

Systematic way of visiting nodes in a tree

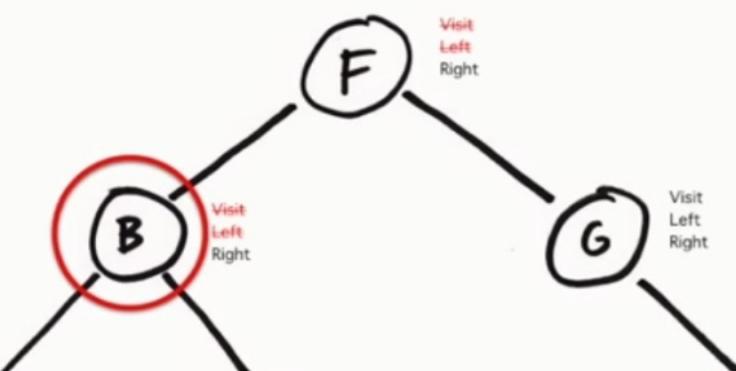
1. Depth-first → Stack
2. Breadth-first - Queue

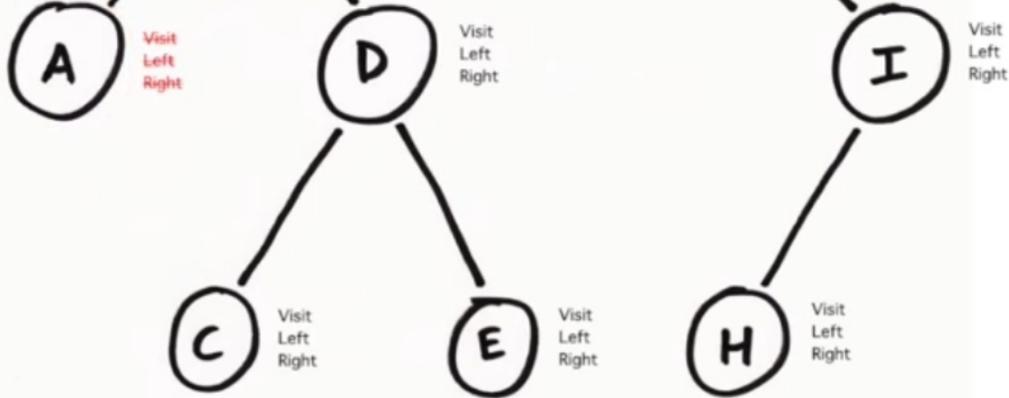
- Pre-order
- In-order
- Post-order
- Level-order

## PRE-ORDER

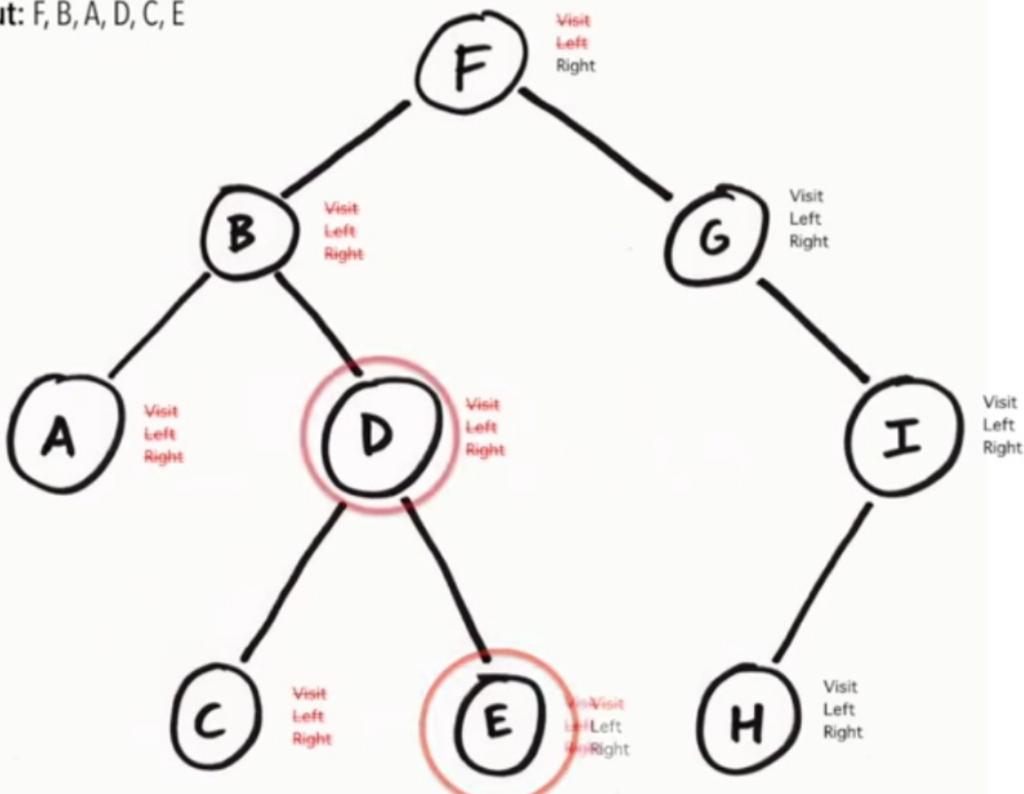
1. Visit node
2. Traverse left
3. Traverse right

