```swift
import Foundation
/// 剑指 Offer 36. 二叉搜索树与双向链表
/// 输入一棵二叉搜索树，将该二叉搜索树转换成一个排序的循环双向链表。
/// 要求不能创建任何新的节点，只能调整树中节点指针的指向。
class Solution {
    /// 私有变量，用于记录上一个遍历到的节点
    private var pre: Node?
    /// 私有变量，用于记录转换后的双向链表的头节点
    private var head: Node?
    /// 主函数，将二叉搜索树转换为双向链表
    func treeToDoublyList(_ root: Node?) -> Node? {
        self.pre = nil
        self.head = nil
        /// 调用 dfs 函数进行遍历和修改节点引用
        dfs(root)
        /// 最后，将双向链表的头尾节点进行连接，并返回头节点
        self.head?.left = self.pre
        self.pre?.right = self.head
        return self.head
    }
    /// 递归函数，深度优先遍历二叉树
    private func dfs(_ cur: Node?) {
        /// 若当前节点为空，直接返回
        guard let cur = cur else { return }
        /// 递归遍历左子树
        dfs(cur.left)
        /// 修改节点引用，若 pre 为空，则当前节点为双向链表的头节点
        if let pre = self.pre {
            pre.right = cur
            cur.left = pre
        } else { self.head = cur }

        /// 保存当前节点为 pre，供下一次遍历使用
        self.pre = cur
        /// 递归遍历右子树
        dfs(cur.right)
    }
}
// 定义二叉树节点类
class Node {
    var val: Int // 节点值
    var left: Node? // 左子节点
    var right: Node? // 右子节点

    init(_ val: Int) {
        self.val = val
    }
}
```