Binary Trees

Written by Rachel Singh, last updated Sep 30, 2018

About

Binary Trees, especially Binary Search Trees, are a common structure you will see in Computer Science.

Topics

1. Binary Trees

2. Tree Traversal

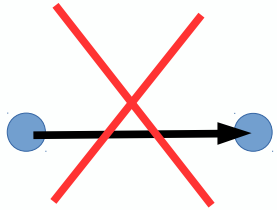
3. Binary Search Trees

Binary Trees

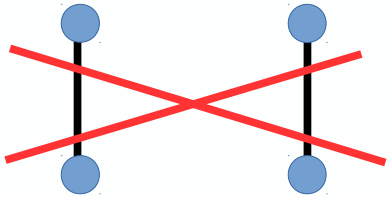
1. Binary Trees

Review: A **tree** is a simple, connected graph with no cycles.

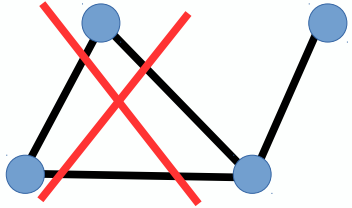
- Simple means not directional



- Connected means that, for any two nodes in the graph, there is a path between those nodes. (No nodes broken off by themselves.)



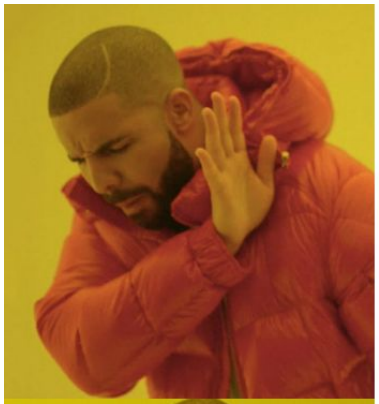
- A cycle is when you have a circuit that begins and ends at the same node and the start/end node is the only repeated node.

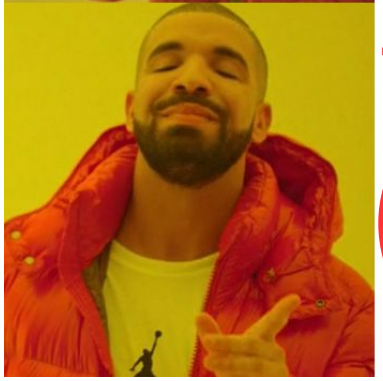


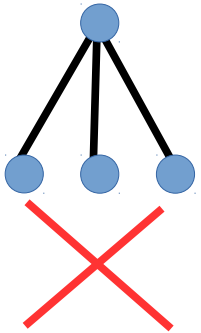
Notes

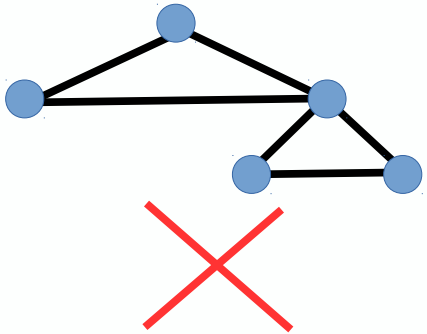
1. Binary Trees

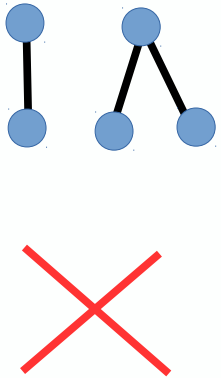
A **binary tree** is a **tree** where each node has **at most two child nodes**, and the tree has a **root node**.

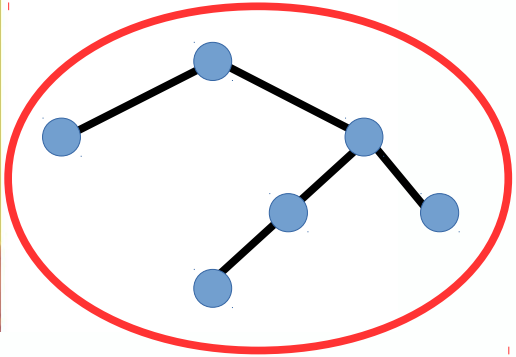










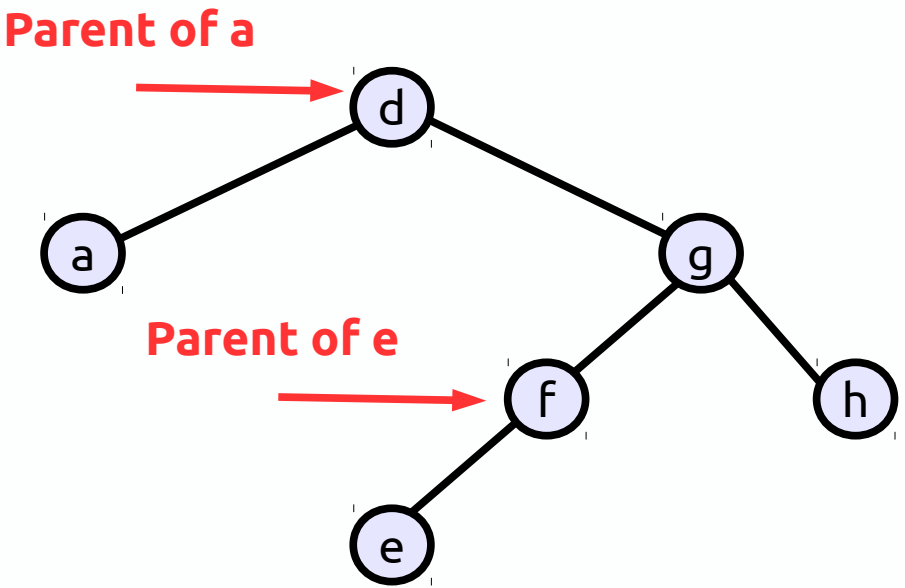


Notes

A binary tree is a tree where each node has at most two child nodes, and the tree has a root node.

1. Binary Trees

The parent of node *n* is the node immediately above it, between the path from *n* to the root. The only node without a parent is the root. :'(



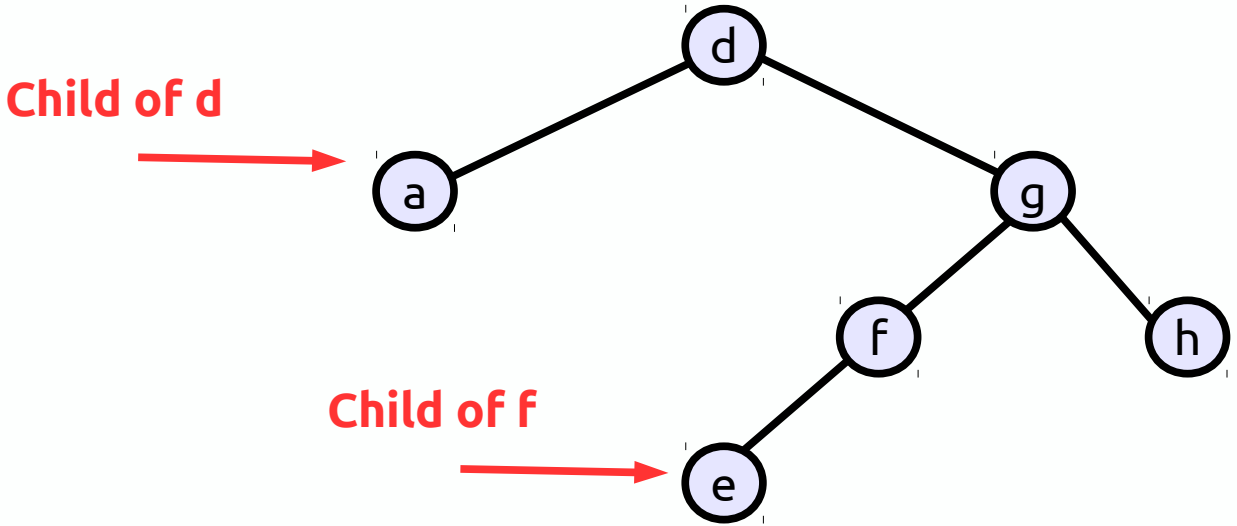
Notes

A binary tree is a tree where each node has at most two child nodes, and the tree has a root node.

The parent of node *n* is the node directly above *n*.

1. Binary Trees

The child node of *n* is a node immediately below *n*.
A node of a binary tree can have 0, 1, or 2 children.



Notes

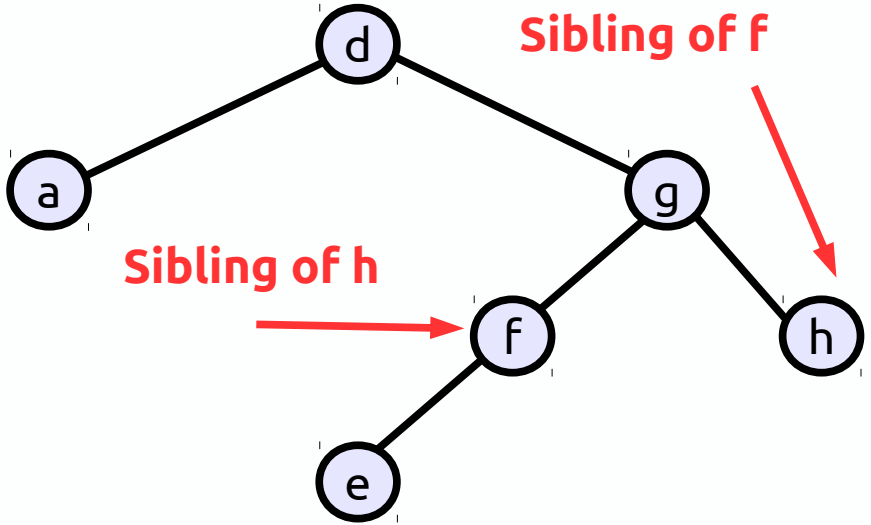
A binary tree is a tree where each node has at most two child nodes, and the tree has a root node.

The parent of node *n* is the node directly above *n*.

The child of node *n* is the node immediately below *n*.

1. Binary Trees

The sibling of node *n* is another node that shares the same parent as *n*.



Notes

A binary tree is a tree where each node has at most two child nodes, and the tree has a root node.

The parent of node *n* is the node directly above *n*.

The child of node *n* is the node immediately below *n*.

The sibling of node *n* is a node that shares the same parent as *n*.

1. Binary Trees

My favorite binary tree, showing how to get letters/numbers in Morse Code.

