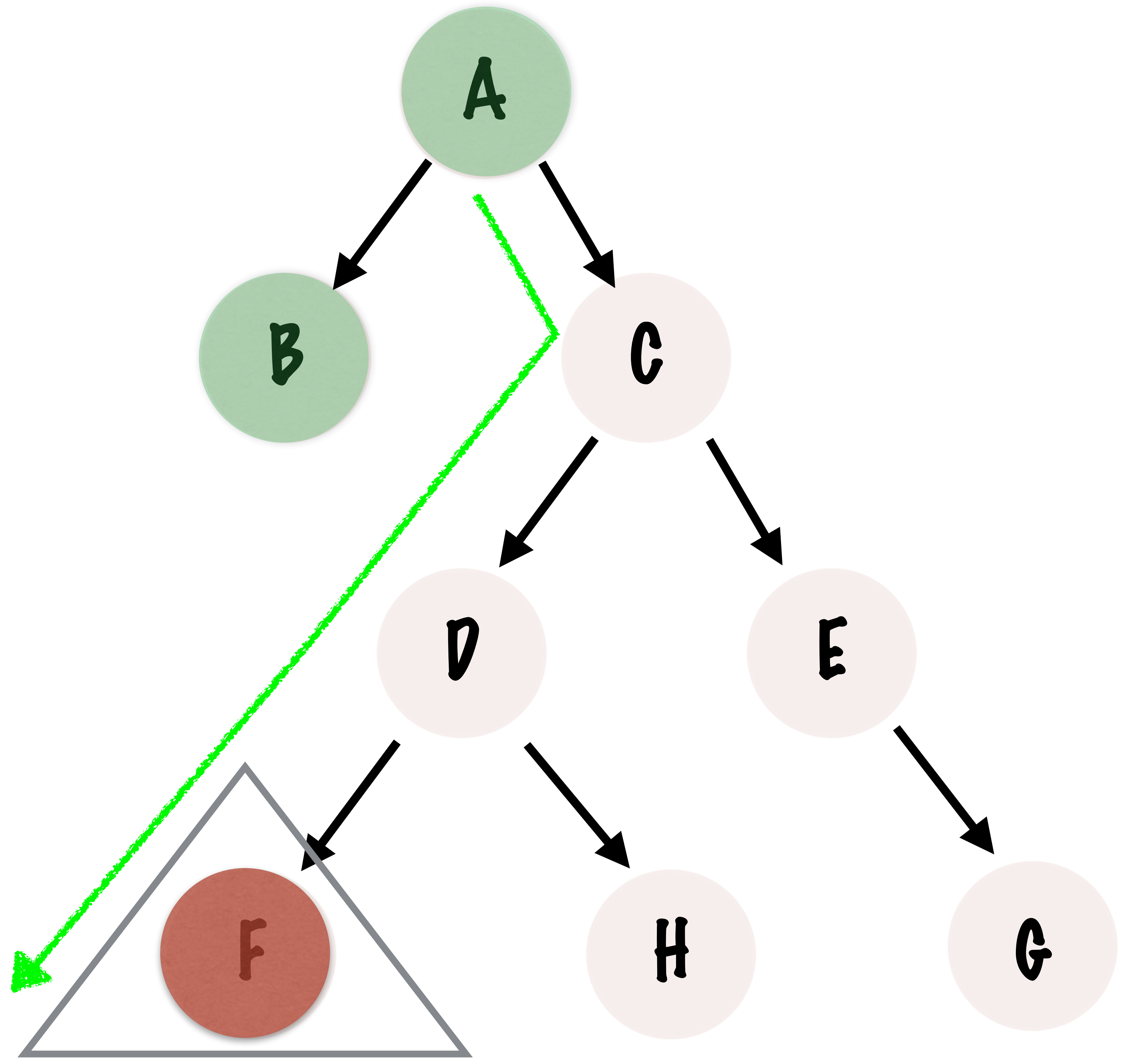
B

IN-ORDER TRAVERSAL

THE SUBTREE ROOTED AT D WILL BE PROCESSED BEFORE C AND D

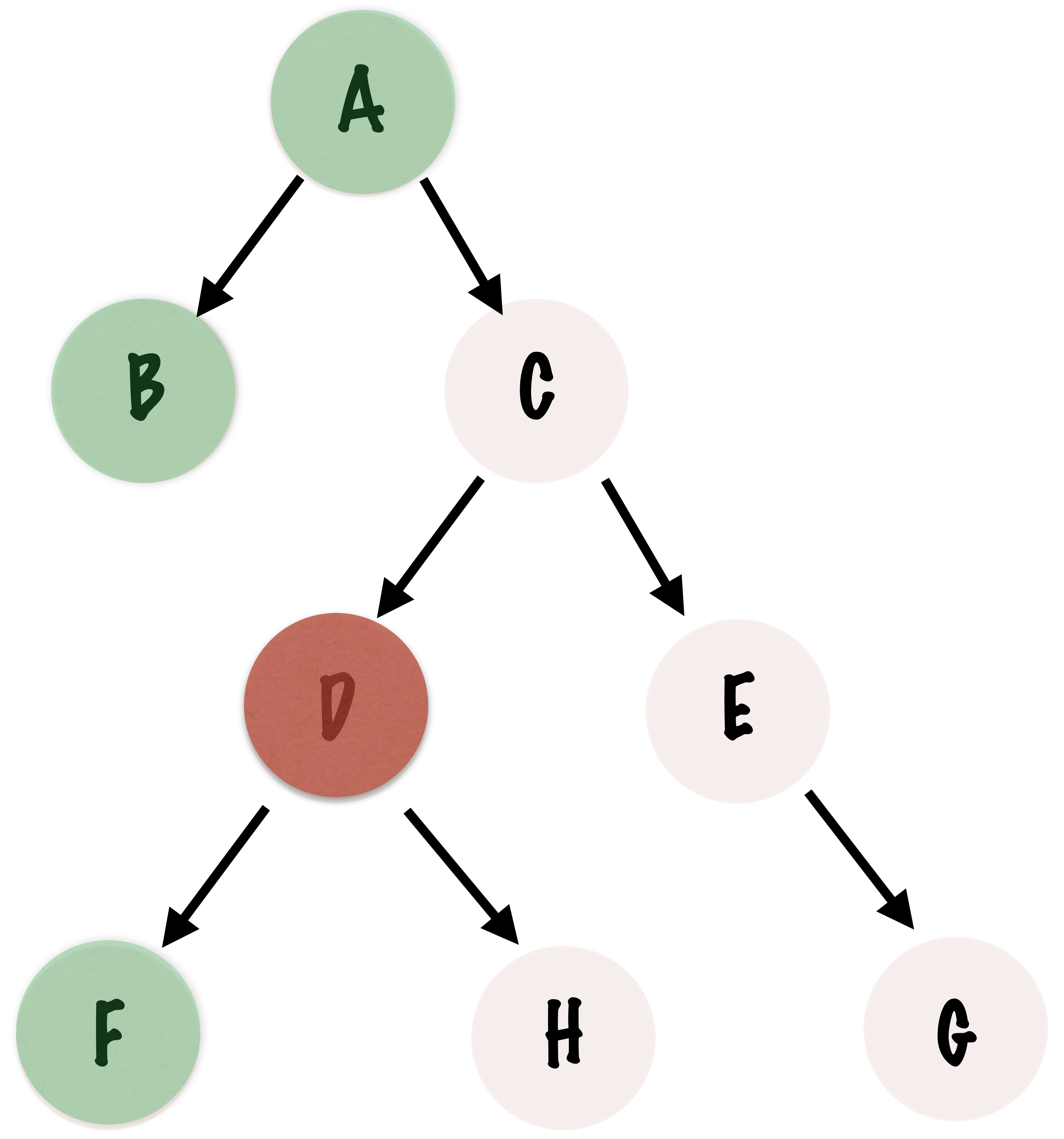
EACH TIME A NODE HAS A LEFT CHILD, WE HAVE TO MOVE DEEPER INTO THE LEFT SUBTREE

F WILL BE THE NEXT NODE PROCESSED