Output: F, B, A

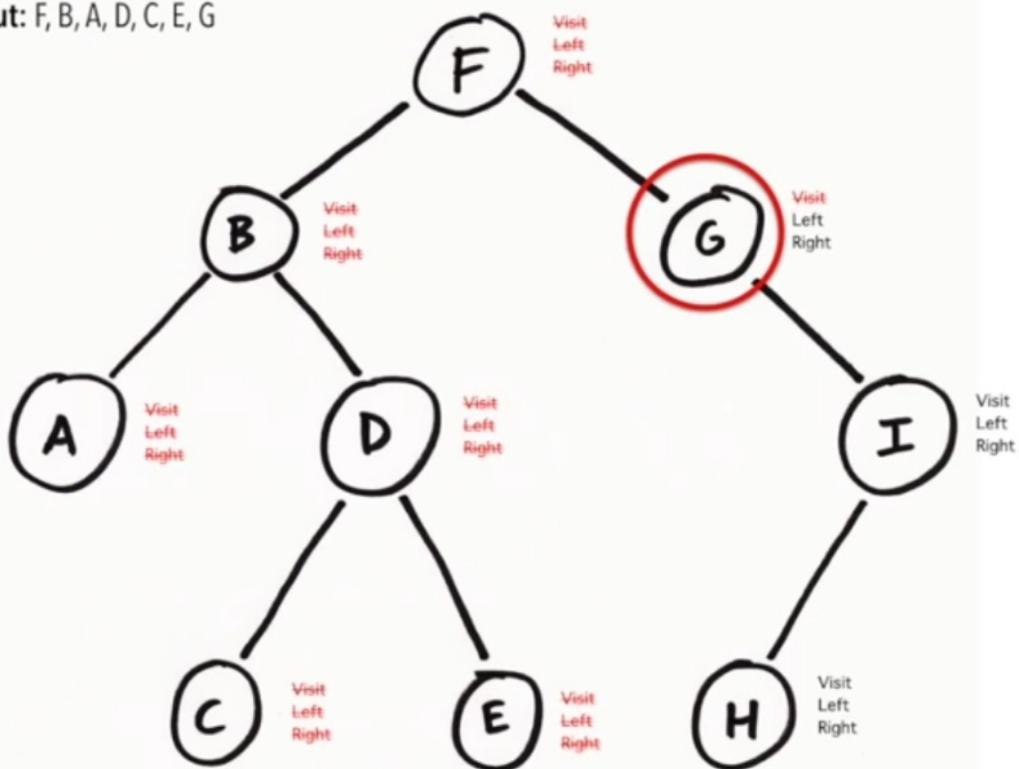




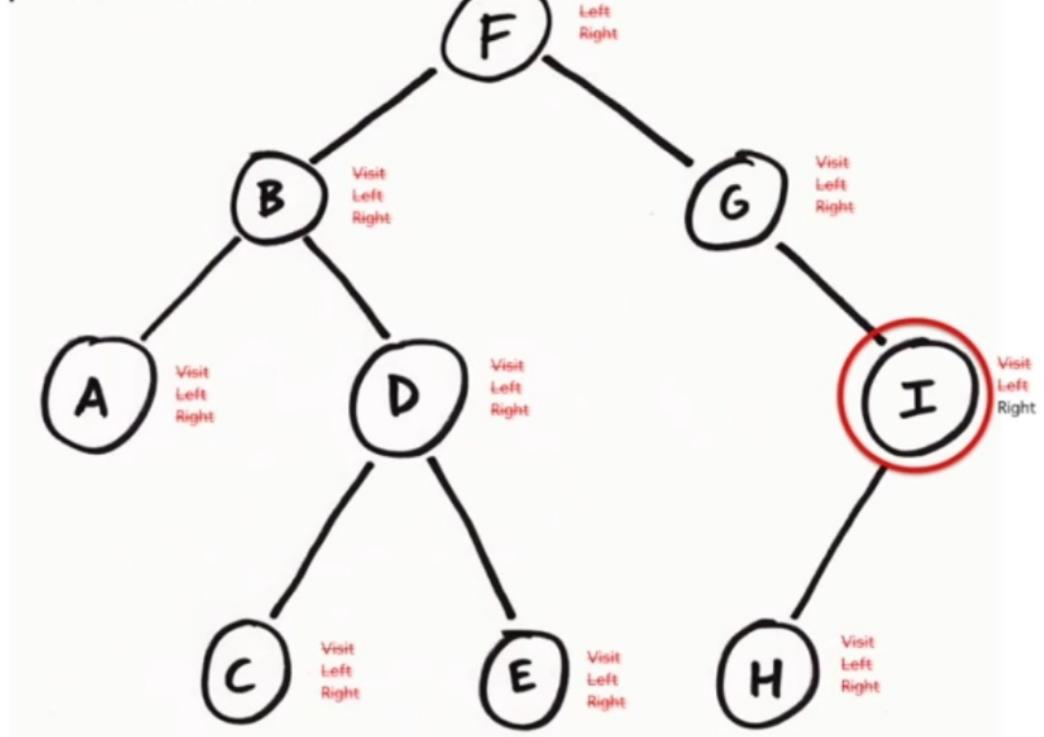
**Output:** F, B, A, D, C, E



**Output:** F, B, A, D, C, E, G



**Output:** F, B, A, D, C, E, G, I, H



## Pseudocode

```

preorder(node)
    if node == null then return
    visit(node)
    preorder(node.left)
    preorder(node.right)
    
```