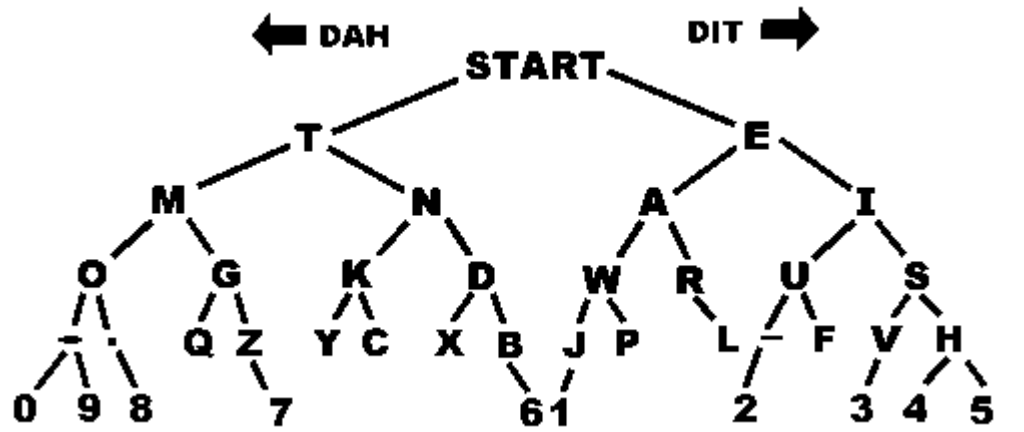


Image from http://www.cranburyscouts.org/MorseTree.htm

Notes

A binary tree is a tree where each node has at most two child nodes, and the tree has a root node.

The parent of node *n* is the node directly above *n*.

The child of node *n* is the node immediately below *n*.

The sibling of node *n* is a node that shares the same parent as *n*.

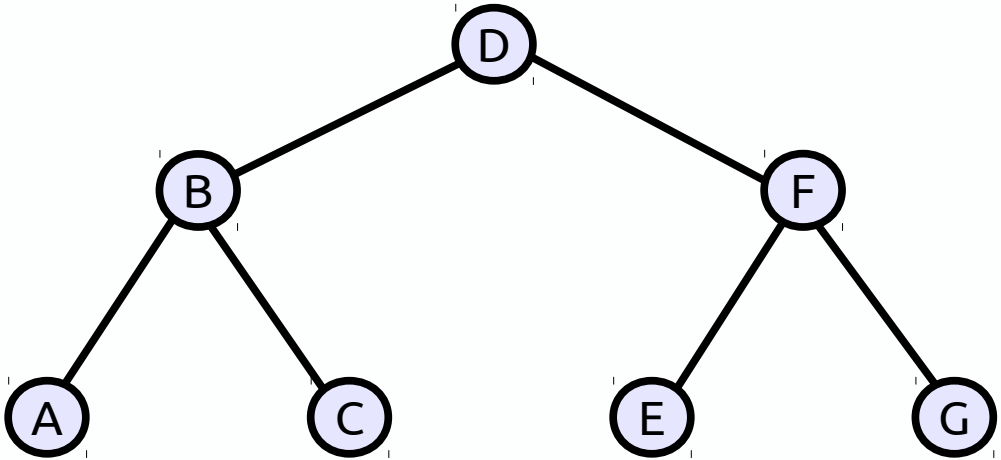
Tree Traversal

2. Tree Traversal

Since a tree is not a **linear structure**, what order do you display its contents?

There are three main methods to traverse a tree…

1. Preorder
2. Inorder
3. Postorder



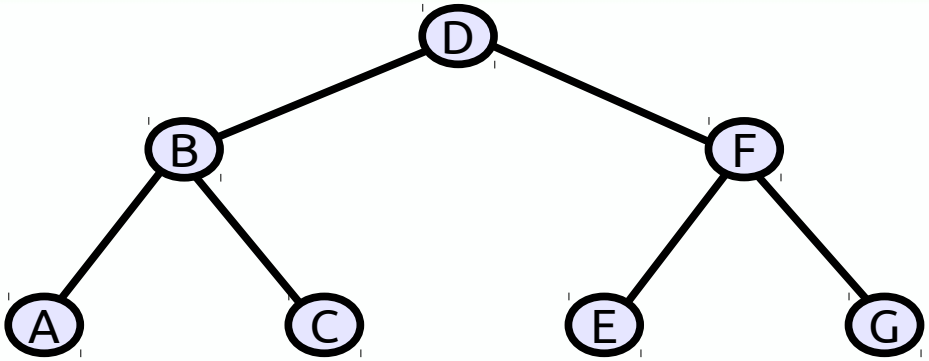
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Traversal: Begin at the root node. For each node…

```
PreorderTraverse( currentNode )  {
        Display currentNode
        PreorderTraverse( currentNode→leftChild )
        PreorderTraverse( currentNode→rightChild )
}
```

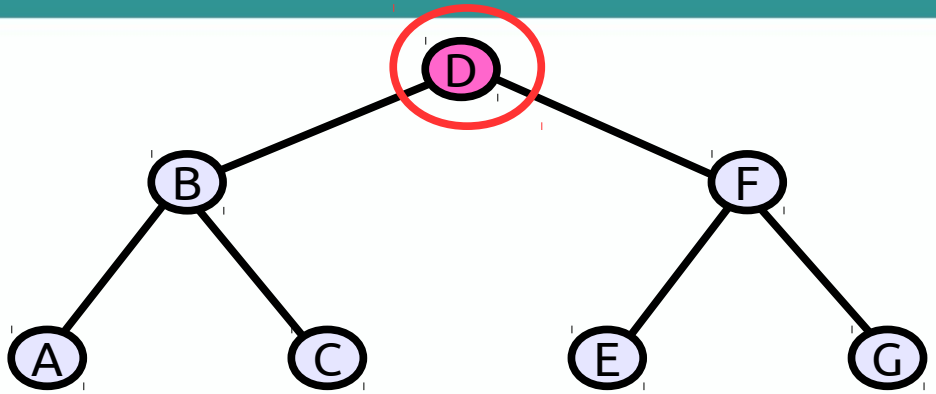
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
1. Begin at D, Display “D”. Go left.

Output: D

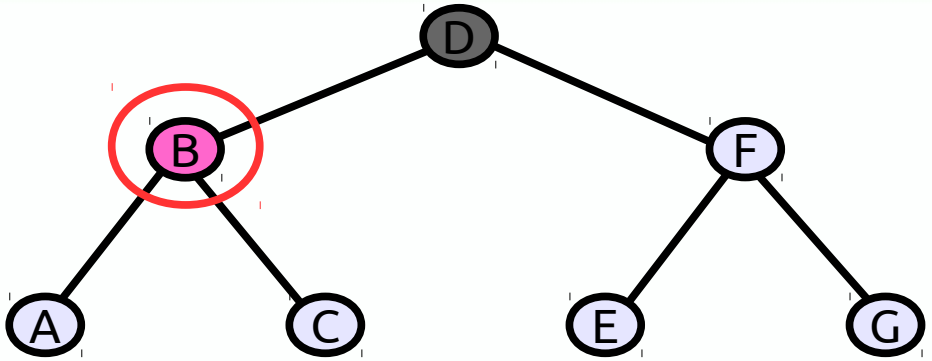
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
1. Begin at D, Display “D”. Go left.
2. At B. Display “B”. Go left.

Output: DB

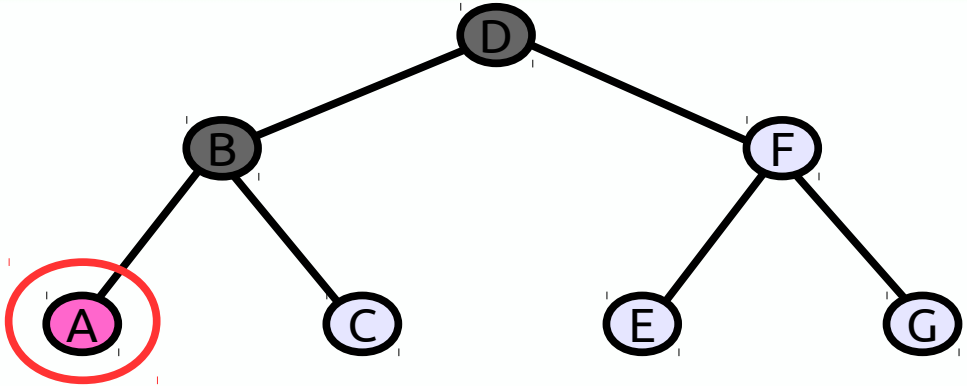
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
1. Begin at D, Display “D”. Go left.
2. At B. Display “B”. Go left.
3. At A. Display “A”. No left child. No right child. Go back.

Output: DBA

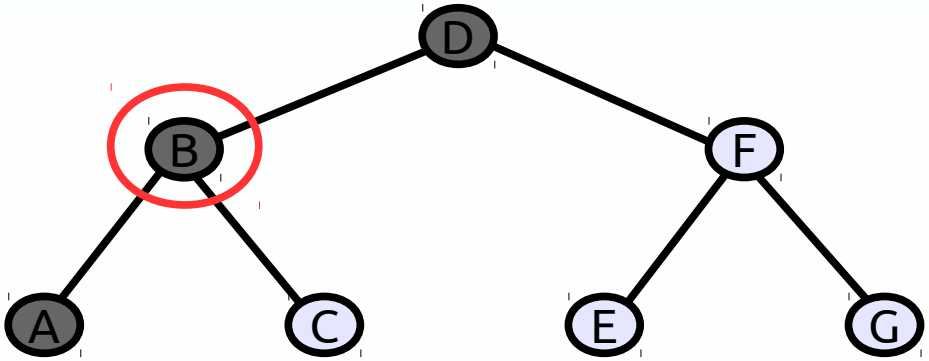
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
1. Begin at D, Display “D”. Go left.
2. At B. Display “B”. Go left.
3. At A. Display “A”. No left child. No right child. Go back.
4. At B. Already went left. Go right.

Output: DBA

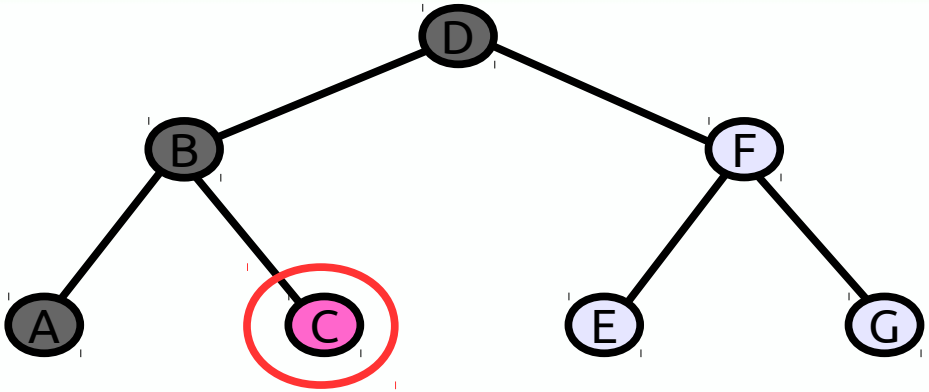
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
1. Begin at D, Display “D”. Go left.
2. At B. Display “B”. Go left.
3. At A. Display “A”. No left child. No right child. Go back.
4. At B. Already went left. Go right.
5. At C. Display “C”. No left child. No right child. Go back.

