```
/*
给定二叉树的根节点 root，找出存在于不同节点 A 和 B 之间的最大值 V，其中 V = |A.val - B.val|，且 A 是 B 的祖先。

（如果 A 的任何子节点之一为 B，或者 A 的任何子节点是 B 的祖先，那么我们认为 A 是 B 的祖先）



示例：

输入：[8,3,10,1,6,null,14,null,null,4,7,13]
输出：7
解释：
我们有大量的节点与其祖先的差值，其中一些如下：
|8 - 3| = 5
|3 - 7| = 4
|8 - 1| = 7
|10 - 13| = 3
在所有可能的差值中，最大值 7 由 |8 - 1| = 7 得出。



提示：

    树中的节点数在 2 到 5000 之间。
    每个节点的值介于 0 到 100000 之间。

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/maximum-difference-between-node-and-ancestor
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:

- 方法一:两重DFS,第一重遍历所有节点,第二重遍历该节点所有后代
- 方法二:一重DFS,最值法,逻辑上只要通过叶子节点回溯到根节点,找出最大值和最小值即可,但我们不是双向节点,所以我们需要两个变量$_max$, $_min$来保存DFS过程到达当前节点所经过的节点之间的最大值和最小值,直到在叶子节点进行结算比较即可.

方法一:C++_两重DFS

```
/**
 * 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
{
    private:


        int ret_val = 0;


        void __dfsSubTree(TreeNode* pnode,TreeNode* cnode)
        {
            int temp = 0;
            if(cnode==NULL)
            {
                return ;
            }
            /*根左右*/
            temp = abs(pnode->val - cnode->val);
            if( temp > ret_val)
            {
                ret_val = temp;
            }

            __dfsSubTree(pnode,cnode->left);
            __dfsSubTree(pnode,cnode->right);

        }

        void __dfsTree(TreeNode* node)
        {
            if(node==NULL)
            {
                return;
            }
            // 根左右
            __dfsSubTree(node,node->left);
            __dfsSubTree(node,node->right);
            __dfsTree(node->left);
            __dfsTree(node->right);

        }

    public:

        int maxAncestorDiff(TreeNode* root)
        {
            ret_val = 0;
            __dfsTree(root);
            return ret_val;
        }
};


/*
执行结果：
通过
显示详情
```

```
执行用时 :192 ms, 在所有 cpp 提交中击败了11.40% 的用户
内存消耗 :11.4 MB, 在所有 cpp 提交中击败了100.00%的用户
*/
```

方法二:一重DFS,最值法

```
/**
 * 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
{
    private:
        int ret_val = 0;

        void __dfsTree( TreeNode*   node    ,
                        int         _max    ,
                        int         _min
                      )
        {
            if(node==NULL)
            {
                return ;
            }
            _max = max(node->val,_max);
            _min = min(node->val,_min);

            /*叶子节点*/
            if(node->left == NULL && node->right == NULL)
            {
                ret_val = max(ret_val,abs(_max-_min));
            }

            //
            __dfsTree(node->left,_max,_min);
            __dfsTree(node->right,_max,_min);
        }

    public:

        int maxAncestorDiff(TreeNode* root)
        {
            ret_val = 0;
            if(root)
            {
                __dfsTree(root,root->val,root->val);
            }
            return ret_val;
        }
};

```

```
51
52 /*
53 执行结果:
54 通过
55 显示详情
56 执行用时 :8 ms, 在所有 cpp 提交中击败了83.94% 的用户
57 内存消耗 :12.3 MB, 在所有 cpp 提交中击败了100.00%的用户
58 */
```

AlimyBreak
2019.10.28