

```

1  /**
2   给定一个树，按中序遍历重新排列树，使树中最左边的结点现在是树的根，并且每个结点没有左子结
   点，只有一个右子结点。
3   示例：
4   输入: [5,3,6,2,4,null,8,1,null,null,null,7,9]
5
6       5
7      /\
8     3  6
9    /\  \
10   2  4  8
11  /\    /\
12 1  7  9
13
14 输出: [1,null,2,null,3,null,4,null,5,null,6,null,7,null,8,null,9]
15
16  1
17  \
18  2
19  \
20  3
21  \
22  4
23  \
24  5
25  \
26  6
27  \
28  7
29  \
30  8
31  \
32  9
33
34 提示:
35  给定树中的结点数介于 1 和 100 之间。
36  每个结点都有一个从 0 到 1000 范围内的唯一整数值。
37  来源: 力扣 (LeetCode)
   链接: https://leetcode-cn.com/problems/increasing-order-search-tree
   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。
   */

```

分析:

- 方法一:利用队列保存中序遍历(递归)的结果,然后根据出队顺序建立链表.
- 方法二:在中序遍历(递归)的过程中,直接根据遍历顺序建立链表.
- 方法三:在中序遍历(迭代)的过程中,直接根据遍历顺序建立链表.(todo)

方法一:C++_DFS+队列

```

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     class solution
11     {
12     private:
13         queue<TreeNode*> qtn;
14         /*中序遍历*/
15         void __midOrder(TreeNode* node)
16         {
17             if(node==NULL)
18             {
19                 return;
20             }
21             /* 左 根 右*/
22             __midOrder(node->left);
23             qtn.push(node);
24             __midOrder(node->right);
25         }
26
27     public:
28         TreeNode* increasingBST(TreeNode* root)
29         {
30             TreeNode* ret_val = NULL;
31             TreeNode* cur      = NULL;
32             queue<TreeNode*> empty;
33             swap(empty, qtn);
34             if(root)
35             {
36                 /*获取中序遍历的结果*/
37                 __midOrder(root);
38                 /*依次取出队列的值组成"链表"*/
39                 ret_val = qtn.front();
40                 cur      = ret_val;
41                 while(1)
42                 {
43                     qtn.pop();
44                     if(qtn.empty())
45                     {
46                         cur->left  = NULL;
47                         cur->right = NULL;
48                         break;
49                     }
50                     cur->left  = NULL;
51                     cur->right = qtn.front();
52                     cur      = cur->right;
53                 }
54             }
55             return ret_val;
56         }
57     };
58     /*
59     执行结果:
60     通过
61     显示详情
62     执行用时 :40 ms, 在所有 cpp 提交中击败了96.31%的用户

```

```
63 | 内存消耗 :18.7 MB, 在所有 cpp 提交中击败了91.46%的用户
64 | */
```

方法二:C++_DFS

```
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 class Solution
11 {
12     private:
13         TreeNode*      ret_val = NULL;
14         TreeNode*      cur      = NULL;
15         /*中序遍历*/
16         void __midOrder(TreeNode* node)
17         {
18             if(node==NULL)
19             {
20                 return;
21             }
22             /* 左 根 右*/
23             __midOrder(node->left);
24             if(ret_val)
25             {
26                 cur->left = NULL;
27                 cur->right = node;
28                 cur        = cur->right;
29             }
30             else
31             {
32                 ret_val = node      ;
33                 cur      = ret_val  ;
34             }
35             __midOrder(node->right);
36         }
37
38     public:
39         TreeNode* increasingBST(TreeNode* root)
40         {
41             ret_val = NULL;
42             cur      = NULL;
43             __midOrder(root);
44             if(cur)
45             {
46                 cur->left = NULL;
47                 cur->right = NULL;
48             }
49             return ret_val;
50
51         }
```

```
52  };
53
54  /*
55  执行结果:
56  通过
57  显示详情
58  执行用时 :44 ms, 在所有 cpp 提交中击败了91.46%的用户
59  内存消耗 :14.7 MB, 在所有 cpp 提交中击败了100.00%的用户
60  */
```

AlimyBreak
2019.10.18