Output: DBAC

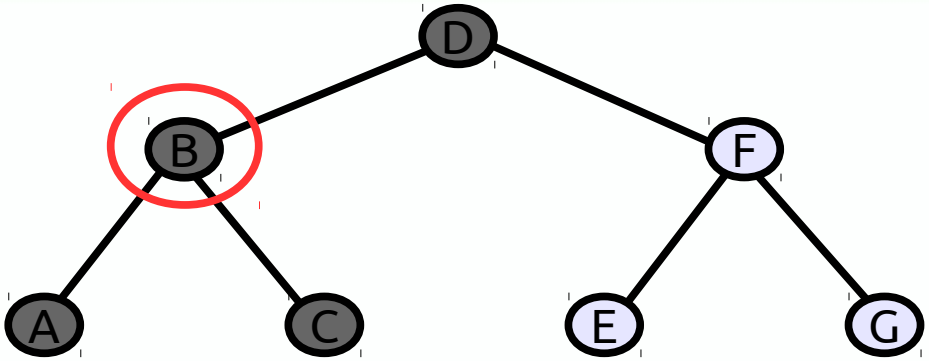
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
1. Begin at D, Display “D”. Go left.
2. At B. Display “B”. Go left.
3. At A. Display “A”. No left child. No right child. Go back.
4. At B. Already went left. Go right.
5. At C. Display “C”. No left child. No right child. Go back.
6. At B. Already went left. Already went right. Go back.

Output: DBAC

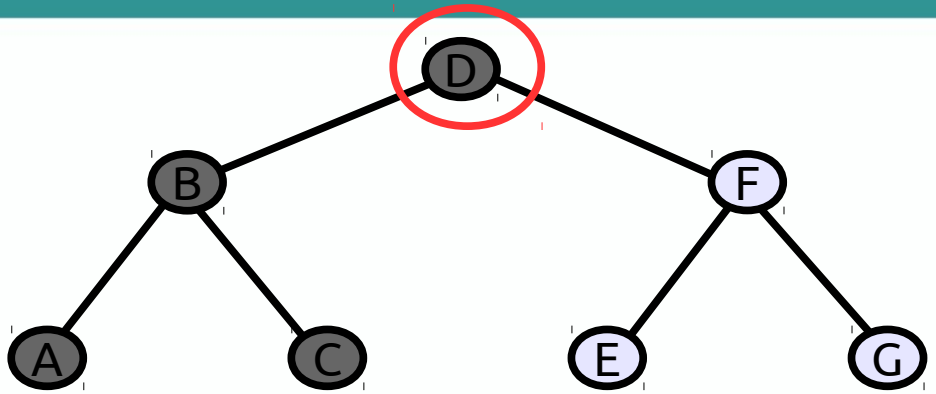
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
2. At B. Display “B”. Go left.
3. At A. Display “A”. No left child. No right child. Go back.
4. At B. Already went left. Go right.
5. At C. Display “C”. No left child. No right child. Go back.
6. At B. Already went left. Already went right. Go back.
7. At D. Already went left. Go right.

Output: DBAC

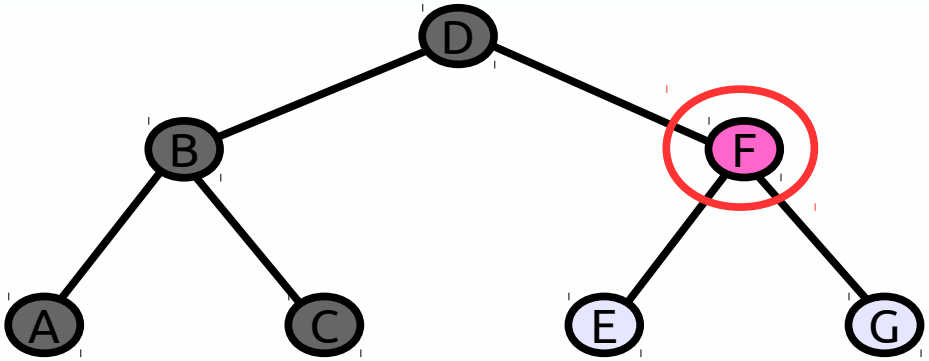
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
3. At A. Display “A”. No left child. No right child. Go back.
4. At B. Already went left. Go right.
5. At C. Display “C”. No left child. No right child. Go back.
6. At B. Already went left. Already went right. Go back.
7. At D. Already went left. Go right.
8. At F. Display “F”. Go left.

Output: DBACF

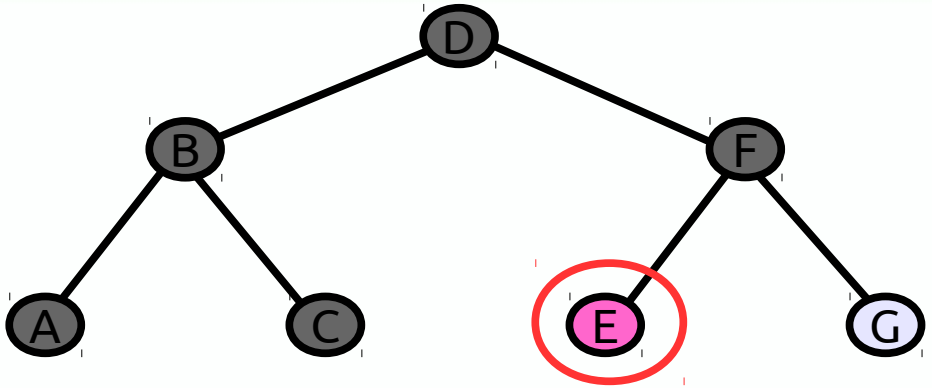
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
4. At B. Already went left. Go right.
5. At C. Display “C”. No left child. No right child. Go back.
6. At B. Already went left. Already went right. Go back.
7. At D. Already went left. Go right.
8. At F. Display “F”. Go left.
9. At E. Display “E”. No left child. No right child. Go back.

Output: DBACFE

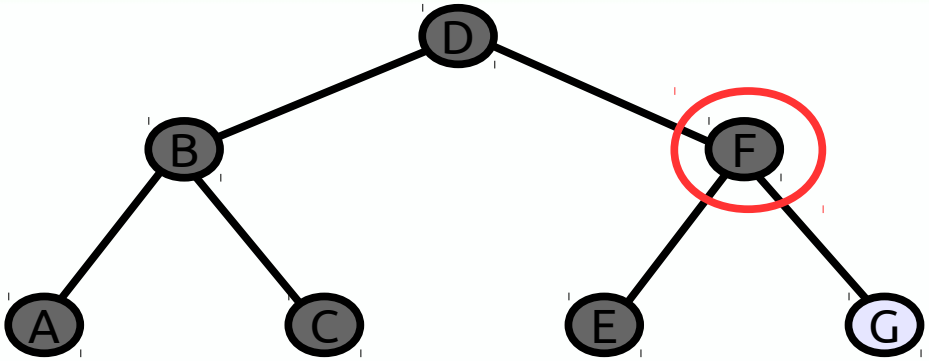
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
5. At C. Display “C”. No left child. No right child. Go back.
6. At B. Already went left. Already went right. Go back.
7. At D. Already went left. Go right.
8. At F. Display “F”. Go left.
9. At E. Display “E”. No left child. No right child. Go back.
10. At F. Already went left. Go right.

Output: DBACFE

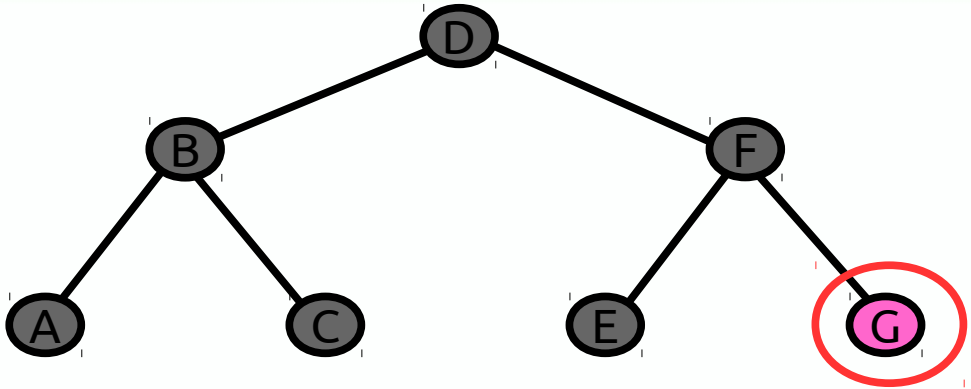
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
6. At B. Already went left. Already went right. Go back.
7. At D. Already went left. Go right.
8. At F. Display “F”. Go left.
9. At E. Display “E”. No left child. No right child. Go back.
10. At F. Already went left. Go right.
11. At G. Display “G”. No left child. No right child. Go back.

Output: DBACFEG

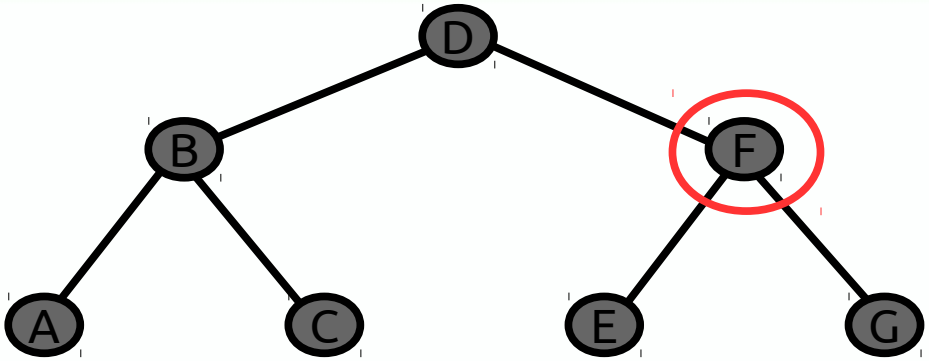
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
7. At D. Already went left. Go right.
8. At F. Display “F”. Go left.
9. At E. Display “E”. No left child. No right child. Go back.
10. At F. Already went left. Go right.
11. At G. Display “G”. No left child. No right child. Go back.
12. At F. Already went left. Already went right. Go back.

Output: DBACFEG

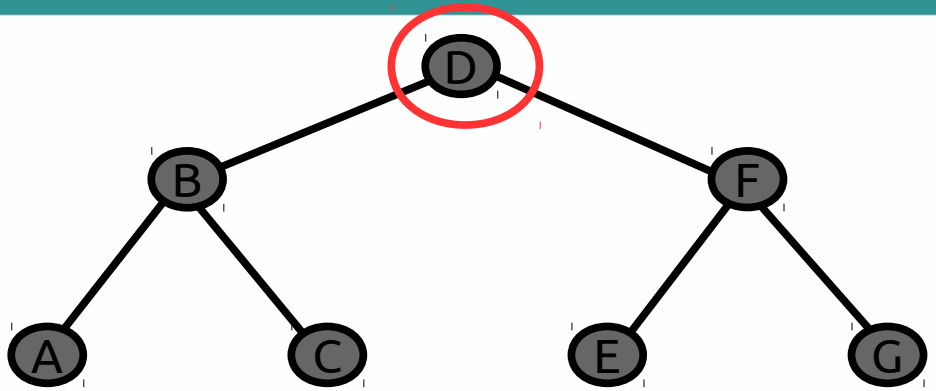
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
8. At F. Display “F”. Go left.
9. At E. Display “E”. No left child. No right child. Go back.
10. At F. Already went left. Go right.
11. At G. Display “G”. No left child. No right child. Go back.
12. At F. Already went left. Already went right. Go back.
13. At D. Already went left. Already went right. Go back.

DONE WITH TRAVERSAL!