## Time complexity

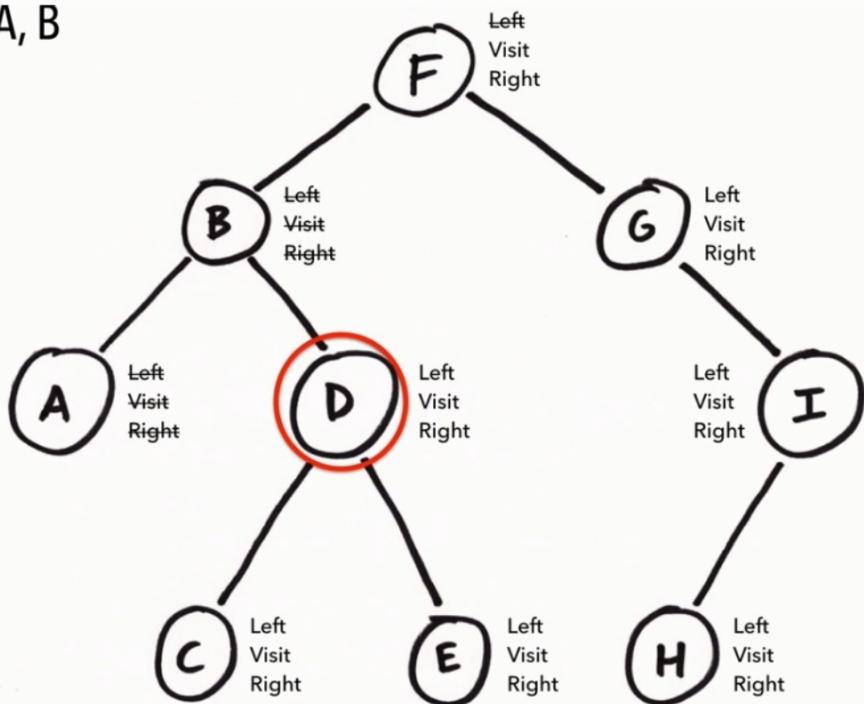
**O(n)**

where **n** is the number of nodes in the tree

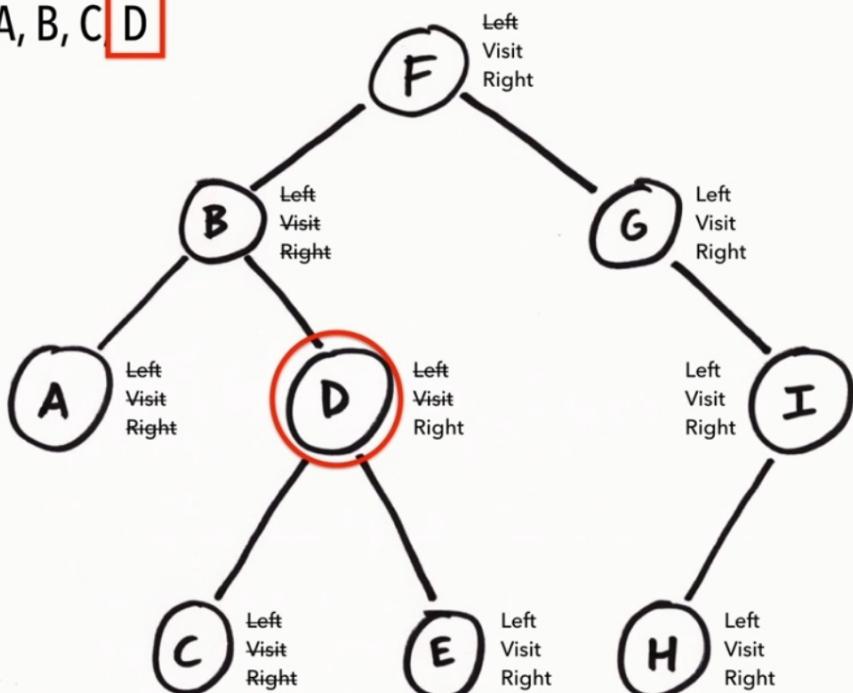
**IN - ORDER**

1. Traverse left
2. Visit node
3. Traverse right

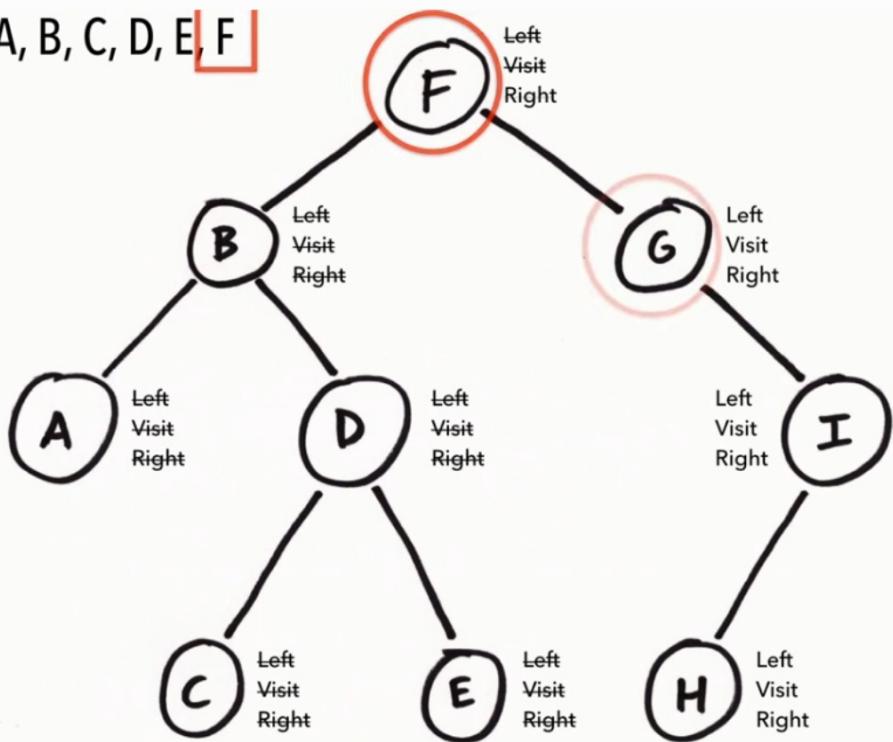
