* Stack-based BST Traversal:
  + Create an empty stack S
  + S.push(root)
  + Each tree node will contain: left, right, data, and flags (leftVisited and rightVisited)
  + While (stack not empty)
    - If S.Top.leftVisited is false
      * Push S.Top.Left onto S
    - If S.Top.leftVisited is true and S.Top.rightVisited is false
      * Push S.Top.Right onto S
    - Check whether S.Top.rightVisited is true
    - You pop the stack when both leftVisited and rightVisited are true for S.Top
      * Get the current path by going down the stack
* Depth-First Traversal/Search (DFS)
  + Base case: if root is null, do nothing
  + Pre-order traversal
    - S.O.P(root.data)
    - Pre-order(root.left)
    - Pre-order(root.right)
  + In-order traversal
    - Pre-order(root.left)
    - S.O.P(root.data)
    - Pre-order(root.right)
  + Post-order traversal
    - Pre-order(root.left)
    - Pre-order(root.right)
    - S.O.P(root.data)
  + Called “Depth-first traversal” because the algorithms always try to go deeper into the tree before exploring siblings.
* Breadth-first Traversal (level-by-level) (top-to-bottom)
  + On each level, go to each node left to right. Then go down to the next level.
  + Use a queue Q.
  + Q.enqueue(root)
  + While (Q is not empty)
    - Node cur = Q.dequeue()
    - S.O.P(cur.data)
    - If (cur.left != null)
      * Q.enqueue(cur.left)
    - If (cur.right != null)
      * Q.enqueue(cur.right)