Output: DBACFEG

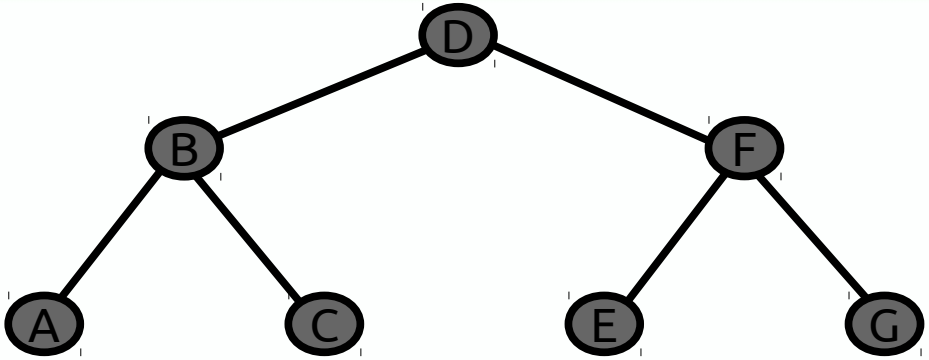
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Traversal: Begin at the root node. For each node…

```
PreorderTraverse( currentNode )  {
        Display currentNode
        PreorderTraverse( currentNode→leftChild )
        PreorderTraverse( currentNode→rightChild )
}
```

So the preorder traversal gives us: DBACFEG

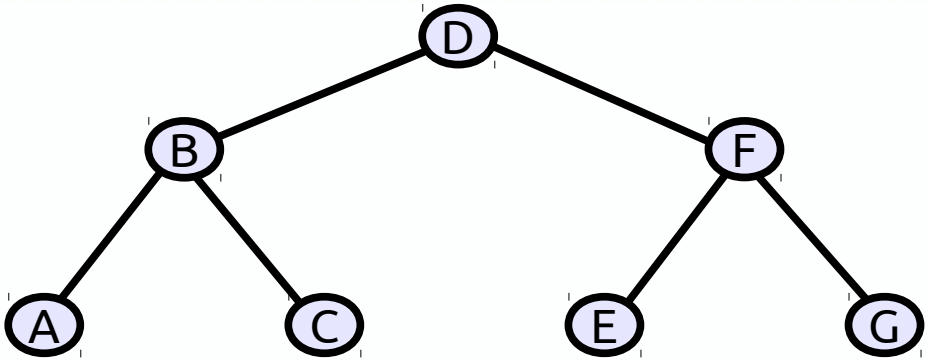
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



In-order Traversal: Begin at the root node. For each node…

```
InorderTraverse( currentNode )  {
        InorderTraverse( currentNode→leftChild )
        Display currentNode
        InorderTraverse( currentNode→rightChild )
}
```

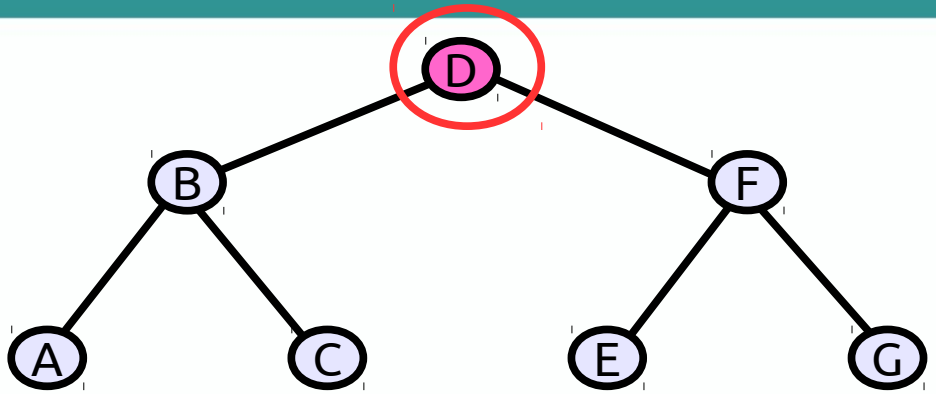
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



In-order Example:
1. Begin at D. First, go left.

Output:

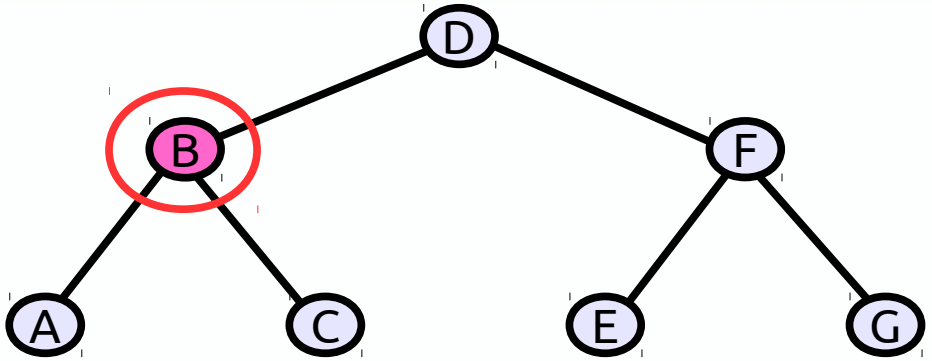
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



In-order Example:
1. Begin at D. First, go left.
2. At B. First, go left.

Output:

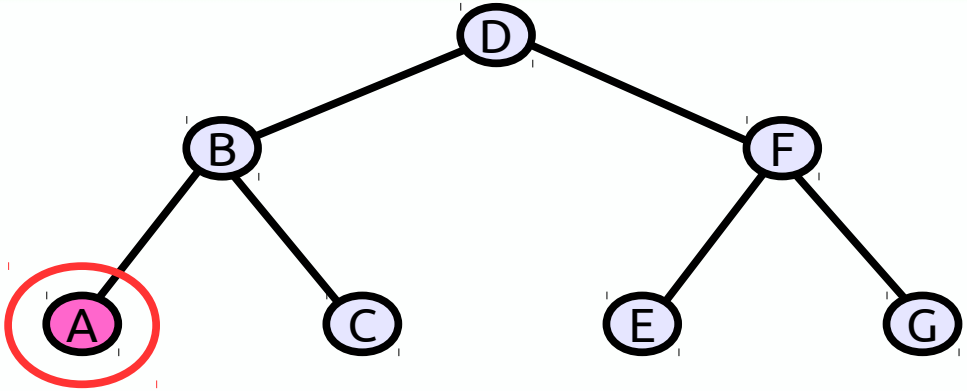
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



In-order Example:
1. Begin at D. First, go left.
2. At B. First, go left.
3. At A. No left child. Display “A”. No right child. Go back.

Output: A

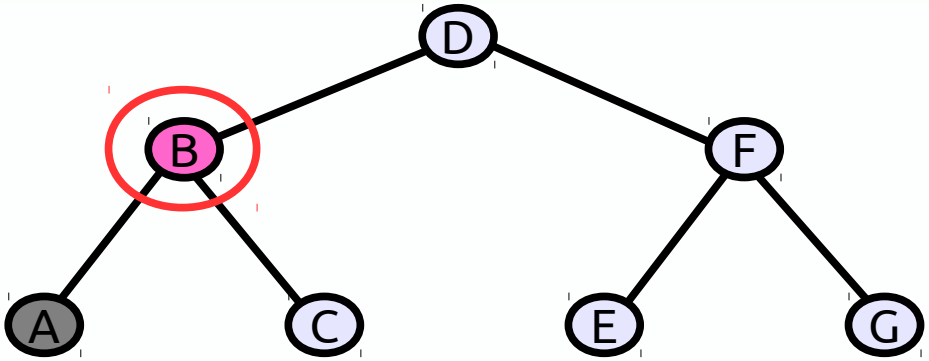
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



In-order Example:
1. Begin at D. First, go left.
2. At B. First, go left.
3. At A. No left child. Display “A”. No right child. Go back.
4. At B. Already went left. Display “B”. Go right.

Output: AB

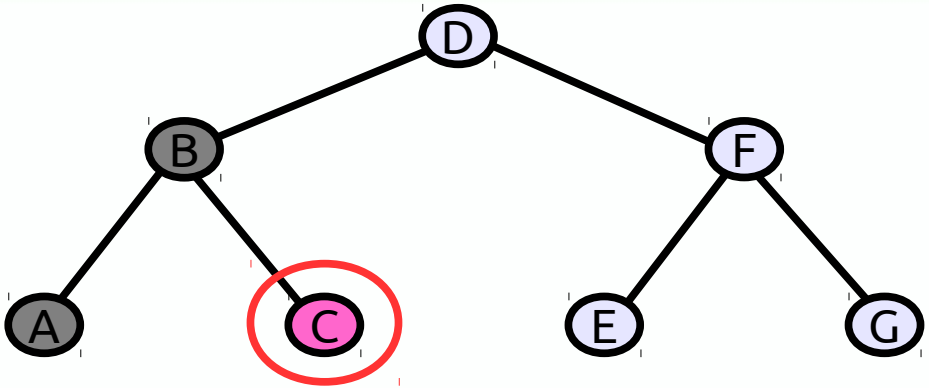
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



In-order Example:
1. Begin at D. First, go left.
2. At B. First, go left.
3. At A. No left child. Display “A”. No right child. Go back.
4. At B. Already went left. Display “B”. Go right.
5. At C. No left child. Display “C”. No right child. Go back.

Output: ABC

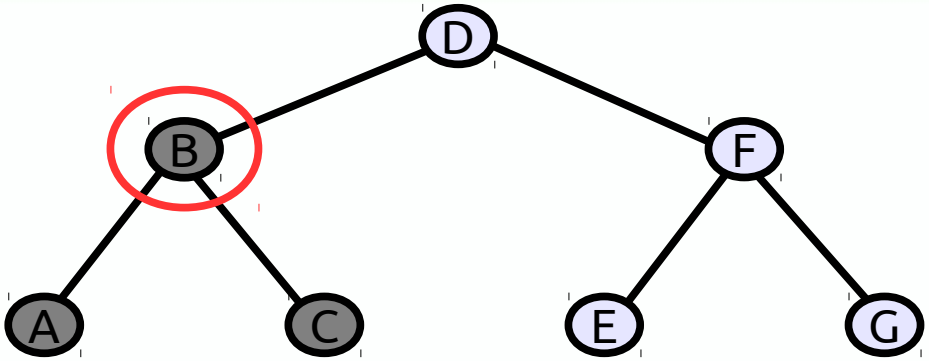
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



In-order Example:
1. Begin at D. First, go left.
2. At B. First, go left.
3. At A. No left child. Display “A”. No right child. Go back.
4. At B. Already went left. Display “B”. Go right.
5. At C. No left child. Display “C”. No right child. Go back.
6. At B. Already went left. Already went right. Go back.