Output: A, B



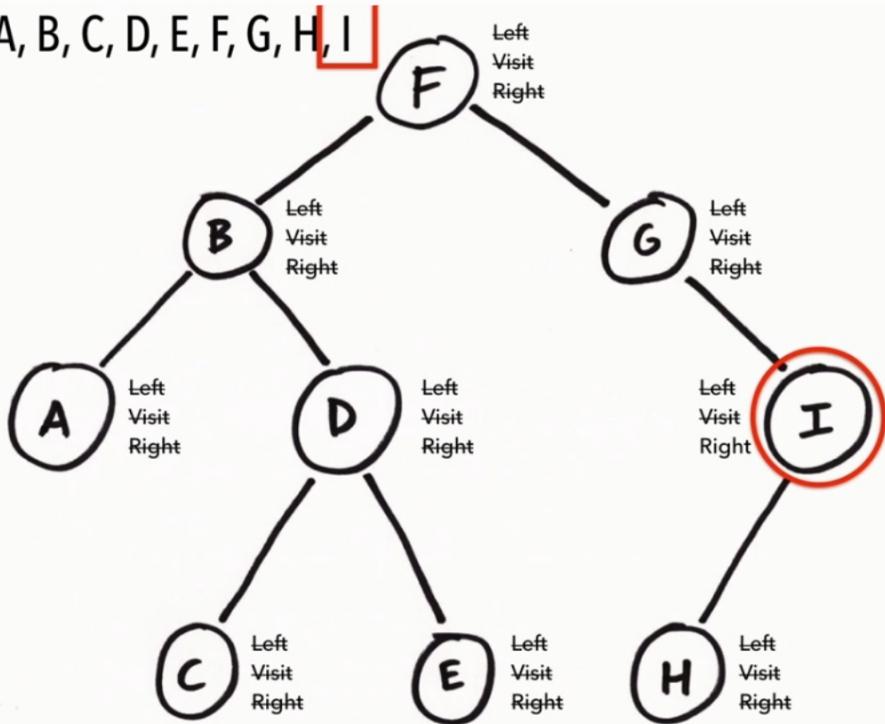
Output: A, B, C | D



Output: A, B, C, D, E, F



Output: A, B, C, D, E, F, G, H, I



## Pseudocode

**inorder**(node)

```
if node == null then return
inorder(node.left)
