```
/*
给定一个二叉树，检查它是否是镜像对称的。

例如，二叉树 [1,2,2,3,4,4,3] 是对称的。

    1
   / \
  2   2
 / \ / \
3  4 4  3
但是下面这个 [1,2,2,null,3,null,3] 则不是镜像对称的:

    1
   / \
  2   2
   \   \
   3    3
说明:

如果你可以运用递归和迭代两种方法解决这个问题，会很加分。

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/symmetric-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：

- 递归法:分别递归判断左右子树是否对称，直到叶子节点.
- 迭代法:层序遍历依次比较,直到叶子节点.

---

方法一:C_递归法

```
 /**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     struct TreeNode *left;
 *     struct TreeNode *right;
 * };
 */

bool __ismirror(    struct TreeNode*    p    ,
                    struct TreeNode*    q
                )
{
```

```
    if(  ( p == NULL )
       &&( q == NULL )
    )
    {
        return true;
    }

    if(  ( p == NULL )
       ||( q == NULL )
    )
    {
        return false;
    }

    if(p->val == q->val)
    {
        return __ismirror(p->left,q->right) && __ismirror(p->right,q->left);
    }
    return false;
}

bool isSymmetric(struct TreeNode* root)
{
    return __ismirror(root,root);
}

/*
执行结果：
通过
显示详情
执行用时:8 ms, 在所有 C 提交中击败了69.91%的用户
内存消耗 :7.9 MB, 在所有 C 提交中击败了100.00%的用户
*/
```

---

方法二:C++_迭代法

```
/**
 * 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
{
    public:
        bool isSymmetric(TreeNode* root)
        {
            vector<TreeNode*>   vt1                     ;
            vector<TreeNode*>   vt2                     ;
```

```
    bool                ret_val     =   true    ;
    int                 size_temp   =   0       ;
    int                 i           =   0       ;
    TreeNode*           temp1       =   NULL    ;
    TreeNode*           temp2       =   NULL    ;

if(  ( root == NULL             )
   ||(   ( root->left  == NULL )
      &&( root->right == NULL )
    )
)
{
    return true;
}
if(root->left)
{
    vt2.push_back(root->left);
}
if(root->right)
{
    vt2.push_back(root->right);
}
if(vt2.size()==1)
{
    return false;
}
while(1)
{
    vt1 = vt2   ;
    vt2.clear() ;
    size_temp = vt1.size() ;
    for(i=0;i<size_temp/2;i++)
    {
        temp1 = vt1[ 2*i+0 ];
        temp2 = vt1[ 2*i+1 ];
        if( temp1->val != temp2->val )
        {
            return false;
        }
        else
        {
            if(  (temp1->left!=NULL)
               &&(temp2->right!=NULL)
            )
            {
                vt2.push_back( temp1->left  );
                vt2.push_back( temp2->right );
            }
            if(  (temp1->right!=NULL)
               &&(temp2->left!=NULL)
            )
            {
                vt2.push_back( temp1->right  );
```

```
                        vt2.push_back( temp2->left   );
                    }

                    if(  (temp1->left==NULL)
                       &&(temp2->right!=NULL)
                    )
                    {
                        return false;
                    }

                    if(  (temp1->left!=NULL)
                       &&(temp2->right==NULL)
                    )
                    {
                        return false;
                    }

                    if(  (temp1->right==NULL)
                       &&(temp2->left!=NULL)
                    )
                    {
                        return false;
                    }

                    if(  (temp1->right!=NULL)
                       &&(temp2->left==NULL)
                    )
                    {
                        return false;
                    }
                }
            }
            if(vt2.empty())
            {
                break;
            }
        }
        return ret_val;
    }
};

/*
执行结果：
通过
显示详情
执行用时 :8 ms, 在所有 C++ 提交中击败了78.79%的用户
内存消耗 :14.7 MB, 在所有 C++ 提交中击败了88.79%的用户
*/
```

AlimyBreak
2019.08.05