

```

1  /*
2  给定一个二叉树，原地将它展开为链表。
3
4  例如，给定二叉树
5
6      1
7     /\
8    2  5
9   /\  \
10  3  4  6
11
12  将其展开为：
13
14  1
15  \
16  2
17  \
18  3
19  \
20  4
21  \
22  5
23  \
24  6
25
26  来源：力扣（LeetCode）
27  链接：https://leetcode-cn.com/problems/flatten-binary-tree-to-linked-list
28  著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
29  */

```

分析:

- 分别递归的展开左右子树,在左子树不为空的情况下,整体将左子树插入到右子树.
- 遍历顺序可以是左右根也可以是右左根.

方法一:C++_递归展开

```

1  /**
2   * Definition for a binary tree node.
3   * struct TreeNode {
4   *     int val;
5   *     TreeNode *left;
6   *     TreeNode *right;
7   *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
8   * };
9   */
10
11
12  /*
13  分别递归展开左右子树,然后将左子树全部展开到右子树.
14  */
15
16  /**

```

```

17  * Definition for a binary tree node.
18  * struct TreeNode {
19  *     int val;
20  *     TreeNode *left;
21  *     TreeNode *right;
22  *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
23  * };
24  */
25  class Solution
26  {
27  private:
28      void __dfsPostOrder(TreeNode* node)
29      {
30          /*当前节点为空*/
31          if(node == NULL)
32          {
33              return;
34          }
35          /* 展开左子树*/
36          __dfsPostOrder(node->left);
37          /* 展开右子树*/
38          __dfsPostOrder(node->right);
39          /* 将左子树插入到右子树*/
40          if(node->left)
41          {
42              TreeNode* node_right_temp = node->right ; /*保存原来的右子树根
节点*/
43              TreeNode* node_left_tail  = node->left  ;
44
45              /*寻找尾巴*/
46              while(node_left_tail&&node_left_tail->right)
47              {
48                  node_left_tail = node_left_tail->right;
49              }
50              /*将左子树整体插入右子树*/
51              node->right          = node->left;
52              node->left            = NULL;
53              node_left_tail->right = node_right_temp;
54              node_left_tail->left  = NULL;
55          }
56      }
57  public:
58      void flatten(TreeNode* root)
59      {
60          __dfsPostOrder(root);
61          return ;
62      }
63  };
64  /*
65  执行结果:
66  通过
67  显示详情
68  执行用时 :4 ms, 在所有 C++ 提交中击败了98.35% 的用户
69  内存消耗 :9.6 MB, 在所有 C++ 提交中击败了91.48%的用户
70  */

```