Output: ABC

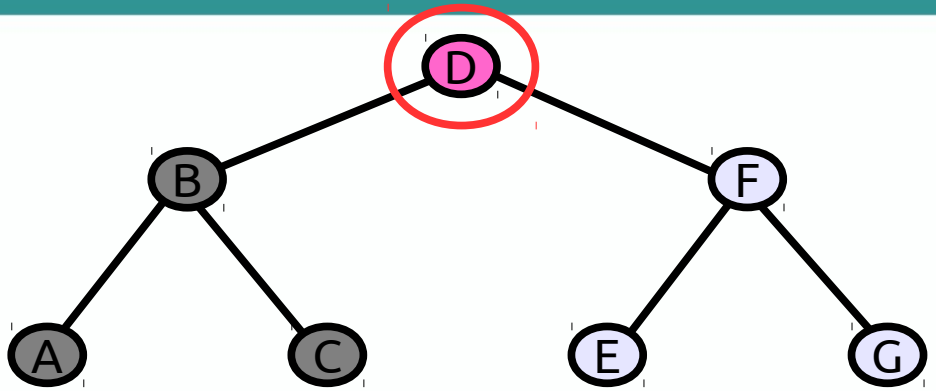
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



In-order Example:
2. At B. First, go left.
3. At A. No left child. Display “A”. No right child. Go back.
4. At B. Already went left. Display “B”. Go right.
5. At C. No left child. Display “C”. No right child. Go back.
6. At B. Already went left. Already went right. Go back.
7. At D. Already went left. Display “D”. Go right.

Output: ABCD

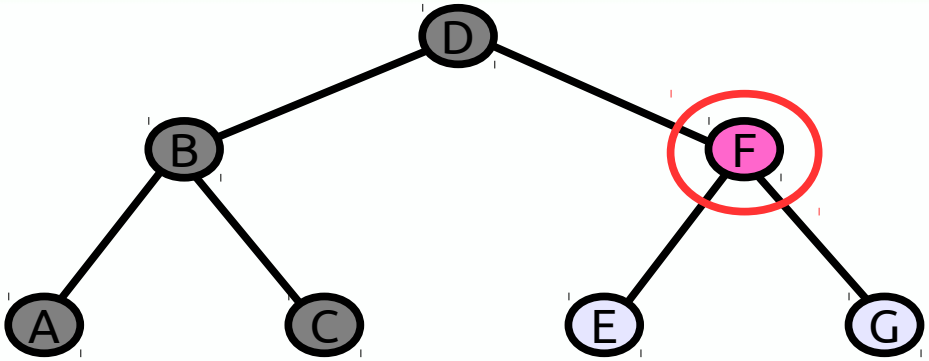
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



In-order Example:
3. At A. No left child. Display “A”. No right child. Go back.
4. At B. Already went left. Display “B”. Go right.
5. At C. No left child. Display “C”. No right child. Go back.
6. At B. Already went left. Already went right. Go back.
7. At D. Already went left. Display “D”. Go right.
8. At F. First, go left.

Output: ABCD

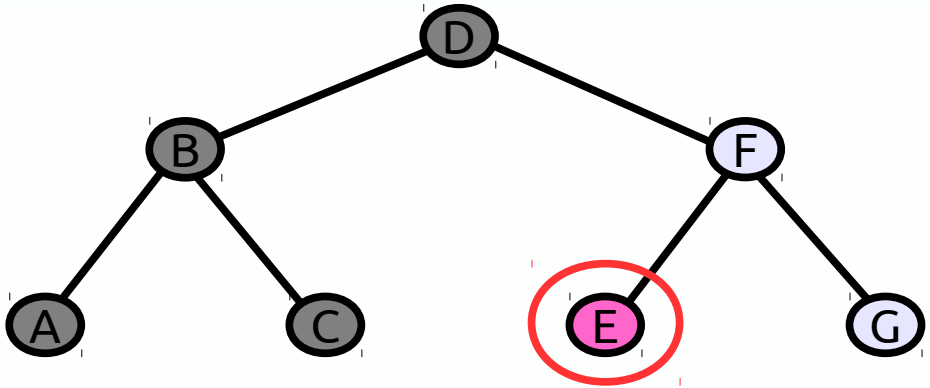
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



In-order Example:
4. At B. Already went left. Display “B”. Go right.
5. At C. No left child. Display “C”. No right child. Go back.
6. At B. Already went left. Already went right. Go back.
7. At D. Already went left. Display “D”. Go right.
8. At F. First, go left.
9. At E. No left child. Display “E”. No right child. Go back.

Output: ABCDE

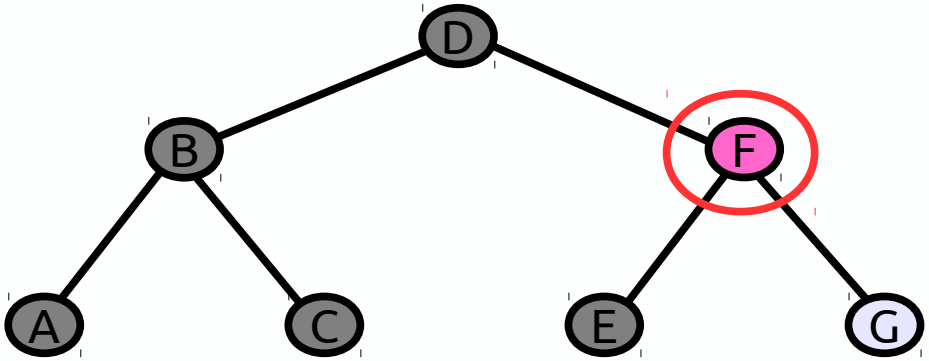
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



In-order Example:
5. At C. No left child. Display “C”. No right child. Go back.
6. At B. Already went left. Already went right. Go back.
7. At D. Already went left. Display “D”. Go right.
8. At F. First, go left.
9. At E. No left child. Display “E”. No right child. Go back.
10. At F. Already went left. Display “F”. Go right.

Output: ABCDEF

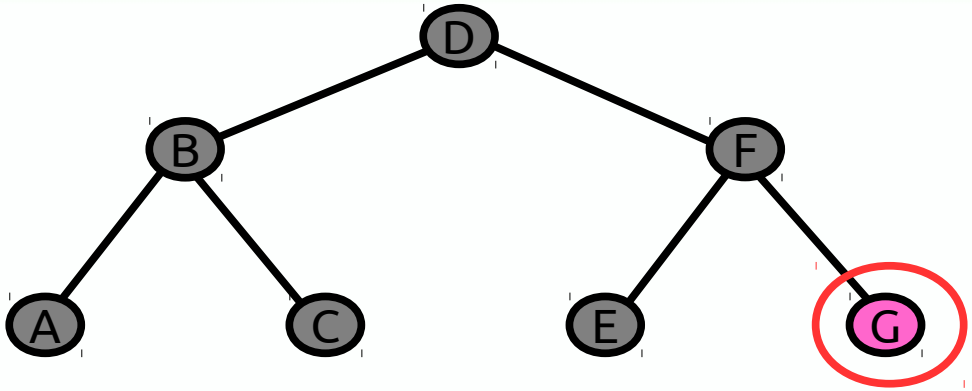
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



In-order Example:
6. At B. Already went left. Already went right. Go back.
7. At D. Already went left. Display “D”. Go right.
8. At F. First, go left.
9. At E. No left child. Display “E”. No right child. Go back.
10. At F. Already went left. Display “F”. Go right.
11. At G. No left child. Display “G”. No right child. Go back.

Output: ABCDEFG

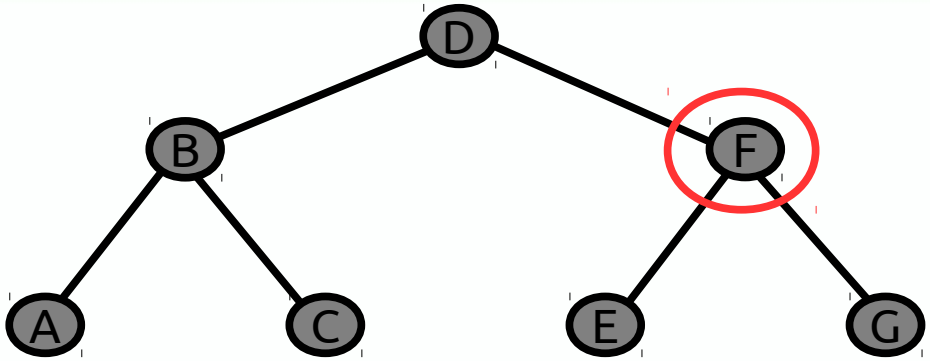
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



In-order Example:
7. At D. Already went left. Display “D”. Go right.
8. At F. First, go left.
9. At E. No left child. Display “E”. No right child. Go back.
10. At F. Already went left. Display “F”. Go right.
11. At G. No left child. Display “G”. No right child. Go back.
12. At F. Already went left. Already went right. Go back.

Output: ABCDEFG

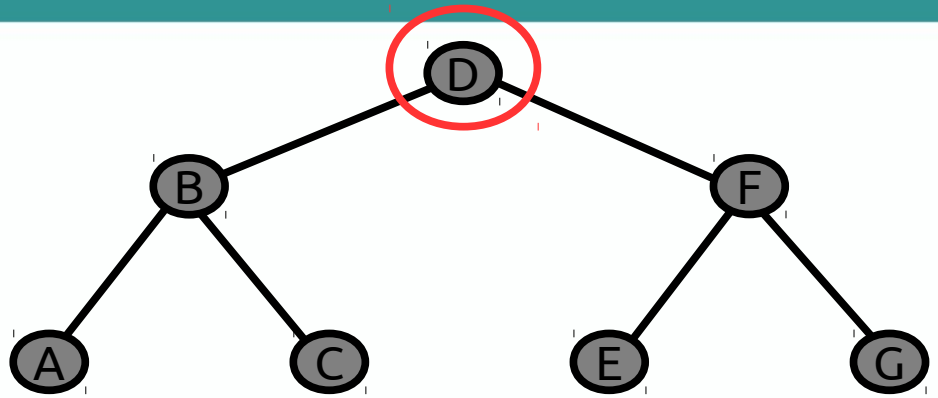
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



In-order Example:
8. At F. First, go left.
9. At E. No left child. Display “E”. No right child. Go back.
10. At F. Already went left. Display “F”. Go right.
11. At G. No left child. Display “G”. No right child. Go back.
12. At F. Already went left. Already went right. Go back.
13. At D. Already went left. Already went right. Go back.

DONE WITH TRAVERSAL.