visit(node)
inorder(node.right)
```

# Time complexity

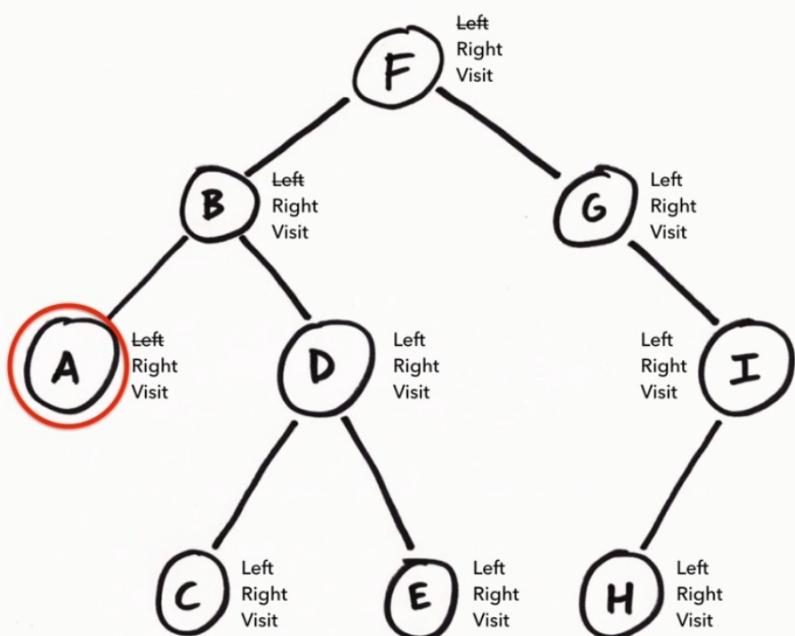
$O(n)$

where  $n$  is the number of nodes

POST-ORDER

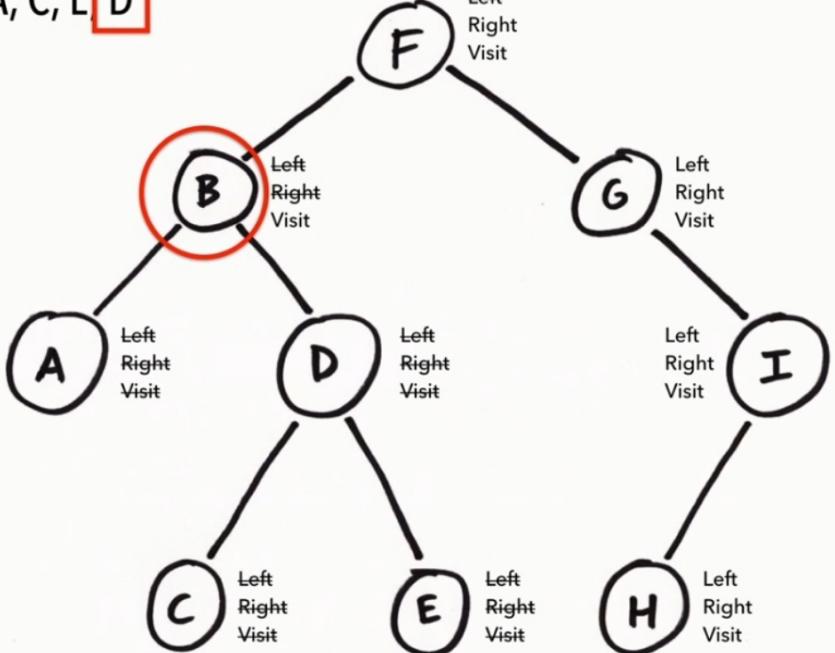
1. Traverse left
2. Traverse right
3. Visit node

Output:

