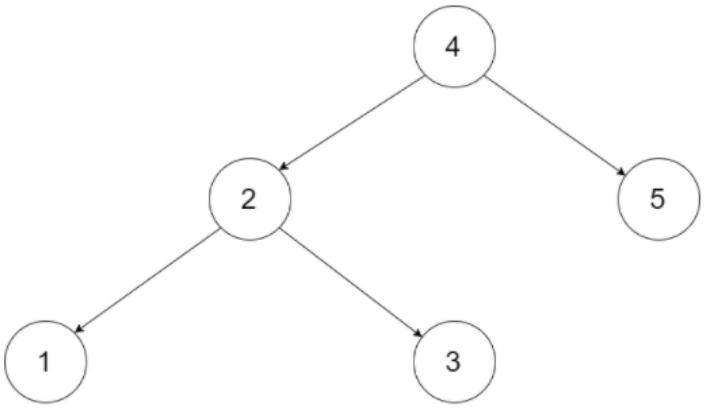
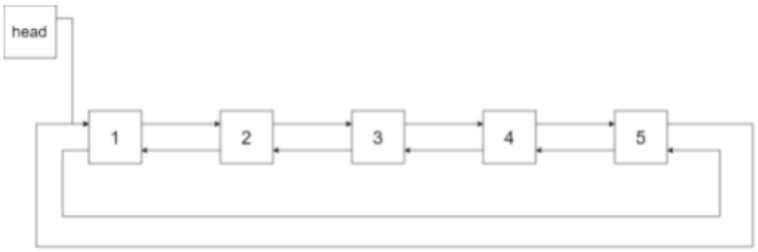
二叉搜索树与双向链表

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。为了让您更好地理解问题，以下面的二叉搜索树为例：



我们希望将这个二叉搜索树转化为双向循环链表。链表中的每个节点都有一个前驱和后继指针。对于双向循环链表，第一个节点的前驱是最后一个节点，最后一个节点的后继是第一个节点。

下图展示了上面的二叉搜索树转化成的链表。“head” 表示指向链表中有最小元素的节点。



特别地，我们希望可以就地完成转换操作。当转化完成以后，树中节点的左指针需要指向前驱，树中节点的右指针需要指向后继。还需要返回链表中的第一个节点的指针。

```
/*
// 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* head;
    Node* prev;
    void dfs(Node* cur)
    {
        if(cur==nullptr)
            return;
        //中序遍历 左根右
        dfs(cur->left);
        if(prev==nullptr)
        {
            head=cur;
            prev=head;
        }
        else
        {
            prev->right=cur;
            cur->left=prev;
            prev=cur;//更新 prev 后移
        }
        dfs(cur->right);
    }
    Node* treeToDoublyList(Node* root) {
        if(root==nullptr)
            return nullptr;
        dfs(root);
        head->left=prev;
        prev->right=head;
        return head;
    }
```

```
};
```