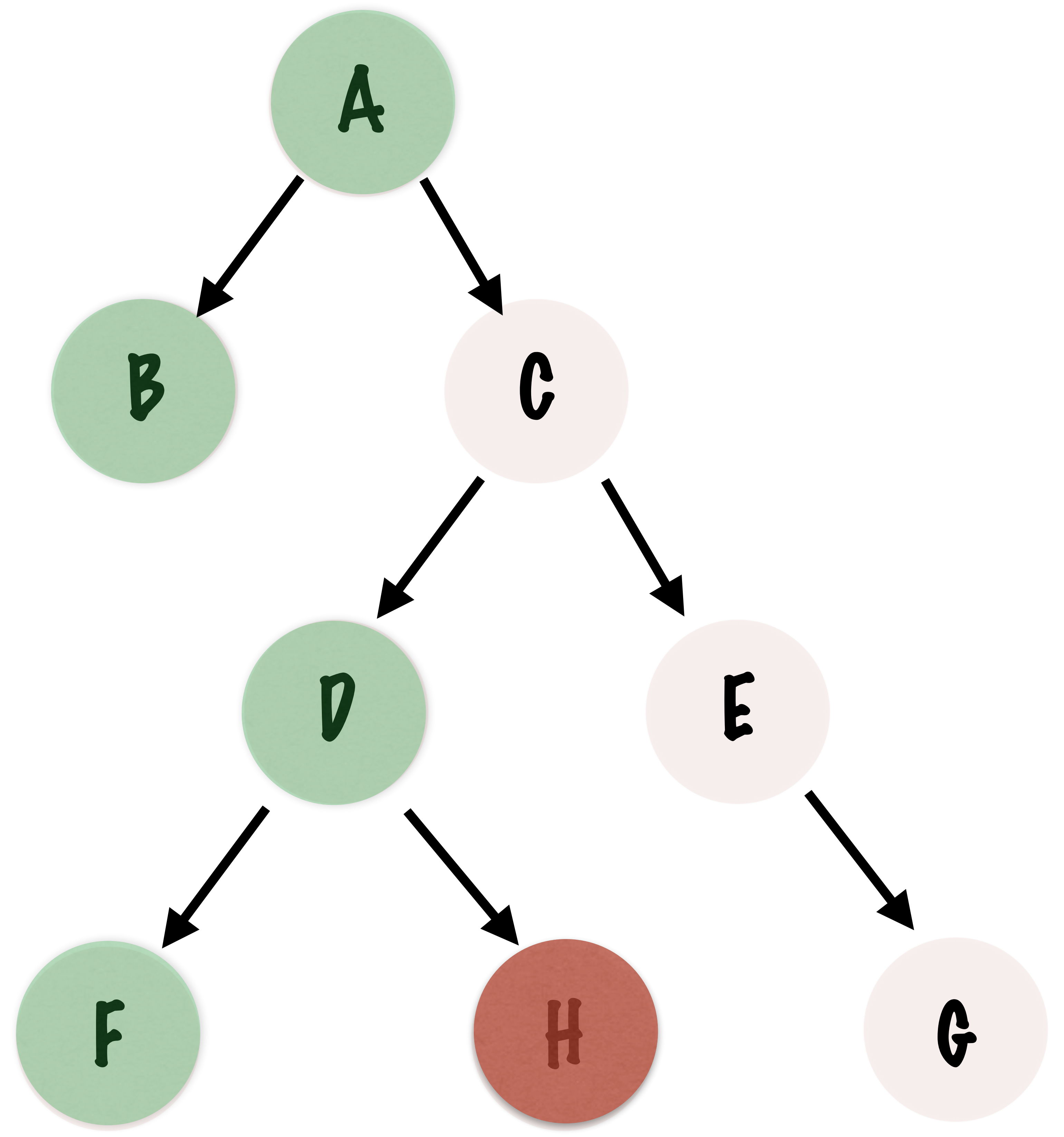
B->A

IN-ORDER TRAVERSAL



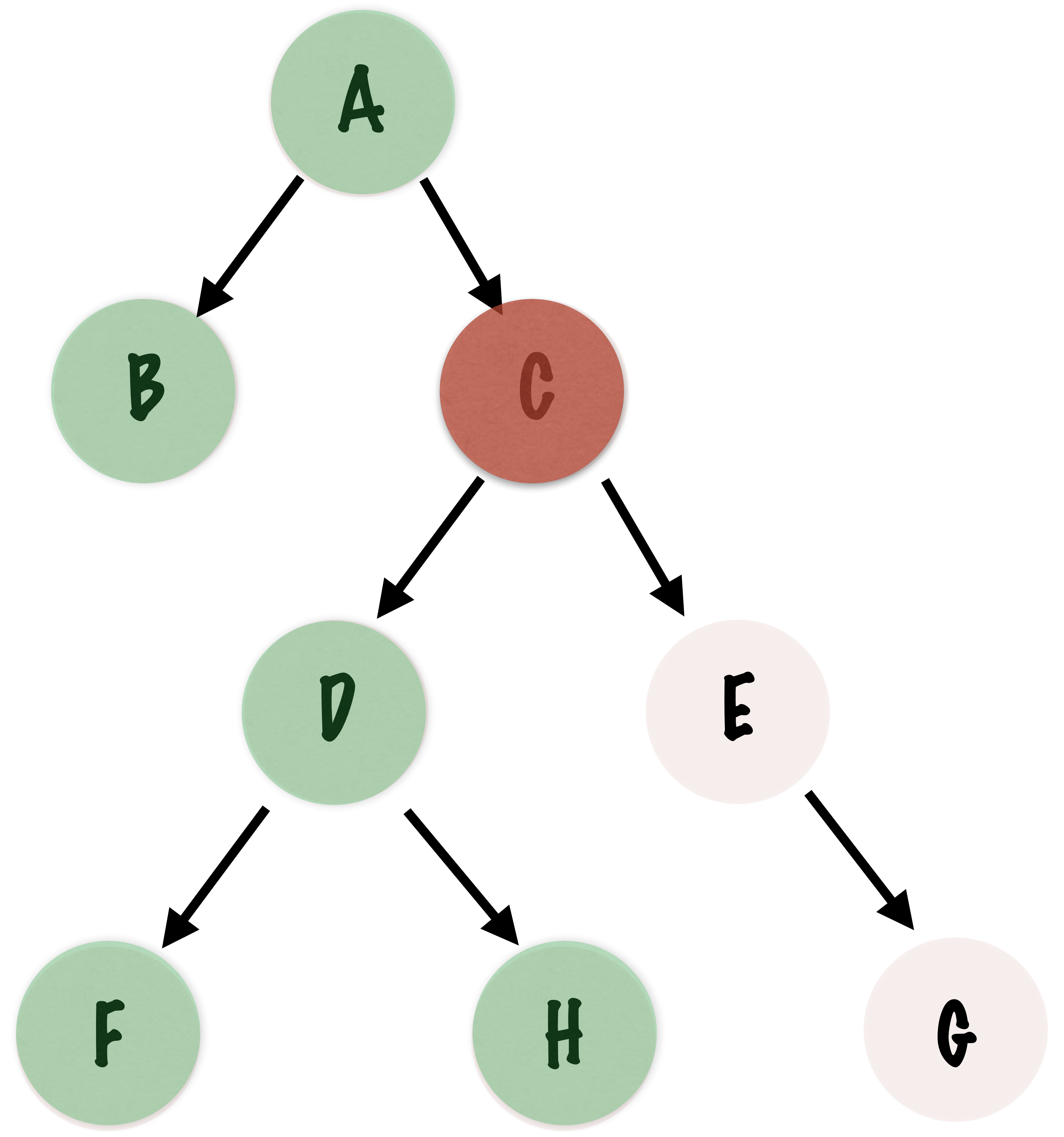
B->A->F

IN-ORDER TRAVERSAL



B->A->F->D

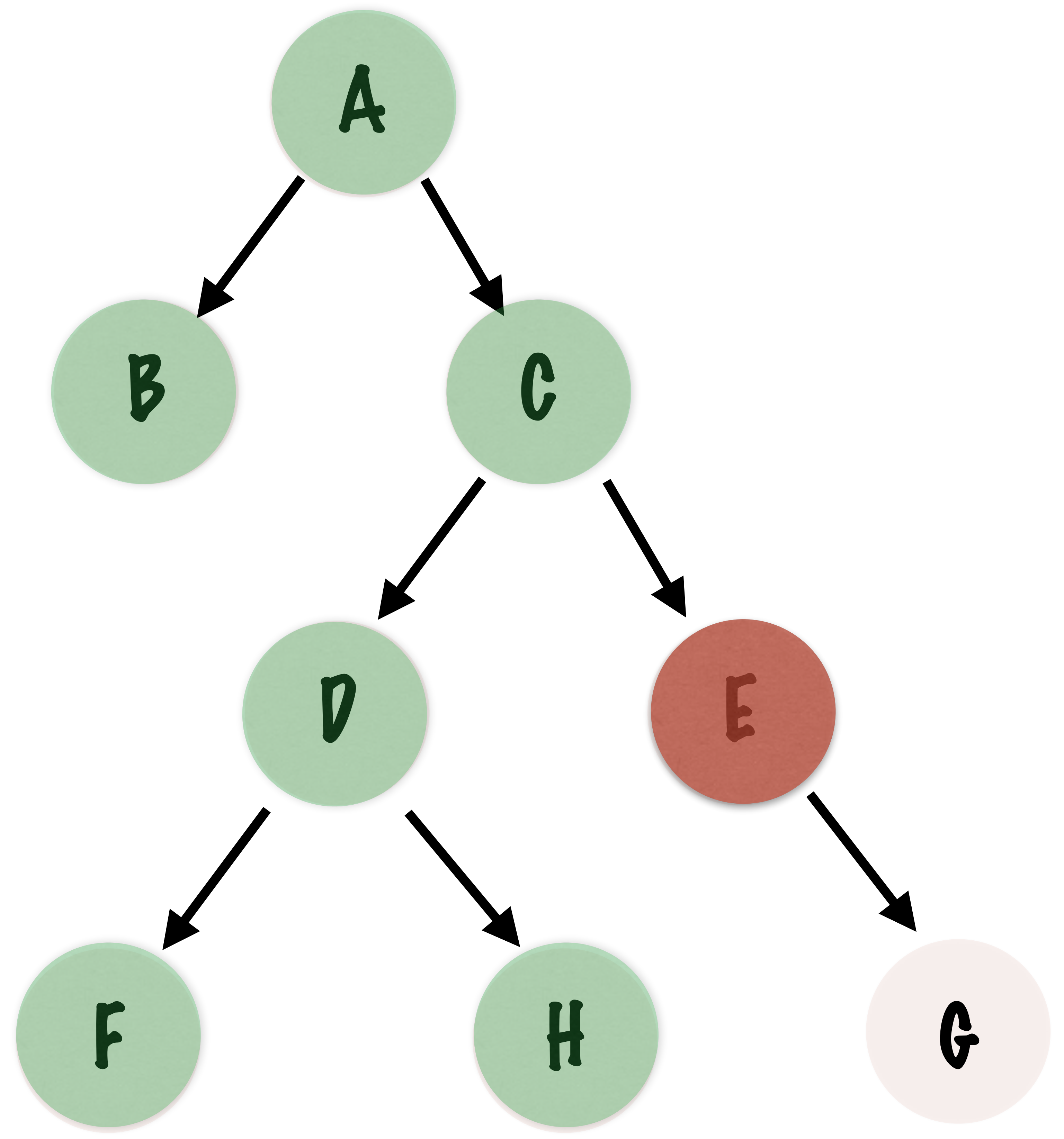
IN-ORDER TRAVERSAL



AFTER PROCESSING C, WE CAN MOVE ON TO IT'S RIGHT CHILD AND THE SUBTREE