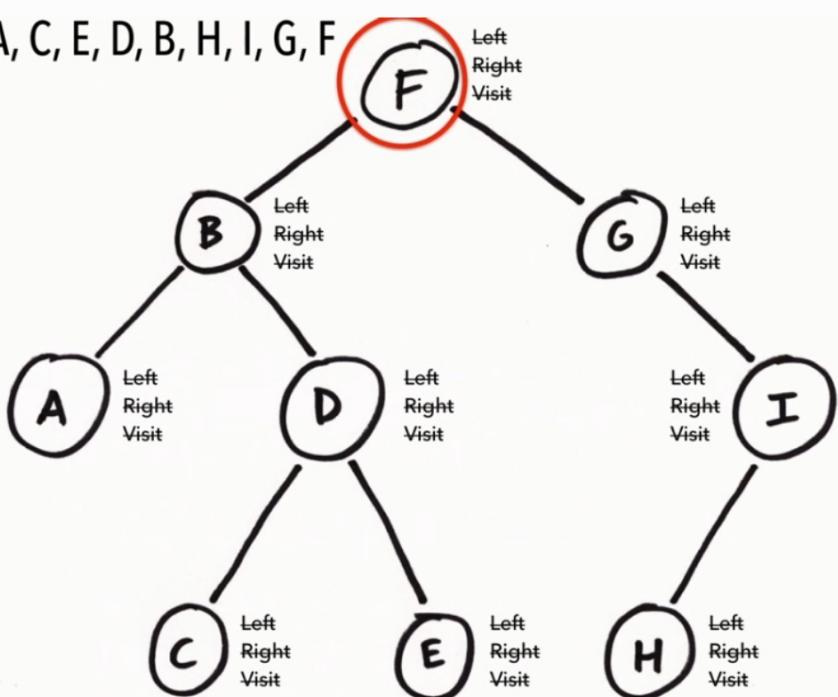


Output: A C E | D |

Output: A, C, E, D, B, H, I, G, F



Output: A, C, E, D, B, H, I, G, F



## Pseudocode

**postorder**(node)

```
if node == null then return
postorder(node.left)
postorder(node.right)
visit(node)
```

## Time complexity

$\Theta(n)$

$O(n)$

where **n** is the number of nodes

## LEVEL ORDER

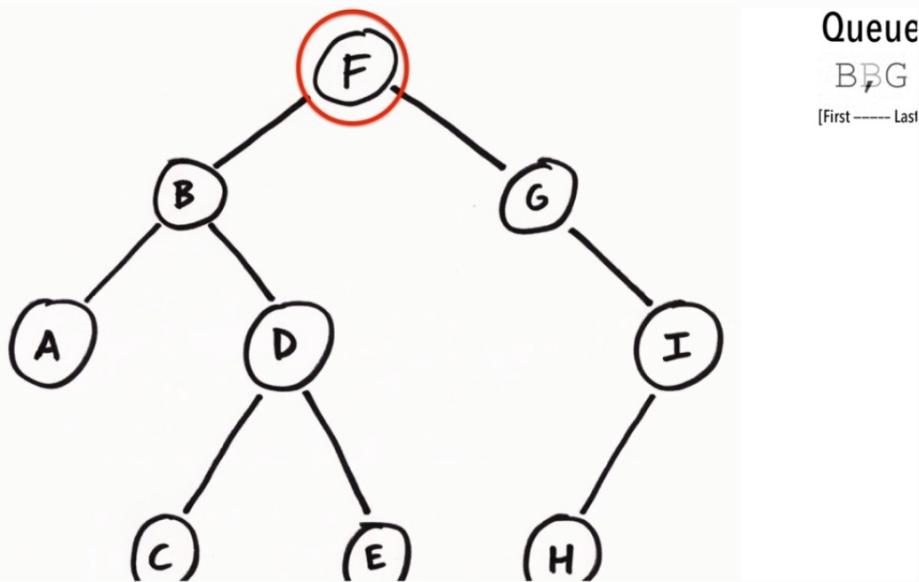
**levelorder** (root)

```

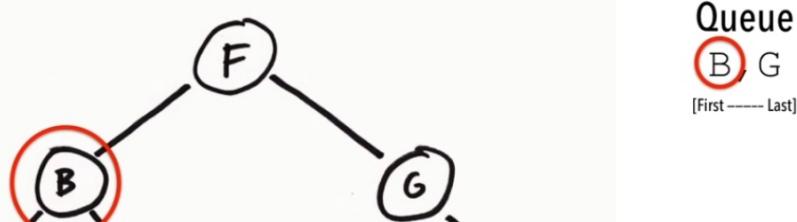
q ← empty queue
q.enqueue (root)
while (not q.isEmpty ( ))
    node ← q.dequeue ()
    visit (node)
    if (node.left ≠ null)
        q.enqueue (node.left)
    if (node.right ≠ null)
        q.enqueue (node.right)

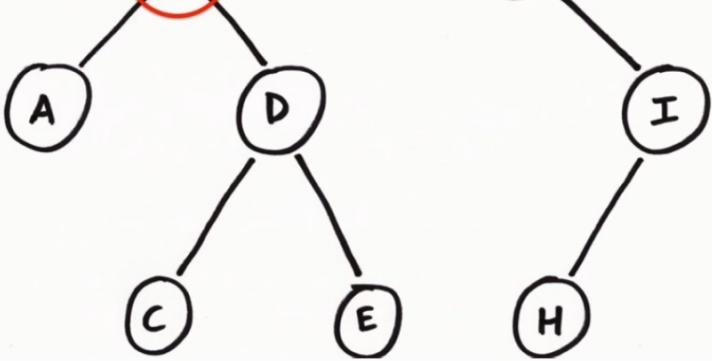
```

