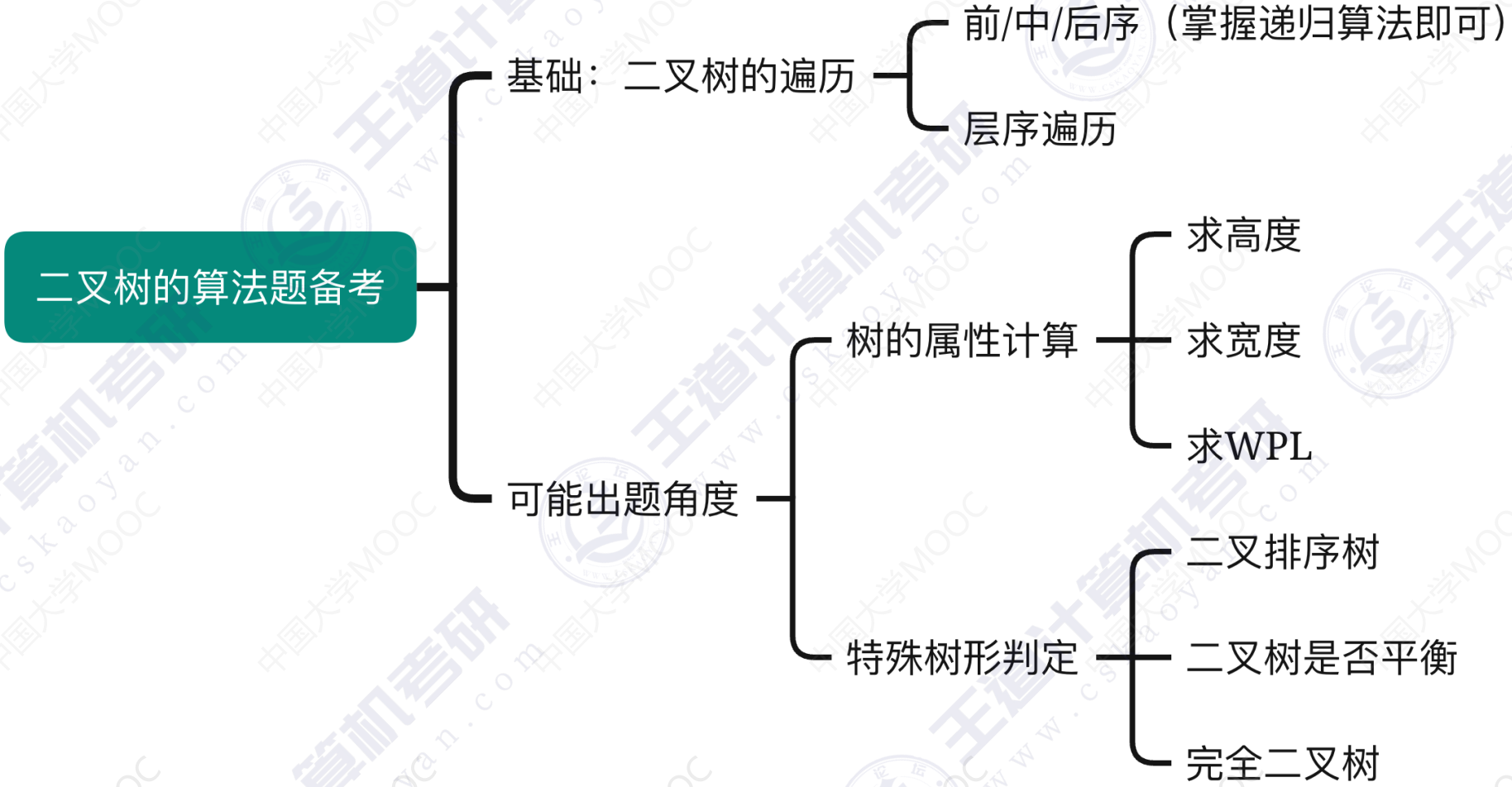
王道计算机考研强化课

算法题备考

二叉树

二叉树的算法题备考



自主训练

- 前/中/后序遍历代码（掌握递归实现即可）
- 层序遍历代码

- 求二叉树的高度
- 求二叉树的宽度
- 求二叉树的WPL

- 判定一棵二叉树是否为二叉排序树？
- 判定一棵二叉树是否平衡？
- 判定一棵二叉树是否为完全二叉树？

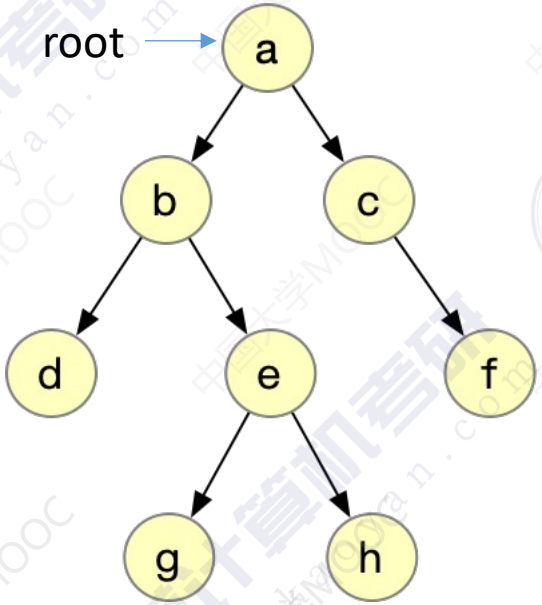
真题实战

- 2014 真题41题——二叉树算法题
- 2017 真题41题——二叉树算法题

处理二叉树时常用的代码思路

处理二叉树时常用的代码思路

- 递归遍历时，通过“参数传递”自顶向下传递信息
- 递归遍历时，通过“函数返回值”自底向上传递信息
- 使用全局变量，在各级函数间共享信息



基本功训练：前/中/后序遍历

假设链式存储的二叉树结点定义如下所示。请写出前/中/后序遍历代码。

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int data;                                  //数据域
   struct BiTNode    *lchild,*rchild;         //左、右孩子指针
}BiTNode,*BiTree;
```

基本功训练：前/中/后序遍历

假设链式存储的二叉树结点定义如下所示。请写出前/中/后序遍历代码。

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int data;                                  //数据域
   struct BiTNode    *lchild,*rchild;         //左、右孩子指针
}BiTNode,*BiTree;
```

```
//先序遍历二叉树
void PreOrder (BiTree root){
   if (root==NULL)
      return;
   visit(root);   //访问根节点
   PreOrder(root->lchild);
   PreOrder(root->rchild);
}
```

```