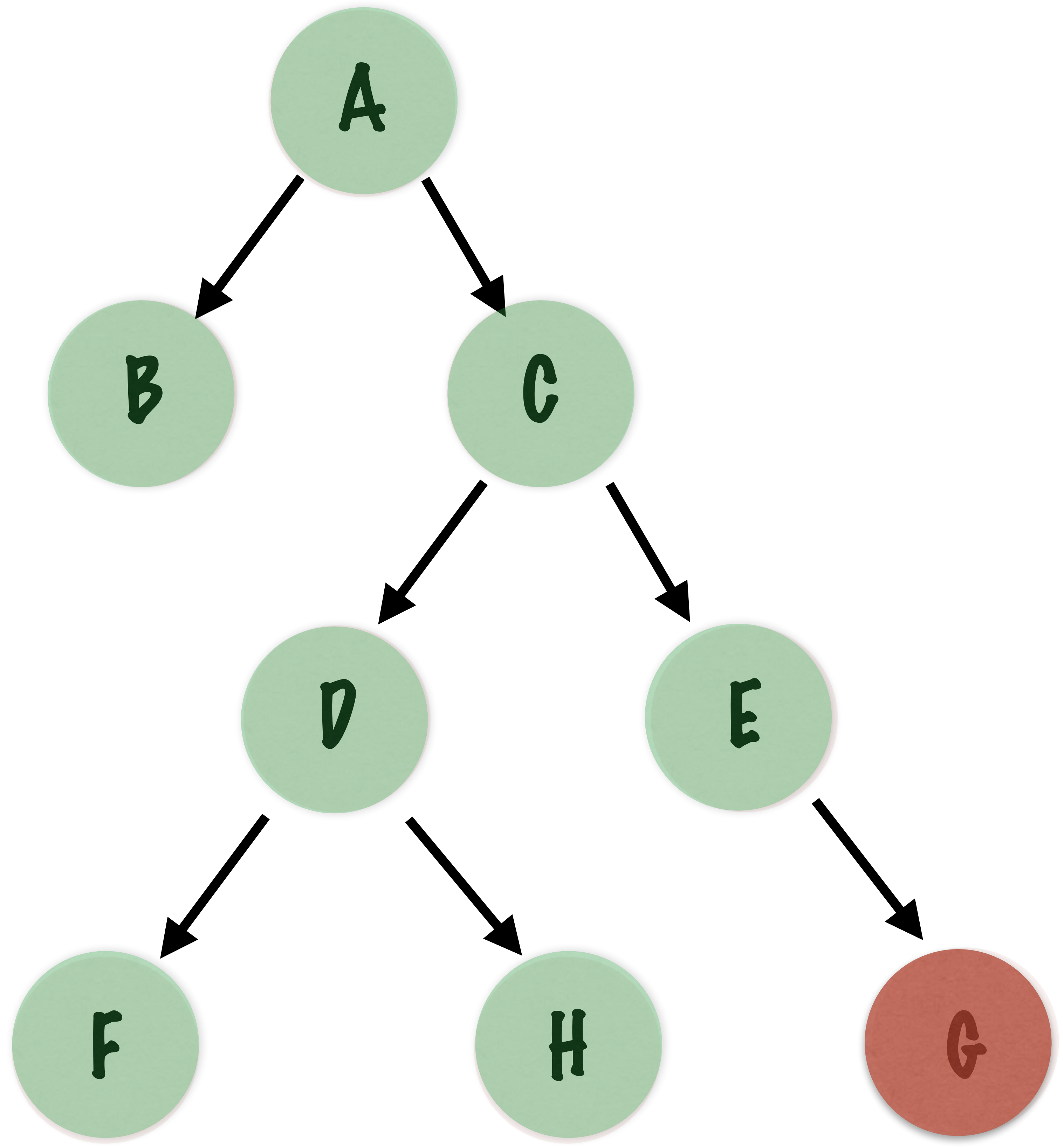
B->A->F->D->H

IN-ORDER TRAVERSAL



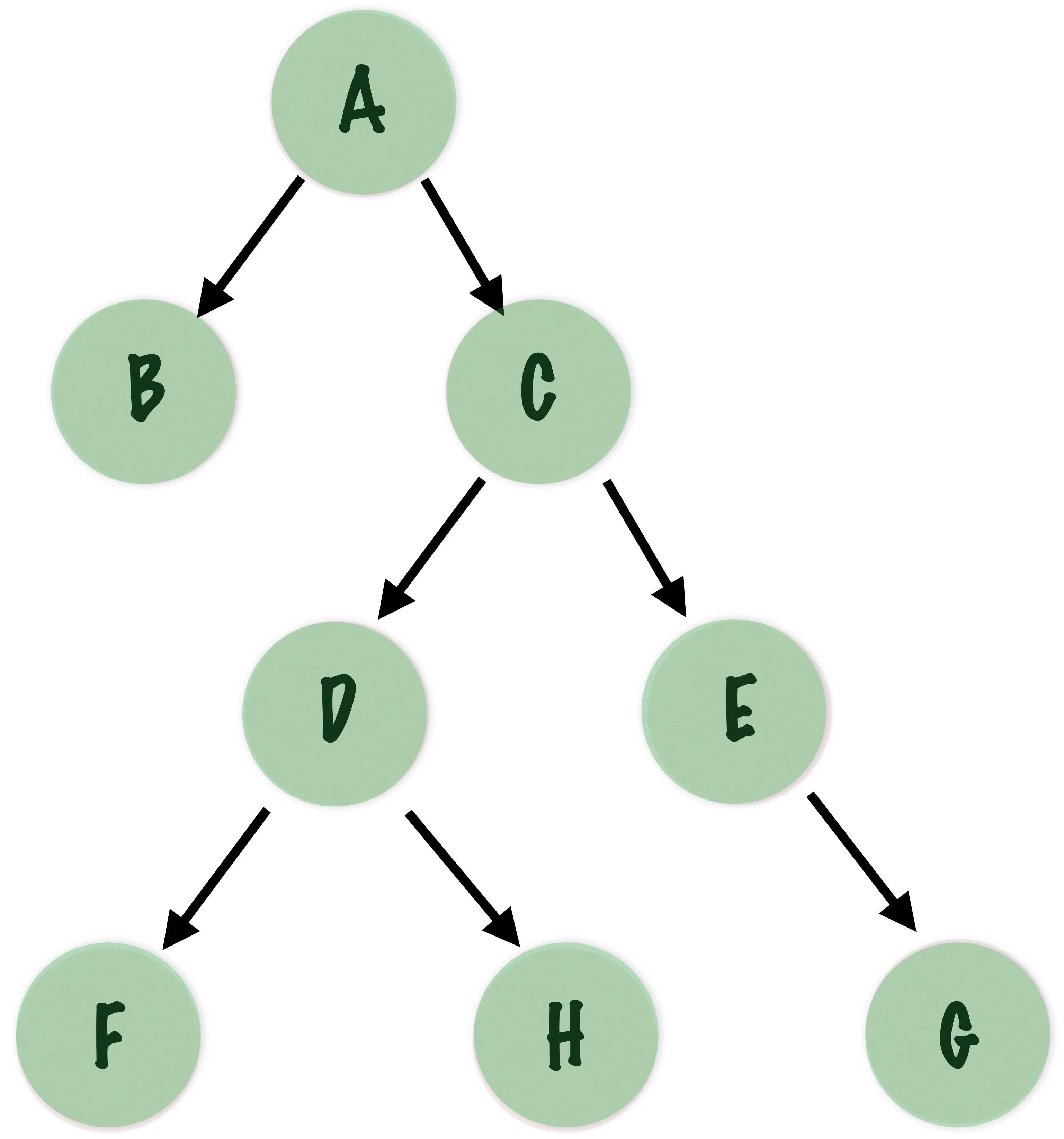
B->A->F->D->H->C

IN-ORDER TRAVERSAL



B->A->F->D->H->C->E

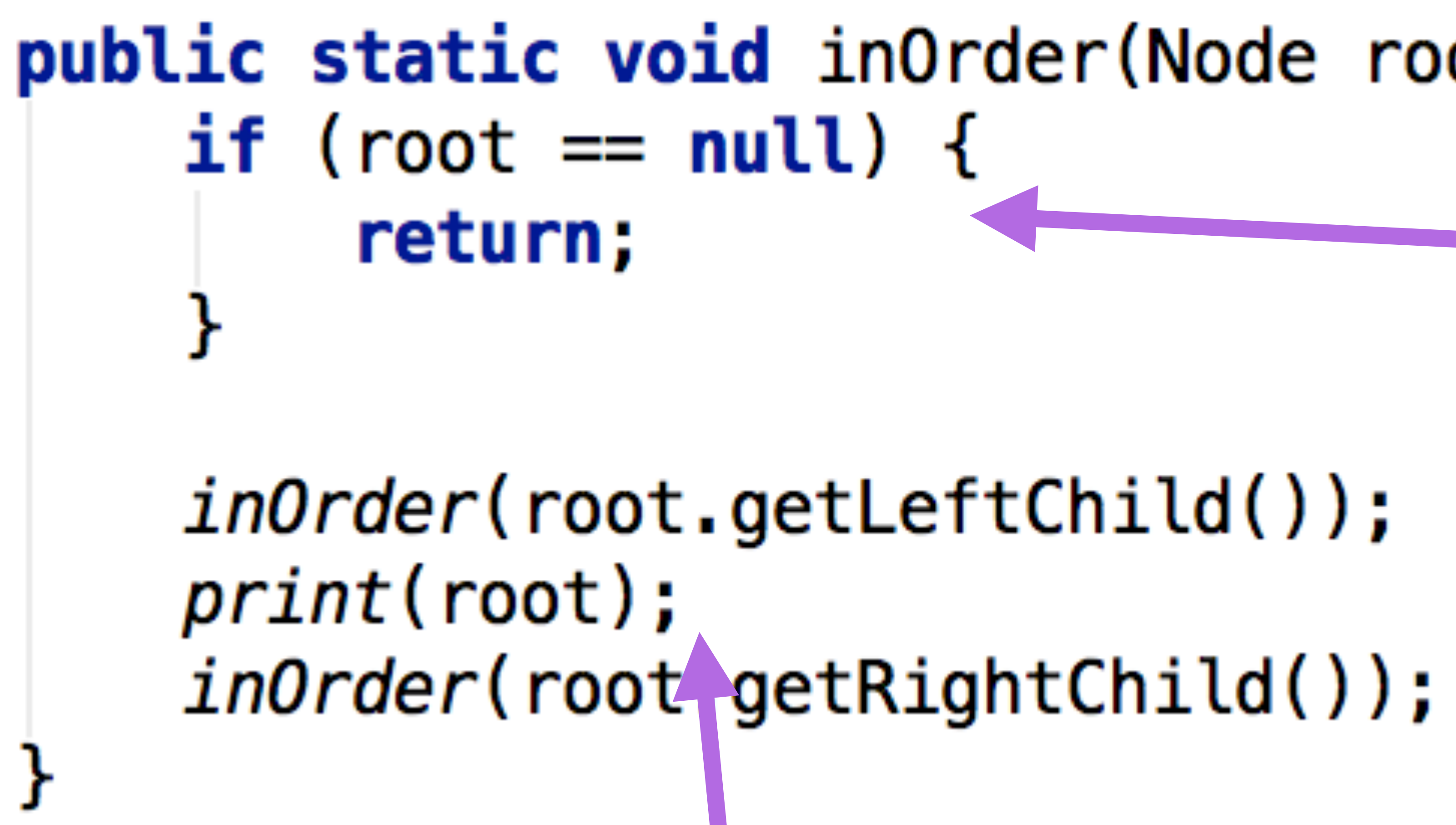
IN-ORDER TRAVERSAL



ALL NODES HAVE BEEN VISITED!

