Problem 426: Convert Binary Search Tree to Sorted Doubly Linked List

Problem Information

Difficulty: Medium
Acceptance Rate: 65.62%
Paid Only: Yes
Tags: Linked List, Stack, Tree, Depth-First Search, Binary Search Tree, Binary Tree, Doubly-Linked List

Problem Description

Convert a **Binary Search Tree** to a sorted **Circular Doubly-Linked List** in place.

You can think of the left and right pointers as synonymous to the predecessor and successor pointers in a doubly-linked list. For a circular doubly linked list, the predecessor of the first element is the last element, and the successor of the last element is the first element.

We want to do the transformation **in place**. After the transformation, the left pointer of the tree node should point to its predecessor, and the right pointer should point to its successor. You should return the pointer to the smallest element of the linked list.

Example 1:

![](https://assets.leetcode.com/uploads/2018/10/12/bstdlloriginalbst.png)

Input: root = [4,2,5,1,3]
![](https://assets.leetcode.com/uploads/2018/10/12/bstdllreturndll.png) **Output:** [1,2,3,4,5]
Explanation: The figure below shows the transformed BST. The solid line indicates the successor relationship, while the dashed line means the predecessor relationship.
![](https://assets.leetcode.com/uploads/2018/10/12/bstdllreturnbst.png)

Example 2:

Input: root = [2,1,3] **Output:** [1,2,3]

Constraints:

* The number of nodes in the tree is in the range `[0, 2000]`. * `-1000 <= Node.val <= 1000` * All the values of the tree are **unique**.

Code Snippets

C++:

```
/*
// Definition for a Node.
class Node {
public:
int val;
Node* left;
Node* right;

Node() {}

Node(int _val) {
val = _val;
left = NULL;
right = NULL;
}

Node(int _val, Node* _left, Node* _right) {
val = _val;
left = _left;
right = _right;
}
};
*/

class Solution {
public:
Node* treeToDoublyList(Node* root) {

}
};
```

Java:

```
/*
// Definition for a Node.
class Node {
public int val;
public Node left;
public Node right;

public Node() {}

public Node(int _val) {
val = _val;
}

public Node(int _val,Node _left,Node _right) {
val = _val;
left = _left;
right = _right;
}
};
*/

class Solution {
public Node treeToDoublyList(Node root) {

}
}
```

Python3:

```
"""
# Definition for a Node.
class Node:
def __init__(self, val, left=None, right=None):
self.val = val
self.left = left
self.right = right
"""

class Solution:
def treeToDoublyList(self, root: 'Optional[Node]') -> 'Optional[Node]':
```