Output: ABCDEFG

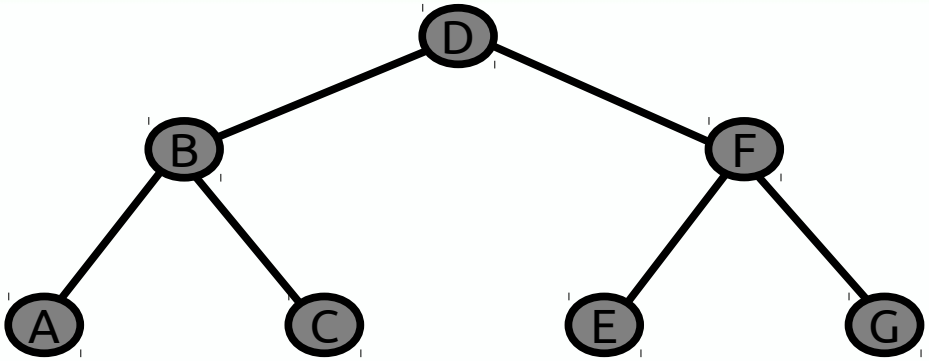
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



In-order Traversal: Begin at the root node. For each node…

```
InorderTraverse( currentNode )  {
        InorderTraverse( currentNode→leftChild )
        Display currentNode
        InorderTraverse( currentNode→rightChild )
}
```

So the inorder traversal gives us: ABCDEFG

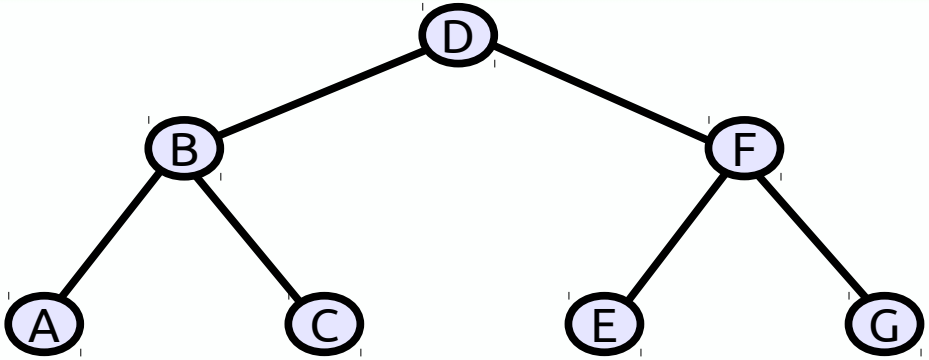
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Post-order Traversal: Begin at the root node. For each node…

```
PostorderTraverse( currentNode )  {
      PostorderTraverse( currentNode→leftChild )
      PostorderTraverse( currentNode→rightChild )
      Display currentNode
}
```

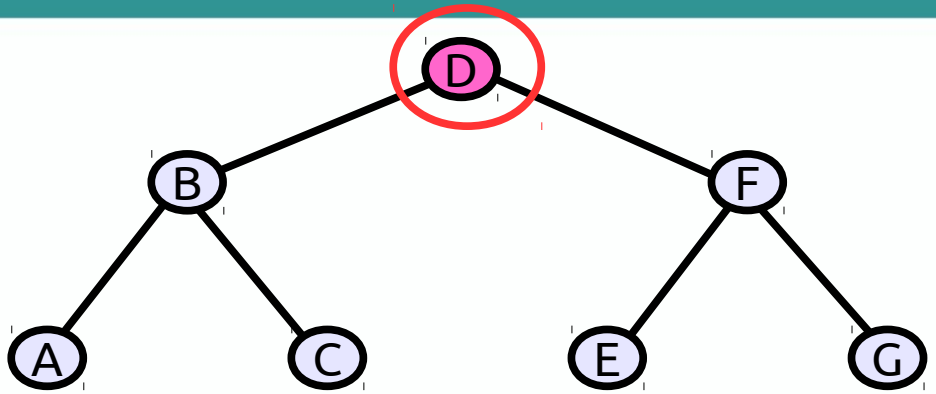
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
1. Begin at D. Go left.

Output:

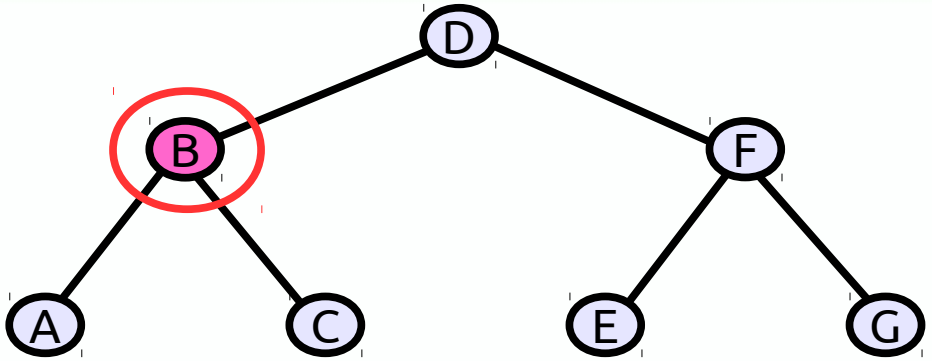
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
1. Begin at D. Go left.
2. At B. Go left.

Output:

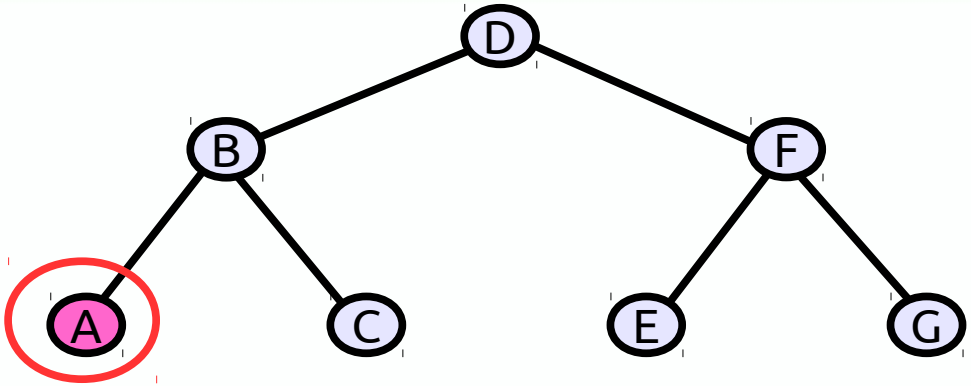
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
1. Begin at D. Go left.
2. At B. Go left.
3. At A. No left child. No right child. Display “A”. Go back.

Output: A

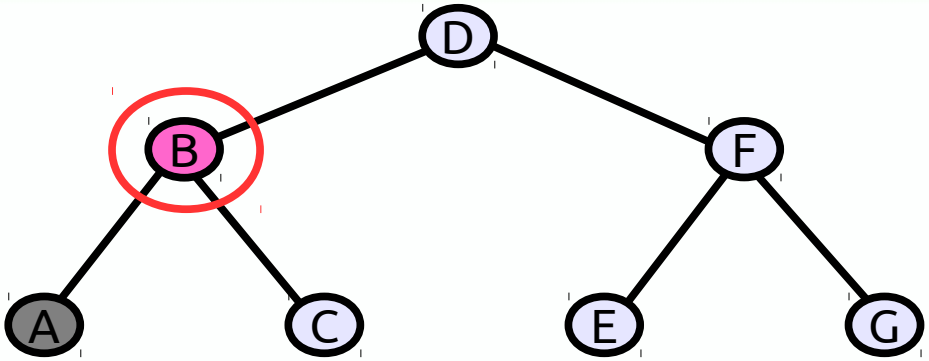
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
1. Begin at D. Go left.
2. At B. Go left.
3. At A. No left child. No right child. Display “A”. Go back.
4. At B. Already did left. Go right.

Output: A

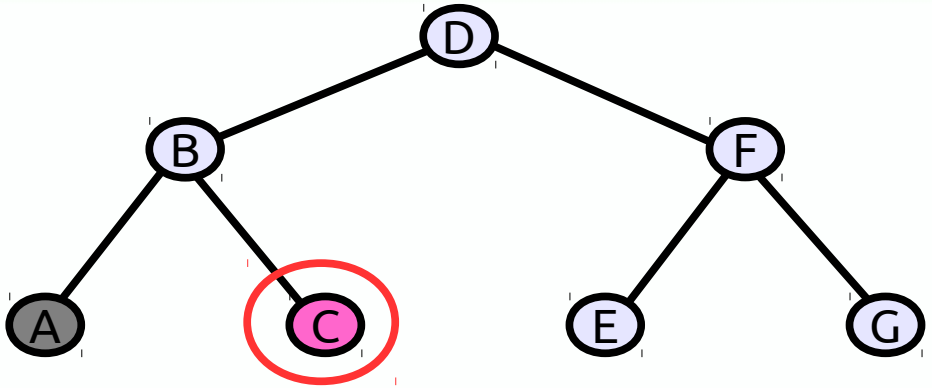
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
1. Begin at D. Go left.
2. At B. Go left.
3. At A. No left child. No right child. Display “A”. Go back.
4. At B. Already did left. Go right.
5. At C. No left child. No right child. Display “C”. Go back.

Output: AC

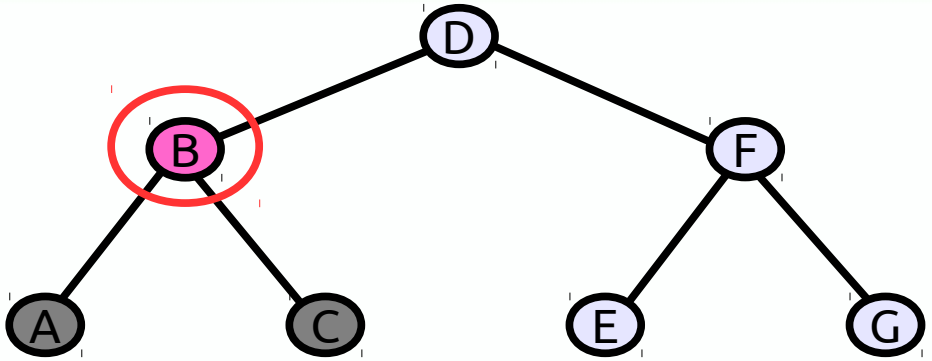
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
1. Begin at D. Go left.
2. At B. Go left.
3. At A. No left child. No right child. Display “A”. Go back.
4. At B. Already did left. Go right.
5. At C. No left child. No right child. Display “C”. Go back.
6. At B. Already did left. Already did right. Display “B”. Go back.

Output: ACB

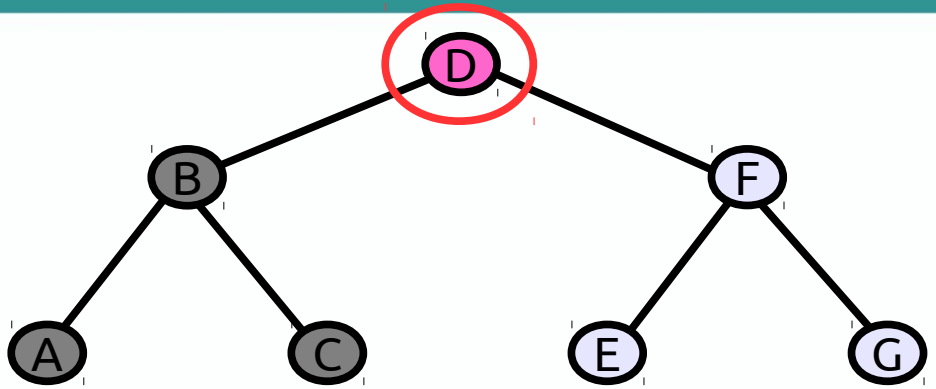
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
2. At B. Go left.
3. At A. No left child. No right child. Display “A”. Go back.
4. At B. Already did left. Go right.
5. At C. No left child. No right child. Display “C”. Go back.
6. At B. Already did left. Already did right. Display “B”. Go back.
7. At D. Already did left. Go right.

