```
/*
给定一个 N 叉树，找到其最大深度。
最大深度是指从根节点到最远叶子节点的最长路径上的节点总数。
例如，给定一个 3叉树 :
           1
       3   2   4
     5   6
我们应返回其最大深度，3。
说明:
    树的深度不会超过 1000。
    树的节点总不会超过 5000。
来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/maximum-depth-of-n-ary-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：

- D叉树的深度问题，BFS或DFS直接往上抡就完了.
- DFS还涉及深度排序问题,较之BFS会更慢.

方法一:C++ BFS

```
class Solution
{
    public:
        int maxDepth(Node* root)
        {
            int             ret_val         = 0         ;
            int             num_1           = 0     ;
            int             num_2           = 0     ;
            int             i               = 0     ;
            int             j               = 0     ;
            Node*           temp            = NULL  ;
            queue<Node*>    qu                      ;

            do
            {
                if(root==NULL)
                {
                    break;
                }

                qu.push(root);
                num_1 = 0;
                num_2 = 1;

                while(num_2!=0)
                {
                    num_1 = num_2;
                    num_2 = 0;
                    ret_val ++;

                    for(i = 0; i < num_1;i++)
                    {
                        temp = qu.front();
                        for(j=0;j<temp->children.size();j++)
                        {
                            qu.push(temp->children[j]);
                            num_2++;
                        }
                        qu.pop();
                    }
                }
            }while(0);

            return ret_val;
        }
};

/*
执行结果：
通过
显示详情
执行用时 :216 ms, 在所有 C++ 提交中击败了71.84% 的用户
内存消耗 :32.4 MB, 在所有 C++ 提交中击败了28.54%的用户
*/
```

方法二:C++ DFS

```
/*
// Definition for a Node.
class Node {
public:
    int val;
    vector<Node*> children;

    Node() {}

    Node(int _val, vector<Node*> _children) {
        val = _val;
        children = _children;
    }
};
*/
class Solution
{

    int __dfs(Node* node,int cur_level)
    {
        vector<int> vi;
        int size    =   0;
        int i       =   0;
        int temp    =   0;

        if(node==NULL)
        {
            return cur_level;
        }

        size = node->children.size();
        if(size==0)
        {
            return (cur_level+1);
        }
        else
        {
            for(i = 0;i<size;i++)
            {
                vi.push_back(__dfs(node->children[i],cur_level+1));
            }
            temp = vi[0];
            for(i = 1;i<size;i++)
            {
                if(vi[i]>temp)
                {
                    temp = vi[i];
                }
            }
            return temp;
        }
    }
public:
    int maxDepth(Node* root)
    {
        int             ret_val =   0       ;
        int             num_1   =   0       ;
        int             num_2   =   0       ;
        int             i       =   0       ;
        int             j       =   0       ;
        Node*           temp    =   NULL    ;
        queue<Node*>    qu                  ;

        do
        {
            if(root==NULL)
            {
                break;
            }
            ret_val = __dfs(root,0);
        }while(0);
        return ret_val;
    }
};
```

```
/*
执行结果：
通过
显示详情
执行用时 :300 ms, 在所有 C++ 提交中击败了37.84% 的用户
内存消耗 :33.2 MB, 在所有 C++ 提交中击败了5.04%的用户
*/
```

AlimyBreak
2019.07.25