B->A->F->D->H->C->E->G

IN-ORDER TRAVERSAL CODE



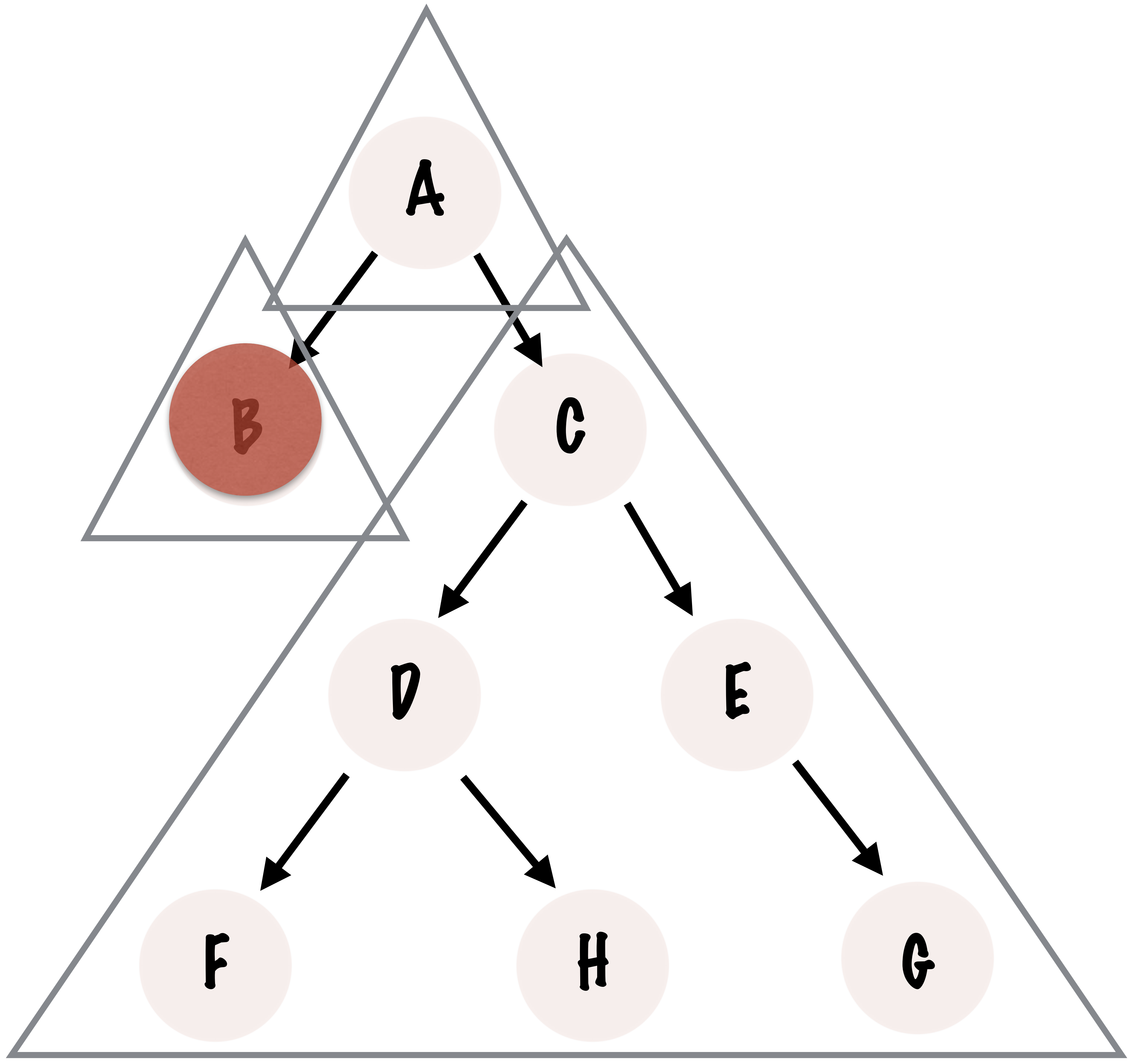
```
public static void inOrder(Node root) {
    if (root == null) {
        return;
    }

    inOrder(root.getLeftChild());
    print(root);
    inOrder(root.getRightChild());
}
```

