Flatten Binary Tree to Linked List
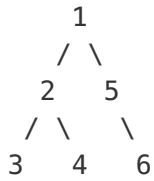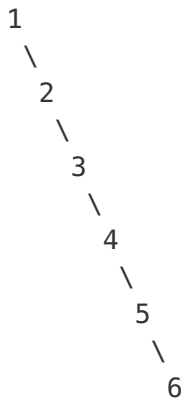
Given a binary tree, flatten it to a linked list in-place.

For example,
Given

```
     1
    / \
   2   5
  / \   \
 3   4   6
```

The flattened tree should look like:

```
 1
  \
   2
    \
     3
      \
       4
        \
         5
          \
           6
```

click to show hints.

Hints:
If you notice carefully in the flattened tree, each node's right child points to the next node of a pre-order traversal.

Solution 1

```java
private TreeNode prev = null;

public void flatten(TreeNode root) {
    if (root == null)
        return;
    flatten(root.right);
    flatten(root.left);
    root.right = prev;
    root.left = null;
    prev = root;
}
```

written by tusizi original link here

Solution 2

```cpp
class Solution {
public:
    void flatten(TreeNode *root) {
        TreeNode*now = root;
        while (now)
        {
            if(now->left)
            {
                //Find current node's prenode that links to current node's right subtree
                TreeNode* pre = now->left;
                while(pre->right)
                {
                    pre = pre->right;
                }
                pre->right = now->right;
                //Use current node's left subtree to replace its right subtree(original right
                //subtree is already linked by current node's prenode
                now->right = now->left;
                now->left = NULL;
            }
            now = now->right;
        }
    }
};
```

written by zjulyx original link here

Solution 3

```java
public void flatten(TreeNode root) {
    if (root == null) return;

    TreeNode left = root.left;
    TreeNode right = root.right;

    root.left = null;

    flatten(left);
    flatten(right);

    root.right = left;
    TreeNode cur = root;
    while (cur.right != null) cur = cur.right;
    cur.right = right;
}
```

This solution is based on recursion. We simply flatten left and right subtree and paste each sublist to the right child of the root. (don't forget to set left child to null)

written by hanzhou87 original link here