Output: ACB

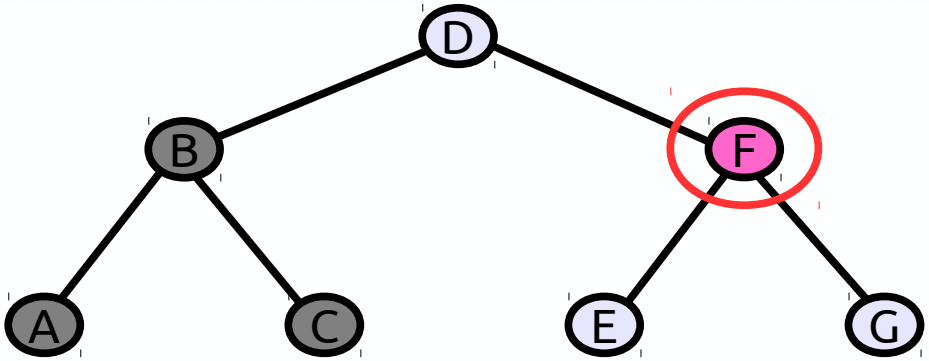
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
3. At A. No left child. No right child. Display “A”. Go back.
4. At B. Already did left. Go right.
5. At C. No left child. No right child. Display “C”. Go back.
6. At B. Already did left. Already did right. Display “B”. Go back.
7. At D. Already did left. Go right.
8. At F. Go left.

Output: ACB

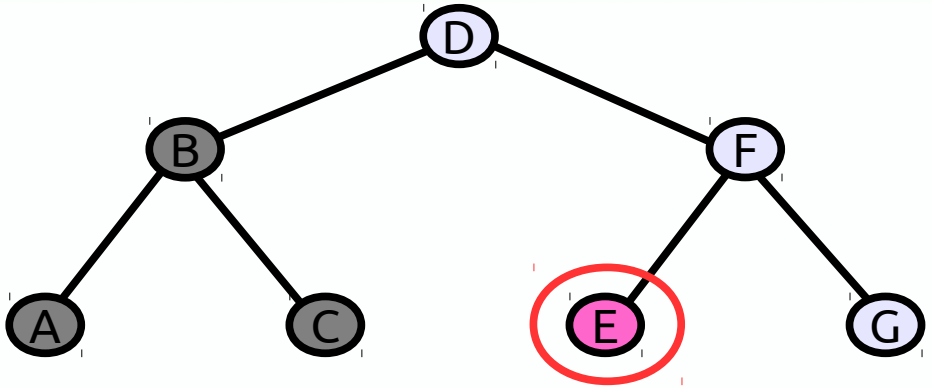
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
4. At B. Already did left. Go right.
5. At C. No left child. No right child. Display “C”. Go back.
6. At B. Already did left. Already did right. Display “B”. Go back.
7. At D. Already did left. Go right.
8. At F. Go left.
9. At E. No left child. No right child. Display “E”. Go back.

Output: ACBE

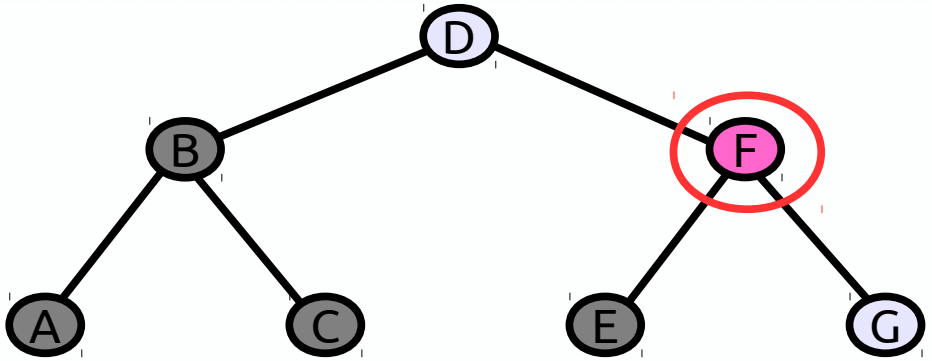
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
5. At C. No left child. No right child. Display “C”. Go back.
6. At B. Already did left. Already did right. Display “B”. Go back.
7. At D. Already did left. Go right.
8. At F. Go left.
9. At E. No left child. No right child. Display “E”. Go back.
10. At F. Already did left. Go right.

Output: ACBE

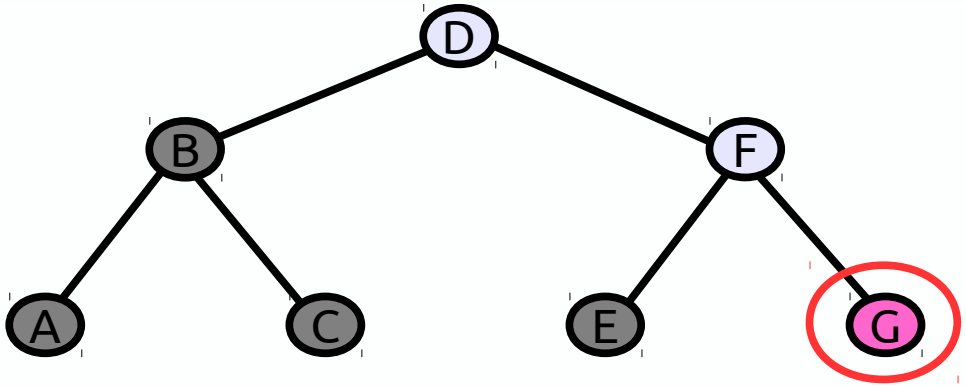
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
6. At B. Already did left. Already did right. Display “B”. Go back.
7. At D. Already did left. Go right.
8. At F. Go left.
9. At E. No left child. No right child. Display “E”. Go back.
10. At F. Already did left. Go right.
11. At G. No left child. No right child. Display “G”. Go back.

Output: ACBEG

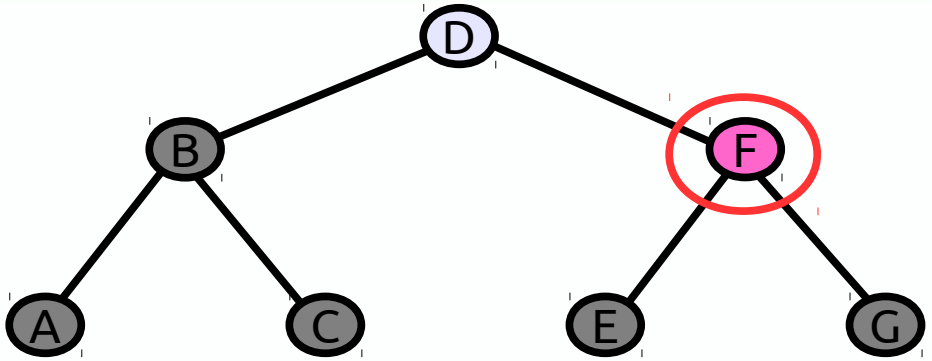
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
7. At D. Already did left. Go right.
8. At F. Go left.
9. At E. No left child. No right child. Display “E”. Go back.
10. At F. Already did left. Go right.
11. At G. No left child. No right child. Display “G”. Go back.
12. At F. Already did left. Already did right. Display “F”. Go back.

Output: ACBEGF

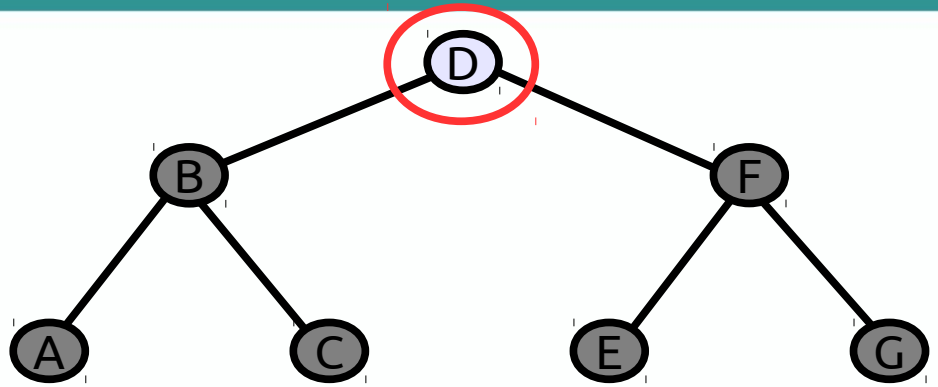
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Pre-order Example:
8. At F. Go left.
9. At E. No left child. No right child. Display “E”. Go back.
10. At F. Already did left. Go right.
11. At G. No left child. No right child. Display “G”. Go back.
12. At F. Already did left. Already did right. Display “F”. Go back.
13. At D. Already did left. Already did right. Display “D”. Go back.

DONE WITH TRAVERSAL!

Output: ACBEGFD

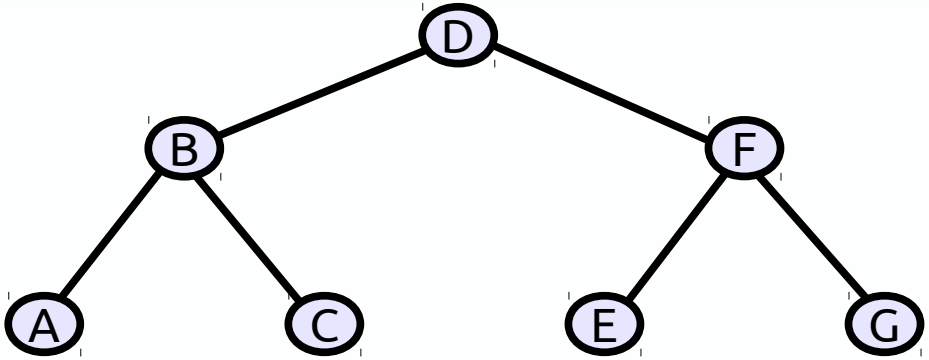
Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal



Post-order Traversal: Begin at the root node. For each node…

```
PostorderTraverse( currentNode )  {
      PostorderTraverse( currentNode→leftChild )
      PostorderTraverse( currentNode→rightChild )
      Display currentNode
}
```

So the postorder traversal gives us: ACBEGFD

Notes

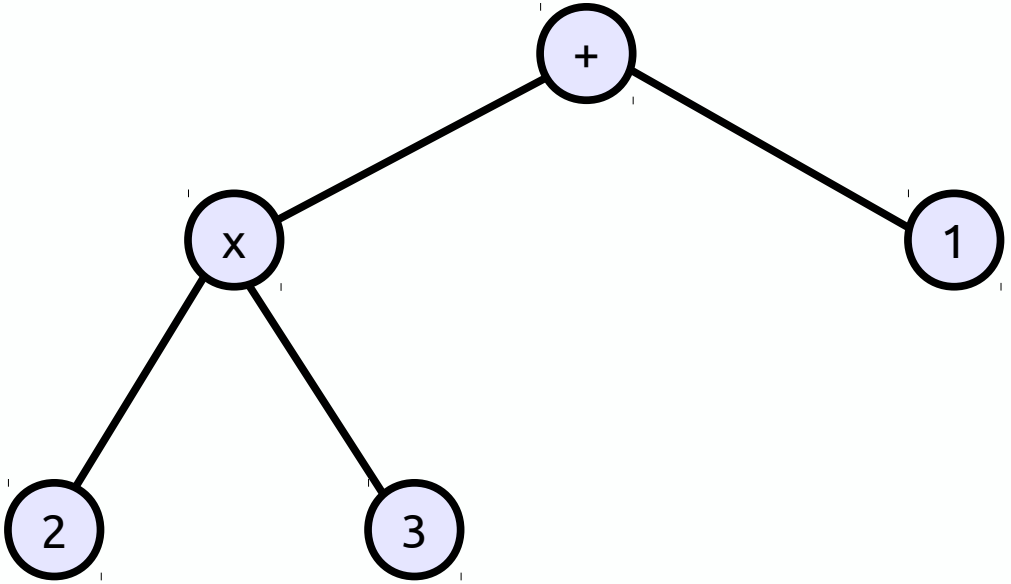
Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal

Practice: What is the output of an in-order traversal of this tree?



