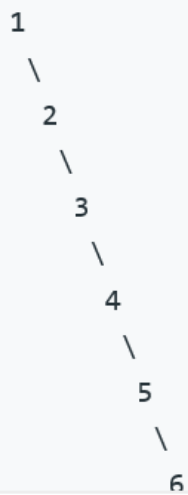


Given a binary tree, flatten it to a linked list in-place.

For example, given the following tree:



The flattened tree should look like:



```

1 package com.leetcode;
2
3 class TreeNode {
4     int val;
5     TreeNode left;
6     TreeNode right;
7     TreeNode(int x) { val = x; }
8 }
9
10 public class L114 {
11     /*
12     * 解题思路:
13     * 对于根节点来说, 它的下一个节点在左子节点存在的情况下, 就是它的左子节点。所以在变换中会将它的右指针
14     * 指向它的左子节点。在原来的过程中, 将左子节点遍历完之后才会去遍历它的右子节点。所以在左子树中最后遍历
15     * 的那个节点是它左子节点最右下角的那个节点。
16     * 这样, 可以概括出这样的一个过程。每次根据一个节点, 找它左子节点的最右下角的元素。如果有, 将这个元素的
17     * right指向根节点的右子节点。然后根节点的right指向它的左子节点。再指向它的下一个位置, 也就是它的right
18     * 节点。
19     */
20     public void flatten(TreeNode root) {
21         while(root != null) {
22             if(root.left != null) {
23                 TreeNode ptr = root.left;
24                 while(ptr.right != null)
25                     ptr = ptr.right;
26                 ptr.right = root.right;
27                 root.right = root.left;
28                 root.left = null;
29             }
30             root = root.right;
31         }
32     }
33 }
34

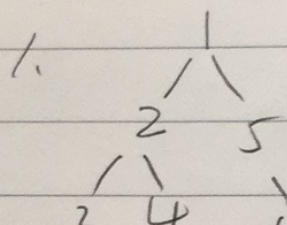
```

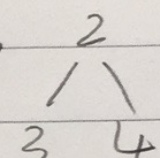
11. 二叉树展开为链表

解题思路：

对于根节点来说，它的下一个节点在左子节点存在的情况下，就是它的左子节点。所以在变换中会将它的右指针指向它的左子节点。在原来的过程中，将左子节点遍历完之后才会去遍历它的右子节点。所以在左子树中最后遍历的那个节点是它左子节点最右下角的那个节点。这样，可以概括出这样的一个过程。每次根据一个节点，找它左子节点的最右下角的元素。如果有，将这个元素的right指向根节点的右子节点。然后根节点的right指向它的左子节点。再指向它的下一个位置，也就是它的right节点。

我们来看第一步：

1.  2.  $ptr = root.left$

$ptr \rightarrow$  

3. while (ptr.right != null)  
ptr = ptr.right.

4. ptr.right = null

5. root.right = root.left.  
root.left = null