Output: F



Output: F



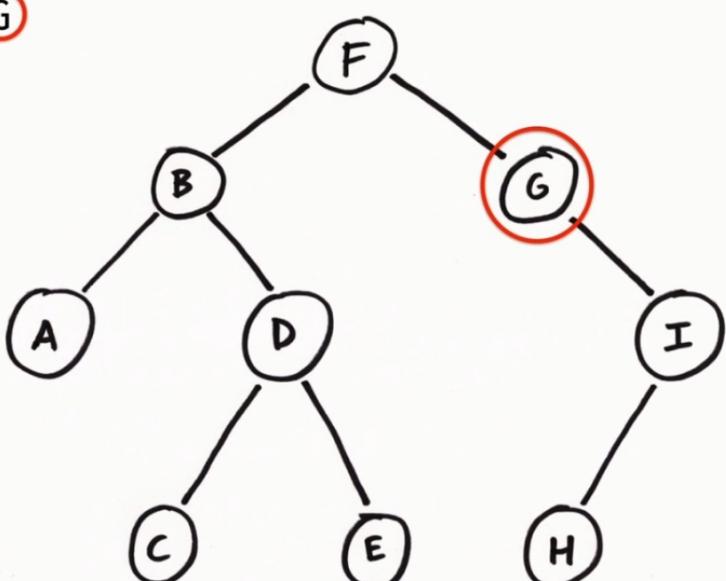


Output: F, B, G

Queue

A, D

[First ----- Last]

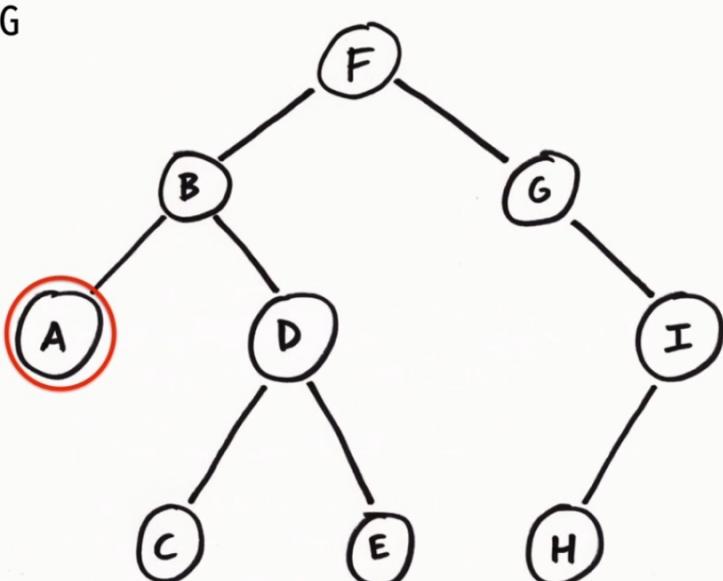


Output: F, B, G

Queue

A, D, I

[First ----- Last]

