### 二叉树层序遍历

// 先建立一棵树

**function** node(obj) {

**var** newobj = {};

**if**(obj.length == 1) {

        newobj.value = obj[0];

    }**else** **if**(obj.length == 0){

        newobj.value == **null**;

    }**else**{

        newobj.left = node(obj[0]);

        newobj.value = obj[1];

        newobj.right = node(obj[2]);

    }

**return** newobj;

}

**var** root = node([[[[7], 11, [2]], 4, []], 5, [[13], 8, [[], 4, [1]]]])

// 中序遍历

**function** goTree1(tree) {

**if**(tree.hasOwnProperty('left')){

        goTree1(tree.left);

        console.log(tree.value);

        goTree1(tree.right);

    }**else**{

        (tree.value == **null**)? console.log(""): console.log(tree.value);

    }

}

// 前序遍历

**function** goTree2(tree) {

    (tree.value == **null**)? console.log(""): console.log(tree.value);

**if**(tree.hasOwnProperty('left')){

        goTree2(tree.left);

        goTree2(tree.right);

    }

}

// 后序遍历

**function** goTree3(tree) {

**if**(tree.hasOwnProperty('left')){

        goTree3(tree.left);

        goTree3(tree.right);

        console.log(tree.value);

    }**else**{

        (tree.value == **null**)? console.log(""): console.log(tree.value);

    }

}

//层序遍历

// 层序遍历需要使用队列的数据结构，这里可以用数组来替代

// 这里不能使用迭代

**function** goTree4(tree) {

    let deque = [];

    deque.push(root);

**while**(deque.length != 0){

        front = deque.shift()

**if**(front.hasOwnProperty('left')){

            deque.push(front.left);

            deque.push(front.right);

        }

        (front.value == **null**)? console.log(""): console.log(front.value);

    }

}

### B树的特性，B树和B+树的区别

一个m 阶的B树满足以下条件：

每个结点至多拥有m棵子树；

根结点至少拥有两颗子树（存在子树的情况下）；

除了根结点以外，其余每个分支结点至少拥有m/2 棵子树；

所有的叶结点都在同一层上；

有k 棵子树的分支结点则存在 k-1 个关键码，关键码按照递增次序进行排列；

关键字数量需要满足ceil(m/2)-1 <= n <= m-1；

B树和B+树的区别：

以一个m阶树为例。

1、关键字的数量不同；B+树中分支结点**有m个关键字**，其叶子结点也有m个，其关键字只是起到了一个索引的作用，但是B树虽然也有m个子结点，但是其只**拥有m-1个关键字**。

2、存储的位置不同；**B+树中的数据都存储在叶子结点**上，也就是其所有叶子结点的数据组合起来就是**完整的数据**，但是**B树的数据存储在每一个结点**中，并不仅仅存储在叶子结点上。

3、分支结点的构造不同；B+树的分支结点**仅仅存储着关键字信息和儿子的指针**（这里的指针指的是磁盘块的偏移量），也就是说内部结点仅仅包含着索引信息。

4、查询不同；B树在**找到具体的数值以后，则结束**，而B+树则需要通过**索引找到叶子结点中的数据才结束**，也就是说**B+树的搜索过程中走了一条从根结点到叶子结点的路径**。