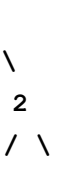
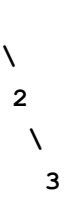
对称的二叉树

请实现一个函数，用来判断一棵二叉树是不是对称的。如果一棵二叉树和它的镜像一样，那么它是对称的。

例如，二叉树 **[1,2,2,3,4,4,3]** 是对称的。



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



存在问题的求解：先镜像 在比较是否相等

```
/**
 * 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:
    TreeNode* dfs(TreeNode* r)
    {
        if(r==nullptr) return nullptr;
        TreeNode* lval=dfs(r->left);
        TreeNode* rval=dfs(r->right);
        r->left=rval;
        r->right=lval;
        return r;
    }
    bool isequal(TreeNode* A,TreeNode* B)
    {
        if(A==nullptr||B==nullptr)
        {
            if(A==nullptr&&B==nullptr)
```

```
                return true;
            else
                return false;
        }

        if(A->val==B->val)
        {
            return isequal(A->left,B->left)&&isequal(A->right,B->right);
        }
        else
            return false;
    }
    bool isSymmetric(TreeNode* root) {
        if(root==nullptr) return true;
        TreeNode* mirror=dfs(root);
        return isequal(root,mirror);
    }
};
```

正解：直接比较 根结点下左子树 和另一根节点下右子树

```
class Solution {
public:
    bool isSymmetric(TreeNode* root) {
        if(!root) return true;
        return judge(root -> left, root -> right);
    }
    bool judge(TreeNode *l, TreeNode *r)
    {
        if(!l && !r) return true; // 如果既没有左子树也没有右子树，true
        if(!l || !r) return false; // 如果有右没左或者有左没右，false
        if(l -> val != r -> val) return false; // 如果左右都有，但是值不相等，
false
        return judge(l -> left, r -> right) && judge(l -> right, r -> left)
;
    }
};
```