# 426. Convert Binary Search Tree to Sorted Doubly Linked List

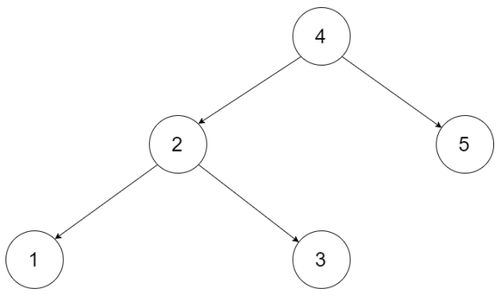
Medium 1473 127 Add to List Share

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

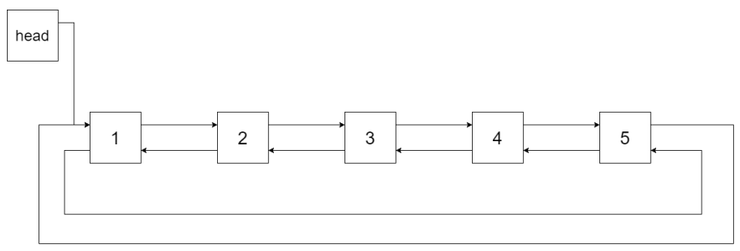
You can think of the left and right pointers as synonymous to the predecessor and successor pointers in a doubly-linked list. For a circular doubly linked list, the predecessor of the first element is the last element, and the successor of the last element is the first element.

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:**

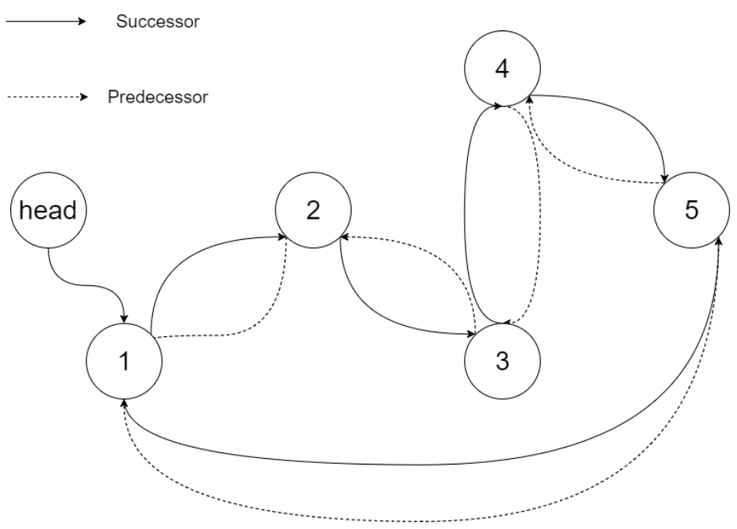



**Input:** root = [4,2,5,1,3]




**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.




**Example 2:**

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

**Example 3:**

**Input:** root = []
**Output:** []
**Explanation:** Input is an empty tree. Output is also an empty Linked List.

**Example 4:**

**Input:** root = [1]
**Output:** [1]

**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**.

Accepted 130,539 Submissions 209,617

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Facebook | 31 Microsoft | 6 Amazon | 4 Google | 2 Expedia | 2

Related Topics

Similar Questions

Python3 Autocomplete

```python
"""
# 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: 'Node') -> 'Node':
        if not root:
            return None

        head = None
        tail = None

        def treeToDoublyListHelper(node):
            nonlocal head, tail
            if not node:
                return
            treeToDoublyListHelper(node.left)
            if tail:
                node.left = tail
                tail.right = node
            else:
                head = node
            tail = node
            treeToDoublyListHelper(node.right)

        treeToDoublyListHelper(root)
        head.left = tail
        tail.right = head
        return head
```

Testcase Run Code Result Debugger

Accepted Runtime: 48 ms

Your input
```
[4,2,5,1,3]
[2,1,3]
```

Output
```
[1,2,3,4,5]
[1,2,3]
```

Diff

Expected
```
[1,2,3,4,5]
[1,2,3]
```