BASE CASE - NOTHING TO TRAVERSE

PROCESS THE LEFT SUBTREE BEFORE THE NODE AND THEN RECURSE TO THE RIGHT SUBTREES

POST-ORDER TRAVERSAL

BOTH SUBTREES ARE PROCESSED BEFORE THE NODE ITSELF. THE NODE IS PROCESSED AFTER (POST) THE SUBTREES



THE SUBTREE ROOTED AT B IS PROCESSED BEFORE THE SUBTREE ROOTED AT C. A IS PROCESSED LAST

LEFT SUBTREE

↓

RIGHT SUBTREE

↓

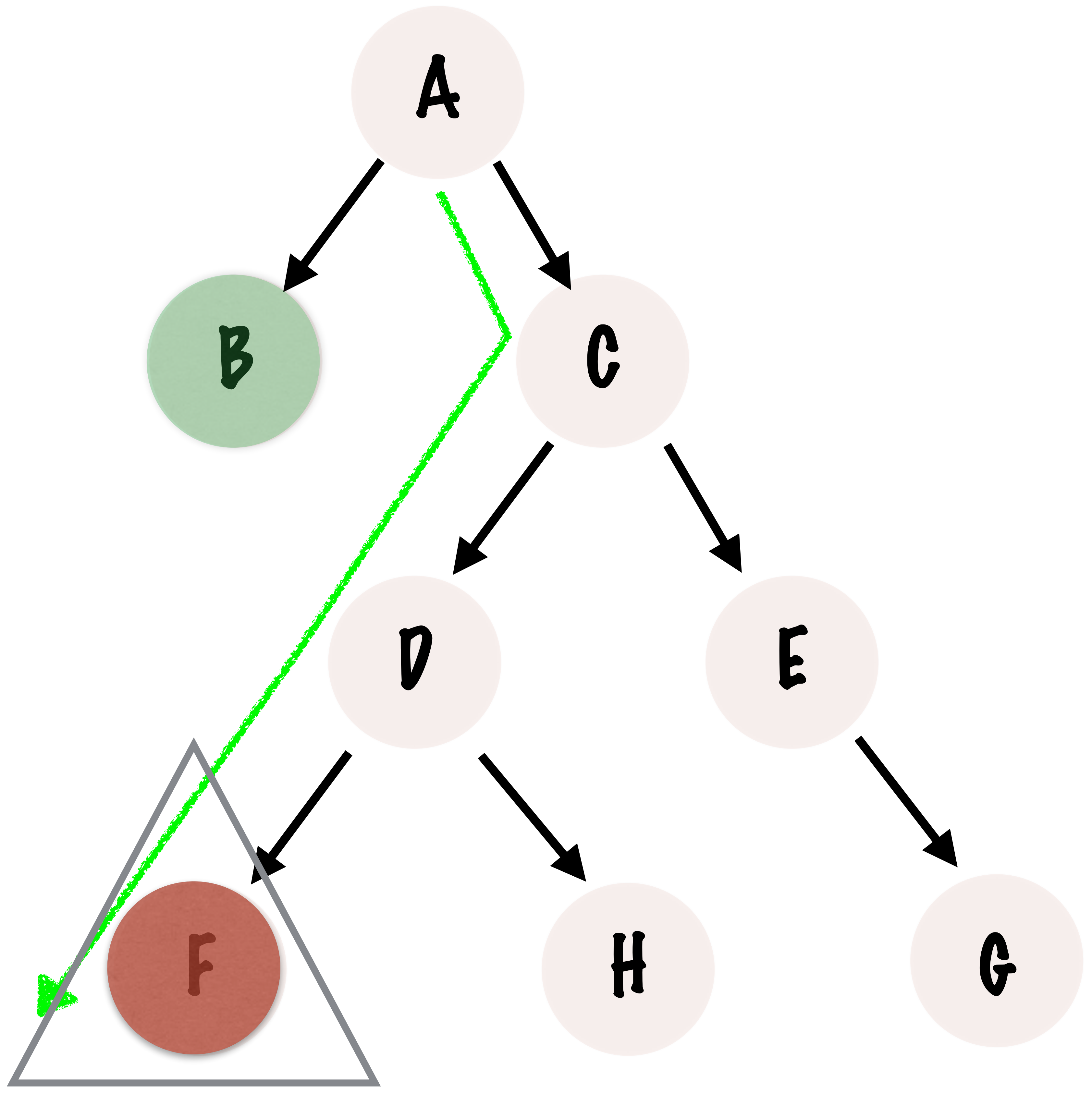
NODE

POST-ORDER TRAVERSAL

