```
/*
如果二叉树每个节点都具有相同的值，那么该二叉树就是单值二叉树。

只有给定的树是单值二叉树时，才返回 true；否则返回 false。
提示：

    给定树的节点数范围是 [1, 100]。
    每个节点的值都是整数，范围为 [0, 99] 。

*/
```

分析：

- 二叉树的题目，各种遍历往上抡就完了。
- 特殊输入$NULL$，特殊处理.
- 方法一：BFS广度优先遍历
- 方法二：根左右DFS前序优先遍历.

方法一： C++_Solution,BFS

```
/**
 * 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 isUnivalTree(TreeNode* root)
        {
            int                 val         = 0     ;
            int                 num_1       = 0     ;
            int                 num_2       = 0     ;
            int                 i           = 0     ;
            TreeNode*           temp        = NULL  ;
            queue<TreeNode*>    qu                  ;


            if(root==NULL)
            {
                return true;
            }
            val = root->val;
            qu.push(root);
```

```
            num_1 = 0;
            num_2 = 1;
            while(num_2!=0)
            {
                num_1 = num_2;
                num_2 = 0;
                for(i = 0; i < num_1;i++)
                {
                    temp = qu.front();
                    if(val!=temp->val)
                    {
                        return false;
                    }
                    if(temp->left != NULL)
                    {
                        qu.push(temp->left);
                        num_2++;
                    }
                    if(temp->right !=NULL)
                    {
                        qu.push(temp->right);
                        num_2++;
                    }
                    qu.pop();
                }
            }
            return true;
        }
};

/*
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61.30% 的用户
内存消耗 :10.5 MB, 在所有 C++ 提交中击败了97.15%的用户
*/
```

方法二： C++_Solution,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:
```

```
        bool dfs(TreeNode* node)
        {
            if(node==NULL)
            {
                return true;
            }

            if(node->val != val)
            {
                return false;
            }
            return dfs(node->left) && dfs(node->right);
        }
        int val = 0;

    public:
        bool isUnivalTree(TreeNode* root)
        {
            if(root==NULL)
            {
                return true;
            }
            val = root->val;
            return dfs(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61.30% 的用户
内存消耗 :10.3 MB, 在所有 C++ 提交中击败了99.64%的用户
*/
```

方法三：C++_Solution BFS的常规写法

```
/**
 * 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 isUnivalTree(TreeNode* root)
    {
        int         val     =   0;
```

```
        TreeNode*   temp    =   NULL;
        queue<TreeNode*> qu;

        if(root==NULL)
        {
            return true;
        }
        val = root->val;
        qu.push(root);
        while(qu.empty()!=true)
        {
            temp = qu.front();
            if(temp->val !=val)
            {
                return false;
            }
            if(temp->left)
            {
                qu.push(temp->left);
            }
            if(temp->right)
            {
                qu.push(temp->right);
            }
            qu.pop();
        }
        return true;
    }
};
/*
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61.30% 的用户
内存消耗 :10.6 MB, 在所有 C++ 提交中击败了84.34%的用户
*/
```

AlimyBreak
2019.07.24