**Depth-first search**

[**https://en.wikipedia.org/wiki/Tree\_traversal**](https://en.wikipedia.org/wiki/Tree_traversal)

**Pre-order**

|  |  |
| --- | --- |
| **preorder**(node)  **if** (node = **null**)  **return**  visit(node)  preorder(node.left)  preorder(node.right) | **iterativePreorder**(node)  **if** (node = **null**)  **return**  s ← **empty stack**  s.push(node)  **while** (**not** s.isEmpty())  node ← s.pop()  visit(node)  //right child is pushed first so that left is processed first  **if** (node.right ≠ **null**)  s.push(node.right)  **if** (node.left ≠ **null**)  s.push(node.left) |

**In-order**

|  |  |
| --- | --- |
| **inorder**(node)  **if** (node = **null**)  **return**  inorder(node.left)  visit(node)  inorder(node.right) | **iterativeInorder**(node)  s ← **empty stack**  **while** (**not** s.isEmpty() **or** node ≠ **null**)  **if** (node ≠ **null**)  s.push(node)  node ← node.left  **else**  node ← s.pop()  visit(node)  node ← node.right |

**Post-order**

|  |  |
| --- | --- |
| **postorder**(node)  **if** (node = **null**)  **return**  postorder(node.left)  postorder(node.right)  visit(node) | **iterativePostorder**(node)  s ← **empty stack**  lastNodeVisited ← **null**  **while** (**not** s.isEmpty() **or** node ≠ **null**)  **if** (node ≠ **null**)  s.push(node)  node ← node.left  **else**  peekNode ← s.peek()  // if right child exists and traversing node  // from left child, then move right  **if** (peekNode.right ≠ **null** **and** lastNodeVisited ≠ peekNode.right)  node ← peekNode.right  **else**  visit(peekNode)  lastNodeVisited ← s.pop() |

**Breadth-first search**

**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)