THE SUBTREE ROOTED AT D WILL BE PROCESSED BEFORE A OR C OR THE NODE D

WE MOVE DEEP TO FIND THE LEFTMOST NODE

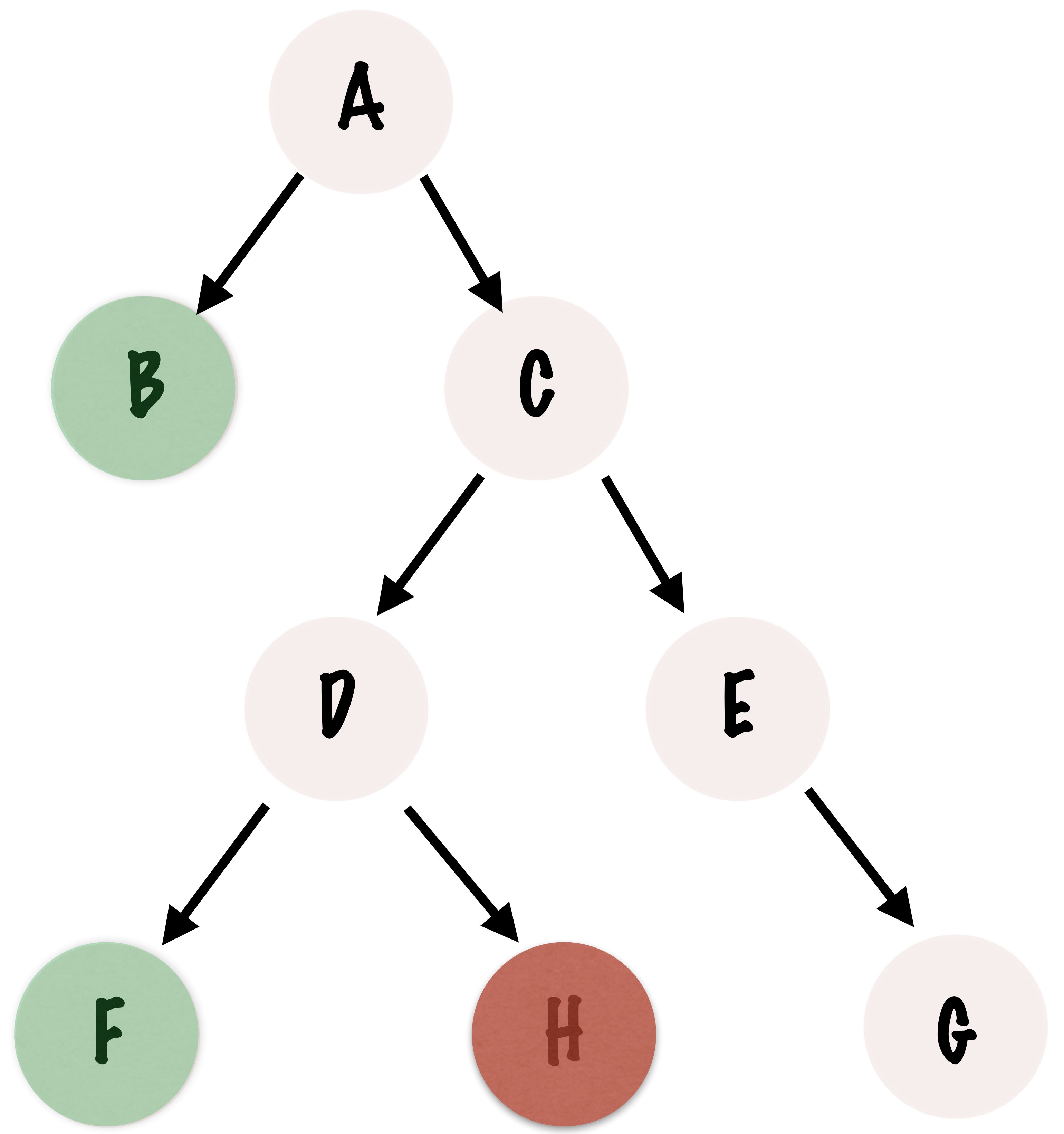
F WILL BE THE NEXT NODE PROCESSED



B

POST-ORDER TRAVERSAL

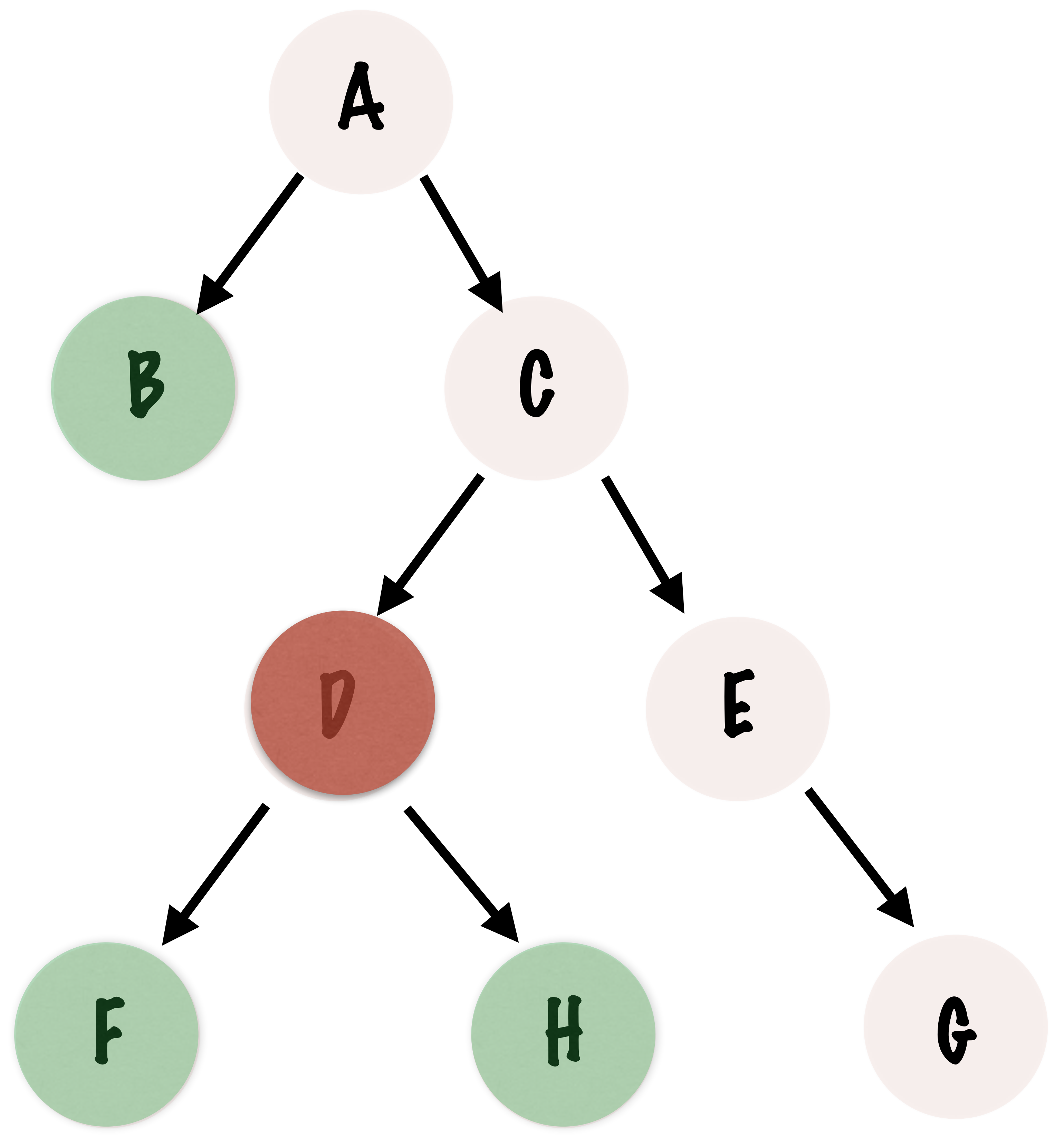
H, THE RIGHT CHILD OF D
WILL BE PROCESSED
BEFORE C OR D



B->F

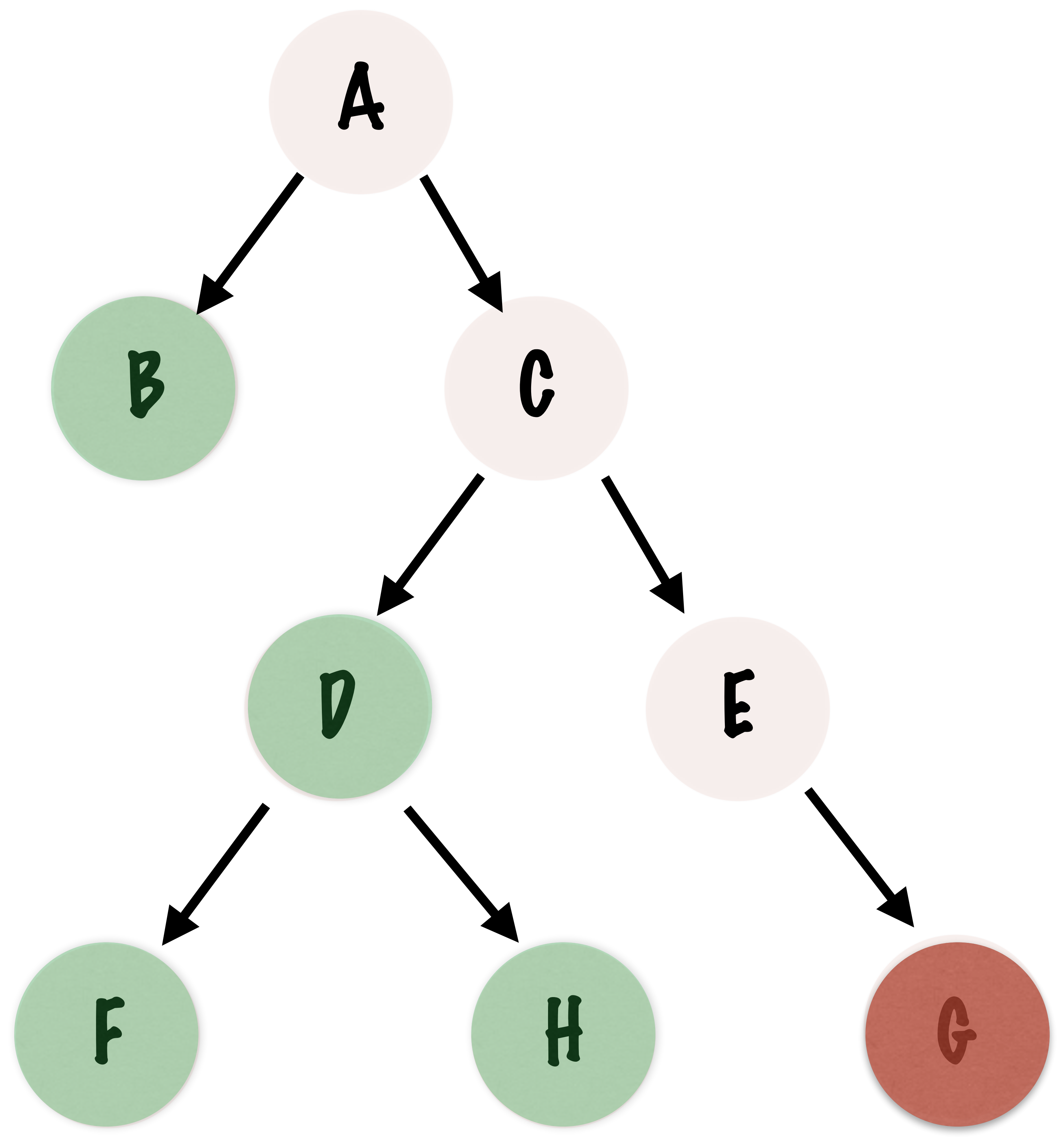
POST-ORDER TRAVERSAL

ONCE BOTH SUBTREES ARE PROCESSED - THEN THE NODE ITSELF CAN BE PROCESSED