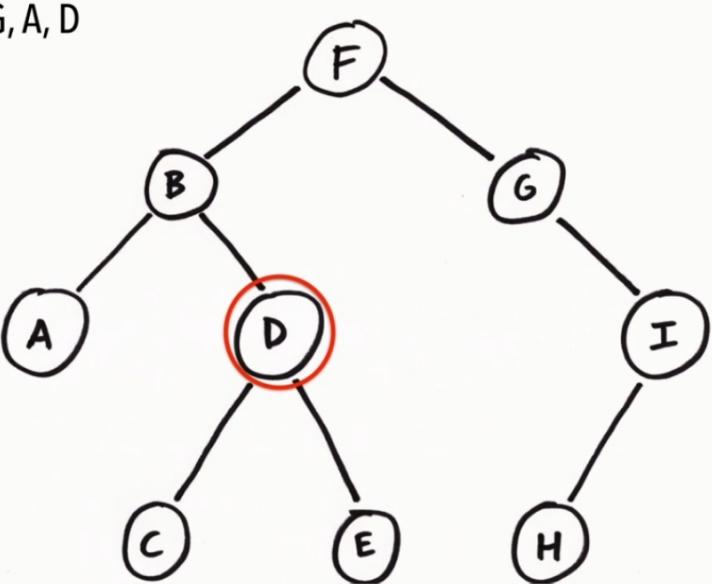


Output: F, B, G, A, D

Queue

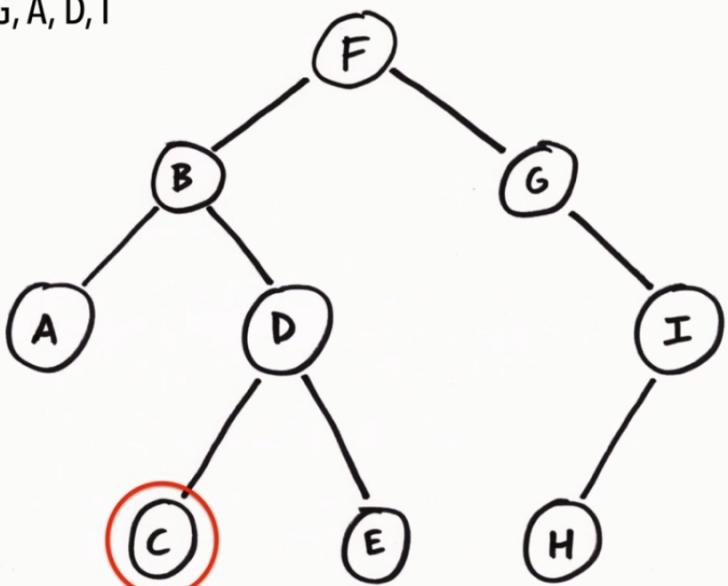
I, C, E

[First ----- Last]



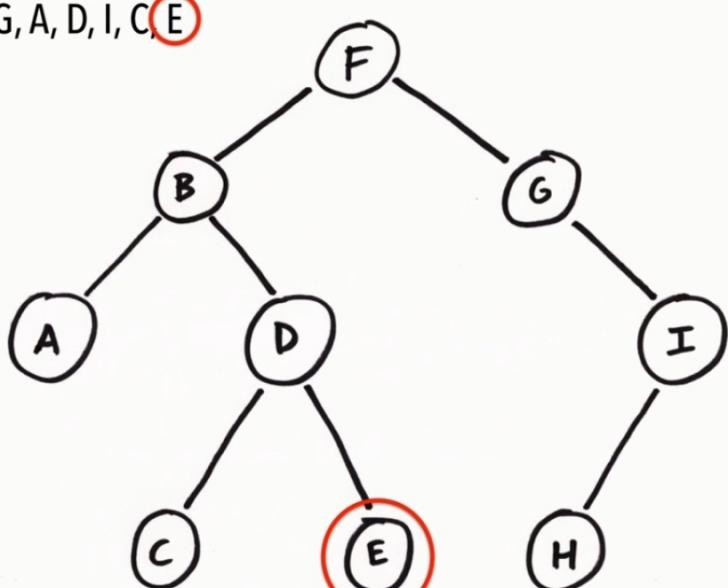
Output: F, B, G, A, D, I

Queue  
C, E, H  
[First ----- Last]



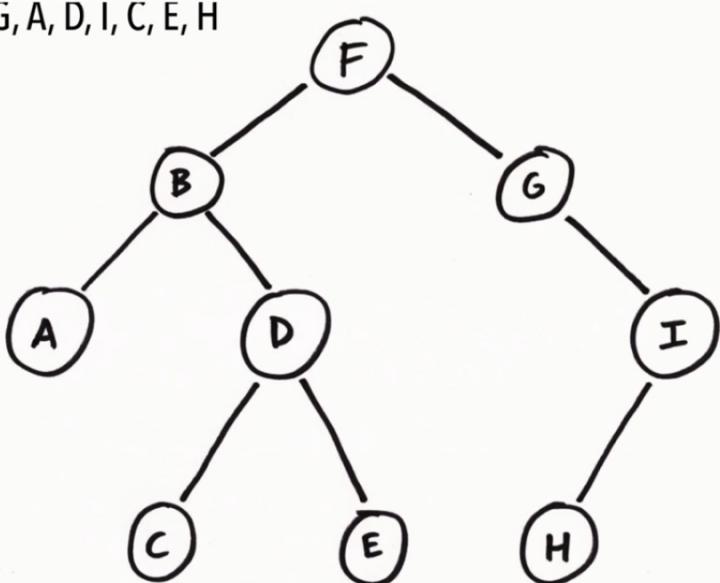
Output: F, B, G, A, D, I, C, E

Queue  
H  
[First ----- Last]



Output: F, B, G, A, D, I, C, E, H

Queue



Time complexity

$O(n)$

where **n** is the number of nodes

