Binary Search Tree Iterator

This problem is about iterating a BTS in in-order traversal efficiently, using $O(h)$ space and $O(1)$ amortized time per operation.

Key Insight:

- In-order traversal of BST gives elements in sorted ascending order.
- Instead of storing the entire traversal ($O(n)$ space), we use a stack to simulate recursion:
  - Push all left children from the current node onto the stack.
  - When next() is called:
    - Pop from stack (smallest element not yet visited).
    - Process its right subtree (push its left children).
- hasNext() simply checks if the stack is non-empty.

Algorithm:

Initialization:

- Constructor: push all leftmost nodes starting from root.

next():

1. Pop the top node (smallest).
2. Push all leftmost nodes of its right child (if exists).
3. Return the popped value.

hasNext():

- Return true if stack is not empty.

Complexity:

- Time (Amortized): $O(1)$ per next() and hasNext().
  (Each node is pushed/popped at most once).
- Space: $O(h)$, where h is tree height (stack depth).