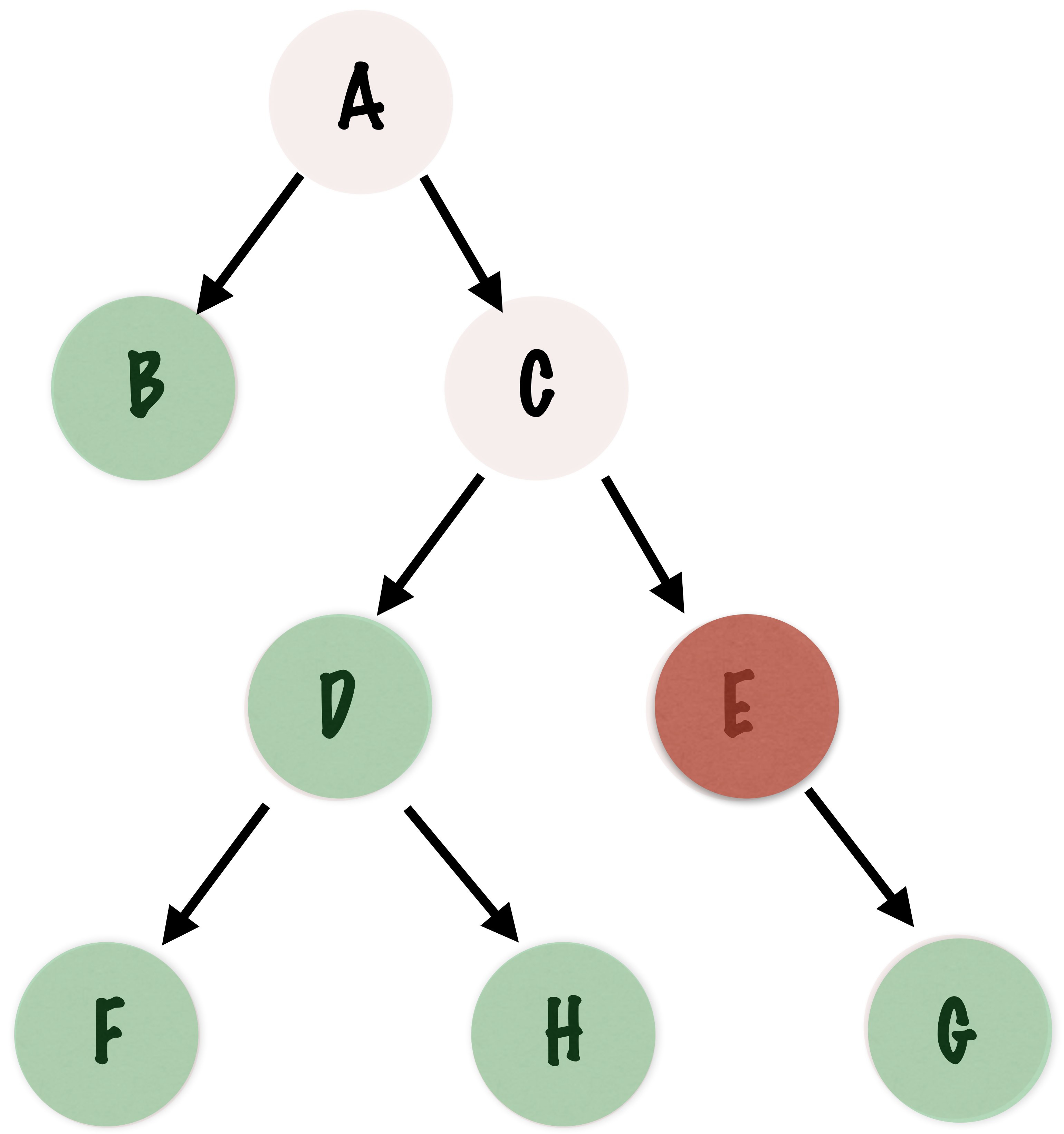
B->F->H

POST-ORDER TRAVERSAL



B->F->H->D

POST-ORDER TRAVERSAL

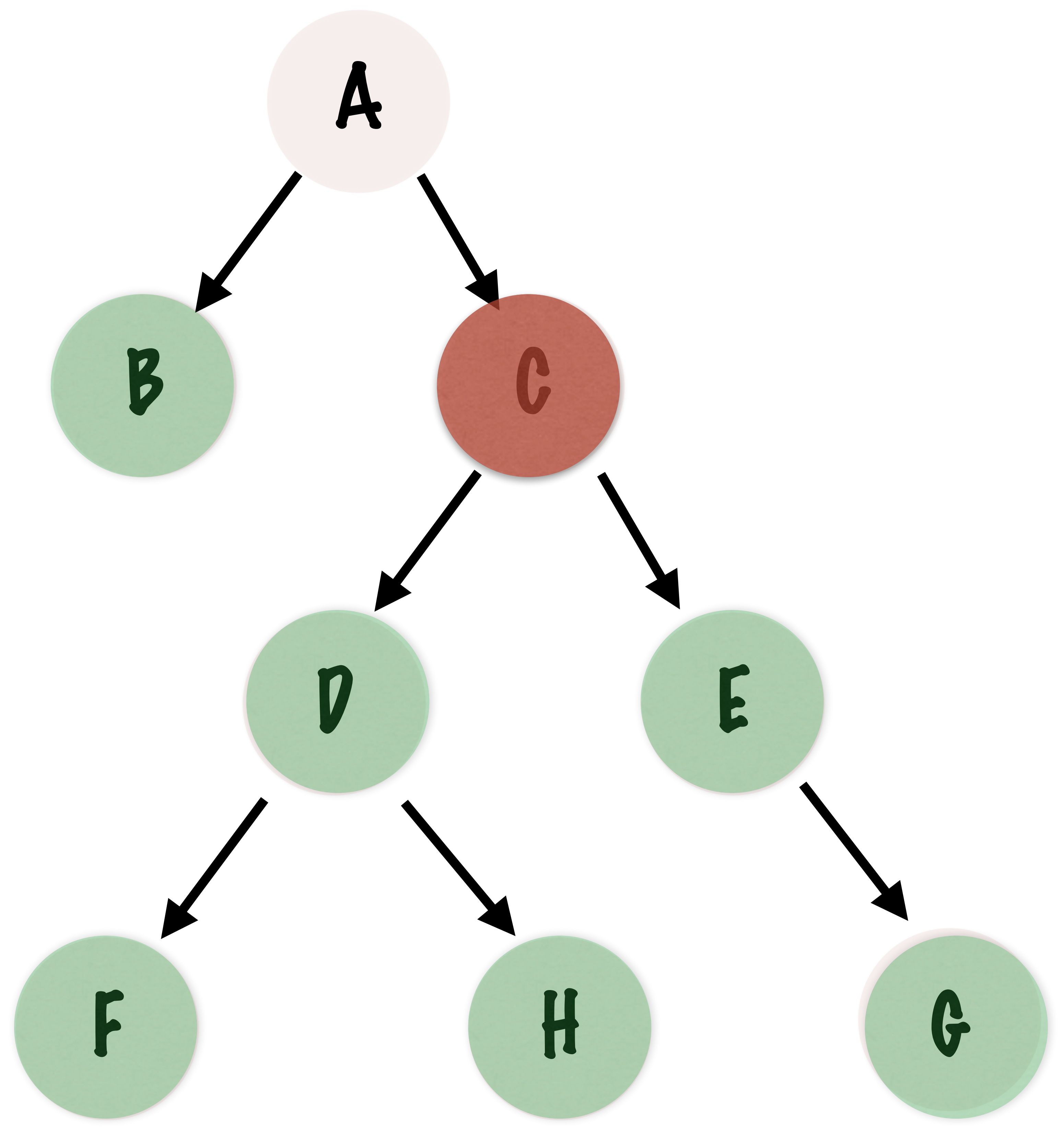


B->F->H->D->G

POST-ORDER TRAVERSAL

BOTH THE LEFT AND RIGHT SUBTREES OF NODE C ARE NOW DONE

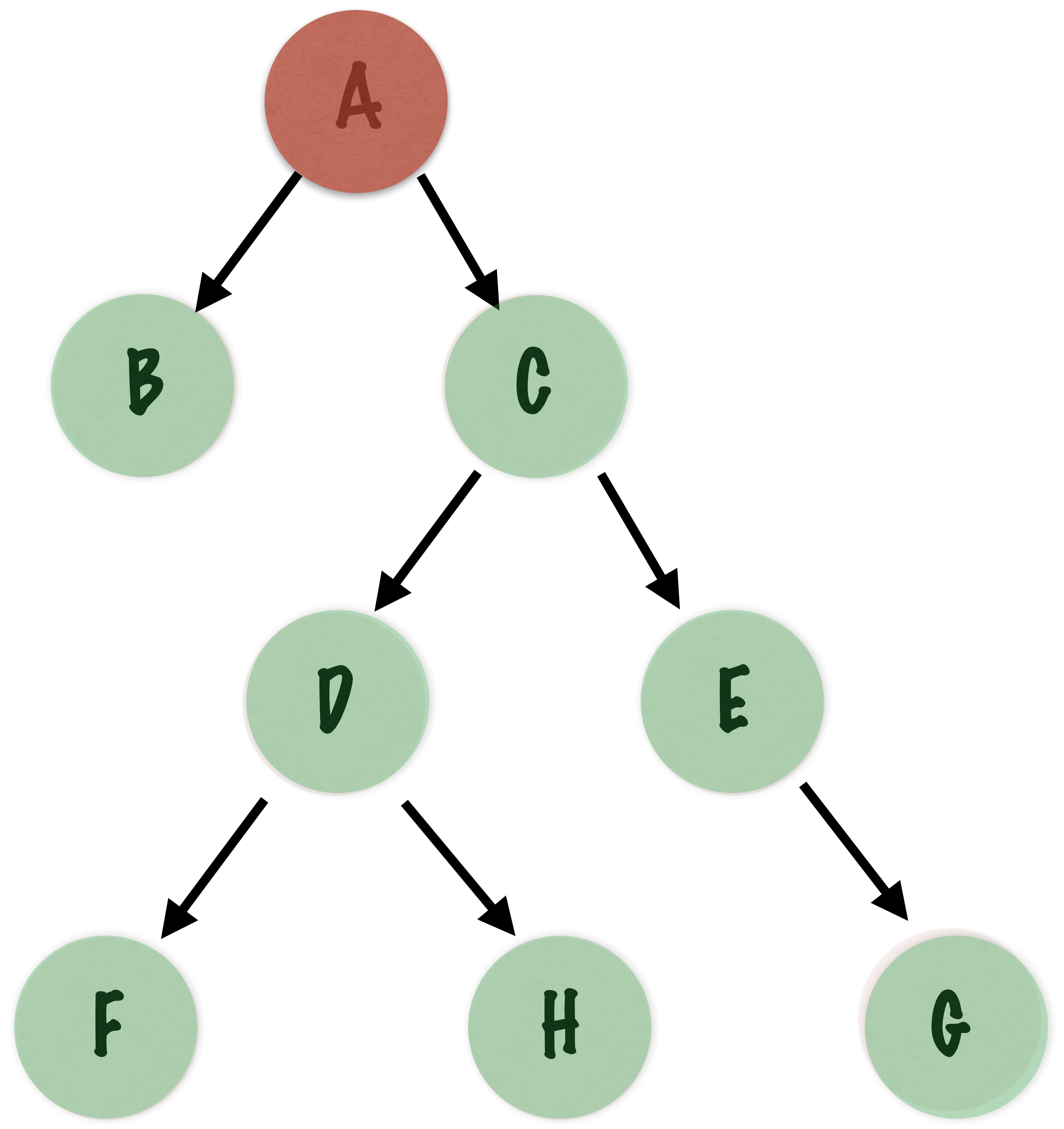
C CAN FINALLY BE PROCESSED



B->F->H->D->G->E

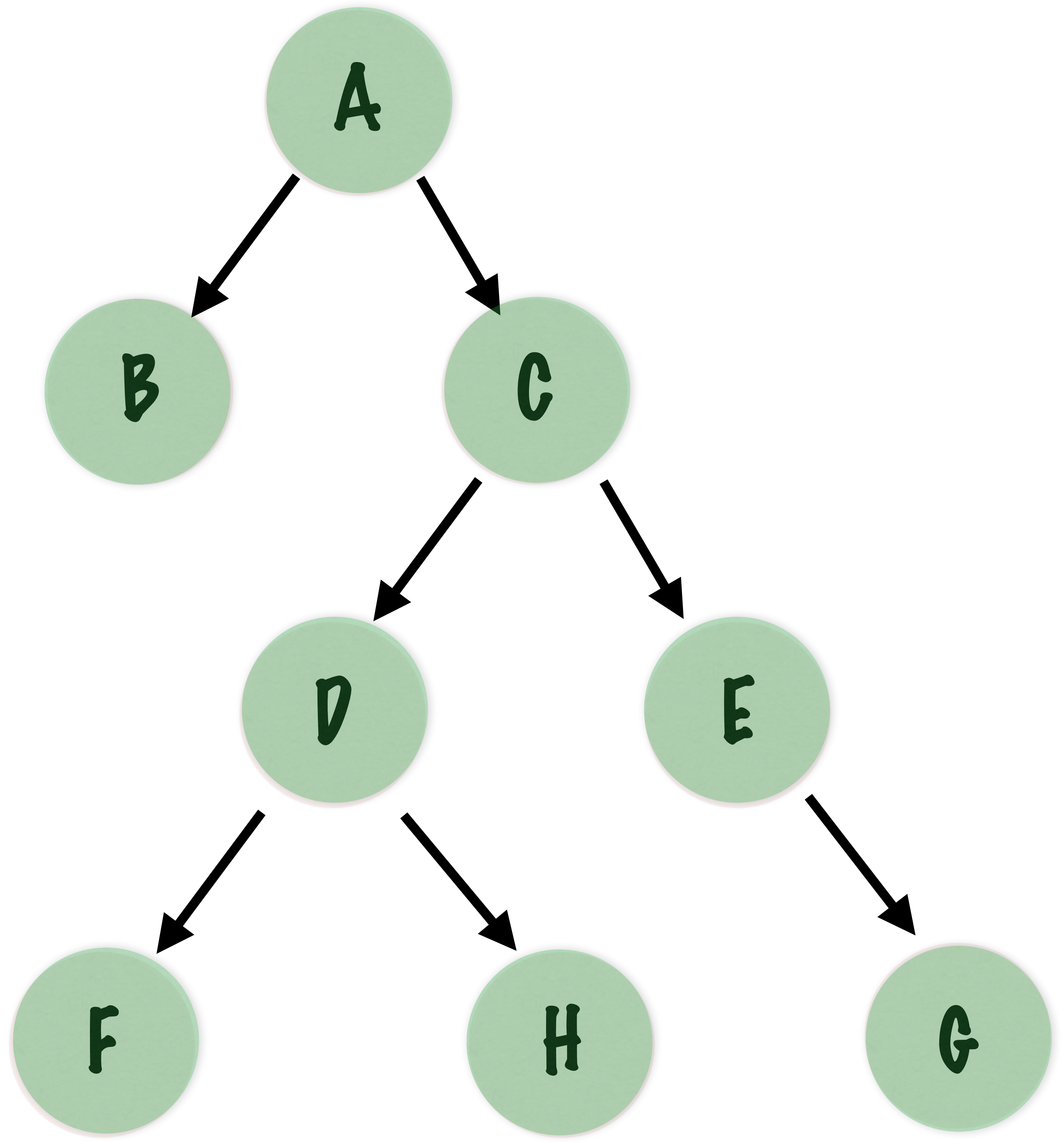
POST-ORDER TRAVERSAL

THE ROOT NODE IS PROCESSED LAST



B->F->H->D->G->E->C