Notes

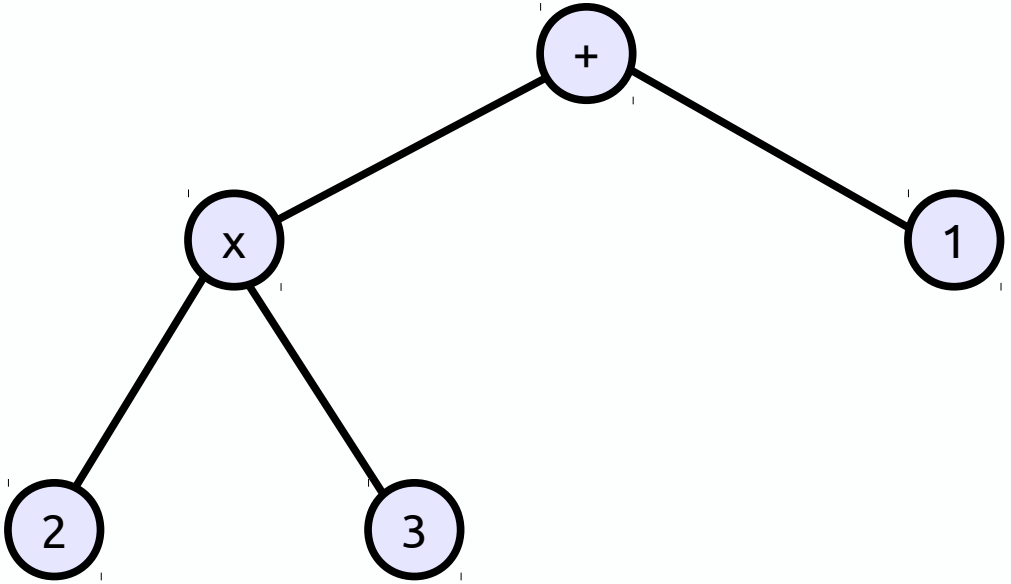
Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

2. Tree Traversal

Practice: What is the output of an in-order traversal of this tree?



2 x 3 + 1

Notes

Preorder Traversal:
- Display node
- Preorder(node→left)
- Preorder(node→right)

Inorder Traversal:
- Inorder(node→left)
- Display node
- Inorder(node→right)

Postorder Traversal:
- Postorder(node→left)
- Postorder(node→right)
- Display node

Binary Search Trees

3. Binary Search Trees

Binary Search Trees are a type of Binary Tree with the additional restriction:

- For node *n*, its left child must have a value that is *less than n's value*.
- For node *n*, its right child must have a value that is *greater than n's value*.

We can use Binary Search Trees to sort data by keeping smaller values on the left and larger values on the right.

Notes

3. Binary Search Trees

When we add a new item to a Binary Search Tree, we begin at the root (if there is one), and begin traversing left and right to find an available space.

Notes

3. Binary Search Trees

Example: Insert 5 3 4 2 1 into a Binary Search Tree

Notes

3. Binary Search Trees

Example: Insert 5 3 4 2 1 into a Binary Search Tree

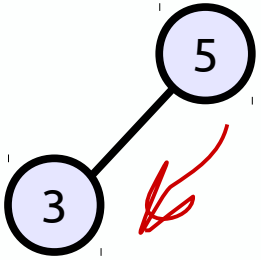
First insert: 5 (becomes root)

Notes

3. Binary Search Trees

Example: Insert 5 3 4 2 1 into a Binary Search Tree

Next: Insert 3. 3 is less than 5 so it goes left.



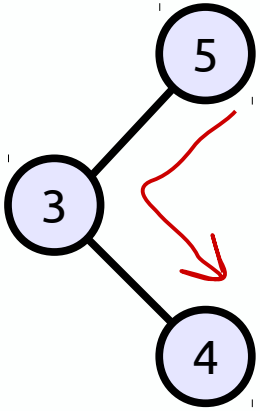
Notes

3. Binary Search Trees

Example: Insert 5 3 4 2 1 into a Binary Search Tree

Next: Insert 4.

4 is less than 5, 4 is greater than 3.

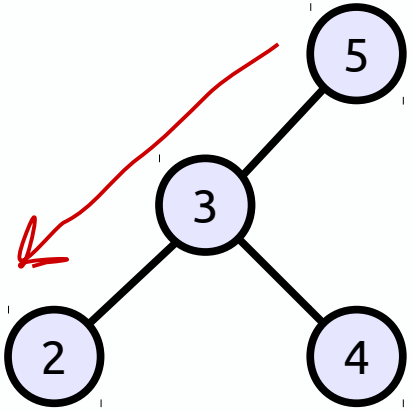


Notes

3. Binary Search Trees

Example: Insert 5 3 4 2 1 into a Binary Search Tree

Next: Insert 2. 2 < 5. 2 < 3.

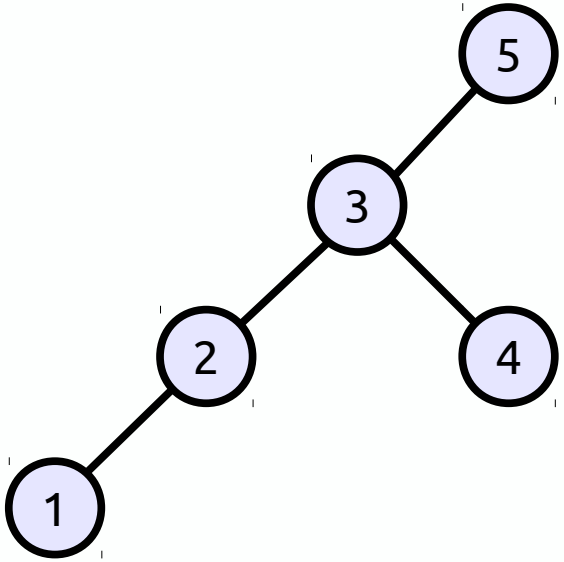


Notes

3. Binary Search Trees

Example: Insert 5 3 4 2 1 into a Binary Search Tree

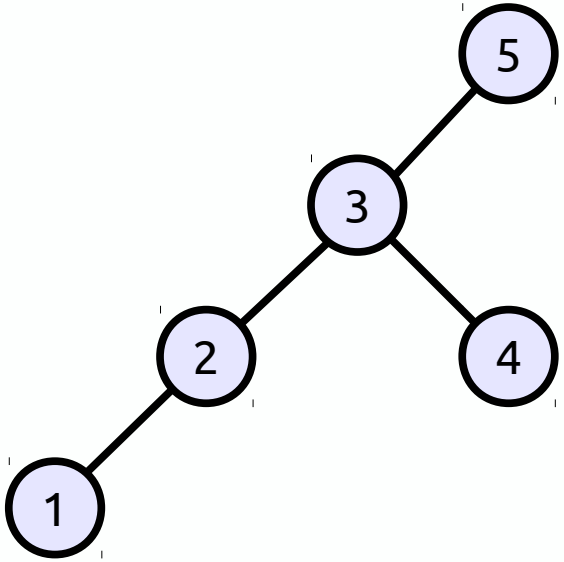
Next: Insert 1. 1 < 5. 1 < 3. 1 < 2.



Notes

3. Binary Search Trees

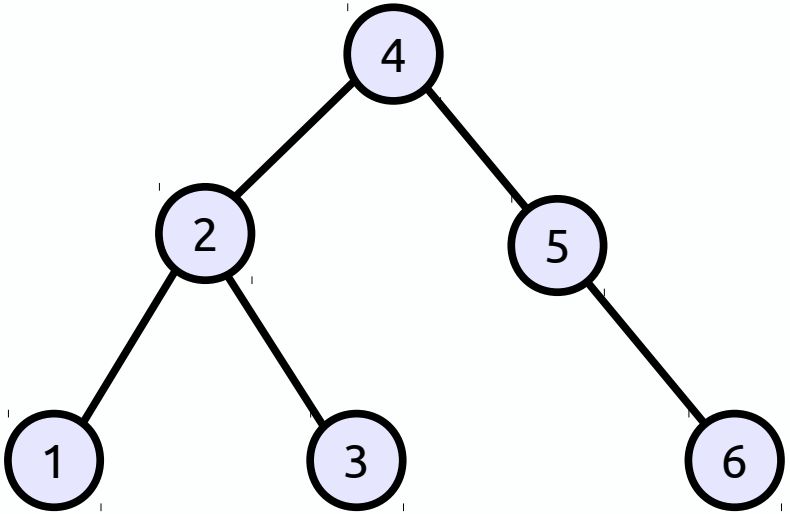
If we traversed this in-order, we would end up getting the numbers back in a sorted order:
1, 2, 3, 4, 5.



Notes

3. Binary Search Trees

Binary Search Trees are best for when data is going to be retrieved somewhat randomly; not in order.

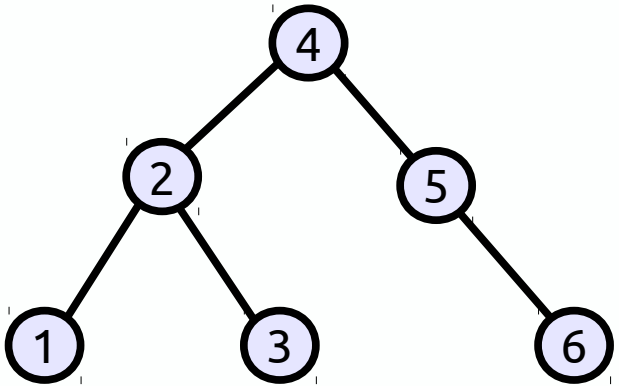


(Insert order: 4, 2, 5, 1, 6, 3)

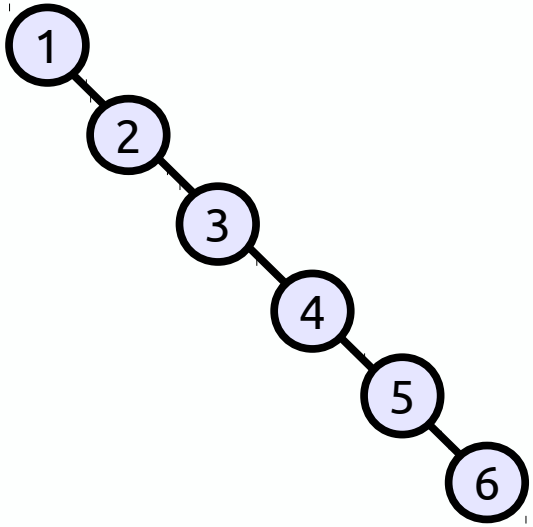
Notes

3. Binary Search Trees

If a sorted list of items are inserted into a Binary Search Tree, then the tree will end up heavily weighted to one side.



(Insert order: 4, 2, 5, 1, 6, 3)



(Insert order: 1, 2, 3, 4, 5, 6)

Notes

Conclusion

You'll see more of Binary Search Trees during a Data Structures course.