```
/*
返回与给定先序遍历 preorder 相匹配的二叉搜索树（binary search tree）的根结点。
(回想一下，二叉搜索树是二叉树的一种，其每个节点都满足以下规则，对于 node.left 的任何后代，值总 < node.val，而 node.right 的任何后代，值总 > node.val。此外，先序遍历首先显示节点的值，然后遍历 node.left，接着遍历 node.right。）
示例：
输入：[8,5,1,7,10,12]
输出：[8,5,10,1,7,null,12]
提示：
    1 <= preorder.length <= 100
    先序 preorder 中的值是不同的。
在真实的面试中遇到过这道题？
来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/construct-binary-search-tree-from-preorder-traversal
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:

- 直接根据先序遍历的顺序(根左右)来进行二叉树的建立,为了满足二叉搜索树的性质还要对新插入的节点值做判断.
- (吐槽一下,C语言里面没有传引用只能多级指针操作真的又丑又麻烦

方法一:C/C++_递归法

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution
{

    void add_node(TreeNode*& node, int val)
    {
        if(node == NULL)
        {
            node = new TreeNode(val);
            return;
        }

        if( val < node->val)
        {
            add_node(node->left,val);
        }
```

```
        else
        {
            add_node(node->right,val);
        }
    }

    public:
        TreeNode* bstFromPreorder(vector<int>& preorder)
        {
            TreeNode*   root    =   NULL    ;
            int         i       =   0       ;

            for(i = 0 ; i < preorder.size();i++)
            {
                add_node(root,preorder[i]);
            }
            return root;
        }
};


/*
执行结果：
通过
显示详情
执行用时 :8 ms, 在所有 cpp 提交中击败了74.75% 的用户
内存消耗 :10.8 MB, 在所有 cpp 提交中击败了70.49%的用户
*/
```

AlimyBreak
2019.11.03