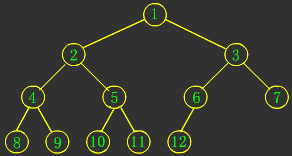
1、满二叉树

定义：从形象上来说满二叉树是一个绝对的三角形，也就是说它的最后一层全部是叶子节点，其余各层全部是非叶子节点。

另外一个定义：深度为k且有2^k-1个结点的二叉树。

2、完全二叉树



如上图，完全二叉树只比满二叉树少了最后几个结点，所以满二叉树是一种特殊的完全二叉树。

3、平衡二叉树

左右两个子树的高度差的绝对值不超过 1。

它是一棵空树或它的左右两个子树的高度差的绝对值不超过1，并且左右两个子树都是一棵平衡二叉树（递归）。

4、最优二叉树（赫夫曼树）

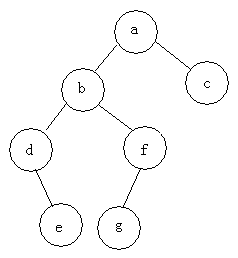
这是一种带权路径长度最短的树。哈夫曼编码就是哈夫曼树的应用。

前序遍历：根节点->左子树->右子树

中序遍历：左子树->根节点->右子树

后序遍历：左子树->右子树->根节点

例如：求下面树的三种遍历



前序遍历：abdefgc

中序遍历：debgfac

后序遍历：edgfbca

**一、已知前序、中序遍历，求后序遍历**

例：

前序遍历:         GDAFEMHZ

中序遍历:         ADEFGHMZ

画树求法：

第一步，根据前序遍历的特点，我们知道根结点为G

第二步，观察中序遍历ADEFGHMZ。其中root节点G左侧的ADEF必然是root的左子树，G右侧的HMZ必然是root的右子树。

第三步，观察左子树ADEF，左子树的中的根节点必然是大树的root的leftchild。在前序遍历中，大树的root的leftchild位于root之后，所以左子树的根节点为D。

第四步，同样的道理，root的右子树节点HMZ中的根节点也可以通过前序遍历求得。在前序遍历中，一定是先把root和root的所有左子树节点遍历完之后才会遍历右子树，并且遍历的左子树的第一个节点就是左子树的根节点。同理，遍历的右子树的第一个节点就是右子树的根节点。

第五步，观察发现，上面的过程是递归的。先找到当前树的根节点，然后划分为左子树，右子树，然后进入左子树重复上面的过程，然后进入右子树重复上面的过程。最后就可以还原一棵树了。该步递归的过程可以简洁表达如下：

1 确定根,确定左子树，确定右子树。

2 在左子树中递归。

3 在右子树中递归。

4 打印当前根。

那么，我们可以画出这个二叉树的形状：



那么，根据后序的遍历规则，我们可以知道，后序遍历顺序为：AEFDHZMG

**二、已知中序和后序遍历，求前序遍历**

依然是上面的题，这次我们只给出中序和后序遍历：

中序遍历:     ADEFGHMZ

后序遍历:       AEFDHZMG

画树求法：

第一步，根据后序遍历的特点，我们知道后序遍历最后一个结点即为根结点，即根结点为G。

第二步，观察中序遍历ADEFGHMZ。其中root节点G左侧的ADEF必然是root的左子树，G右侧的HMZ必然是root的右子树。

第三步，观察左子树ADEF，左子树的中的根节点必然是大树的root的leftchild。在前序遍历中，大树的root的leftchild位于root之后，所以左子树的根节点为D。

第四步，同样的道理，root的右子树节点HMZ中的根节点也可以通过前序遍历求得。在前后序遍历中，一定是先把root和root的所有左子树节点遍历完之后才会遍历右子树，并且遍历的左子树的第一个节点就是左子树的根节点。同理，遍历的右子树的第一个节点就是右子树的根节点。

第五步，观察发现，上面的过程是递归的。先找到当前树的根节点，然后划分为左子树，右子树，然后进入左子树重复上面的过程，然后进入右子树重复上面的过程。最后就可以还原一棵树了。该步递归的过程可以简洁表达如下：

1 确定根,确定左子树，确定右子树。

2 在左子树中递归。

3 在右子树中递归。

4 打印当前根。

这样，我们就可以画出二叉树的形状，如上图所示，这里就不再赘述。

那么，前序遍历:         GDAFEMHZ