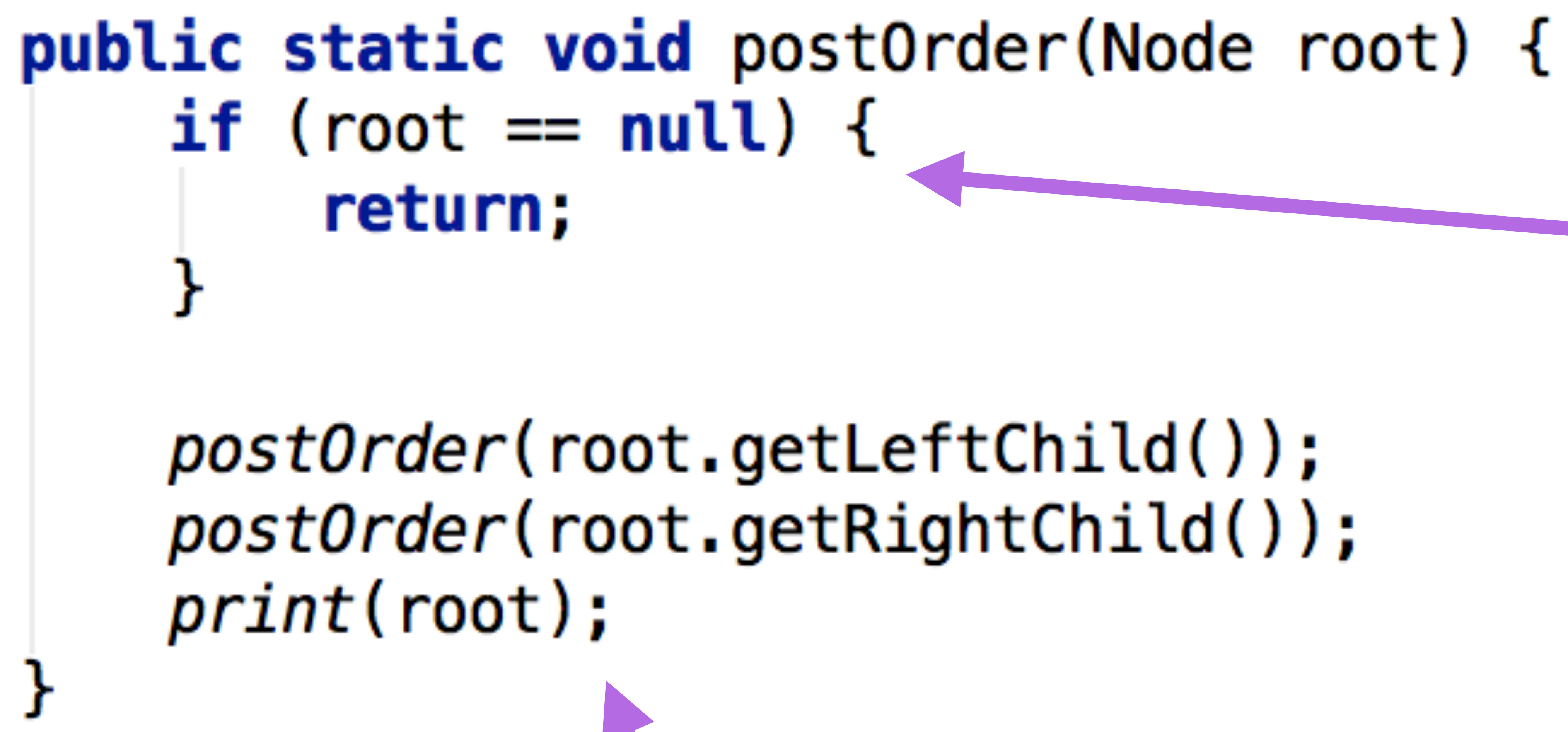
POST-ORDER TRAVERSAL



ALL NODES HAVE BEEN VISITED!

B->F->H->D->G->E->C->A

POST-ORDER TRAVERSAL CODE



```java
public static void postOrder(Node root) {
    if (root == null) {
        return;
    }

    postOrder(root.getLeftChild());
    postOrder(root.getRightChild());
    print(root);
}
```

BASE CASE - NOTHING TO TRAVERSE

PROCESS THE LEFT AND RIGHT SUBTREE BEFORE PROCESSING THE NODE ITSELF