//中序遍历二叉树
void InOrder (BiTree root){
   if (root==NULL)
      return;
   InOrder(root->lchild);
   visit(root);   //访问根节点
   InOrder(root->rchild);
}
```

```
//后序遍历二叉树
void PostOrder (BiTree root){
   if (root==NULL)
      return;
   PostOrder(root->lchild);
   PostOrder(root->rchild);
   visit(root);   //访问根节点
}
```

基本功训练：层序遍历

假设链式存储的二叉树结点定义如下所示。请写出层序遍历代码。

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int data;                                  //数据域
   struct BiTNode    *lchild,*rchild;         //左、右孩子指针
}BiTNode,*BiTree;
```

注：可定义数据结构队列 Queue 存储二叉树结点。并且可使用以下基本操作处理队列。

```
//初始化队列
void InitQueue(Queue &Q)
//判断队列是否为空
bool IsEmpty(Queue Q)
//新元素x入队
void EnQueue(Queue &Q, BiTNode * x)
//队头元素出队，并使用 x 返回队头元素
void DeQueue(Queue &Q, BiTNode * &x)
```

基本功训练：层序遍历

假设链式存储的二叉树结点定义如下所示。请写出层序遍历代码。

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int data;                              //数据域
   struct BiTNode    *lchild,*rchild;     //左、右孩子指针
}BiTNode,*BiTree;
```

注：可定义数据结构队列 Queue 存储二叉树结点。并且可使用以下基本操作处理队列。

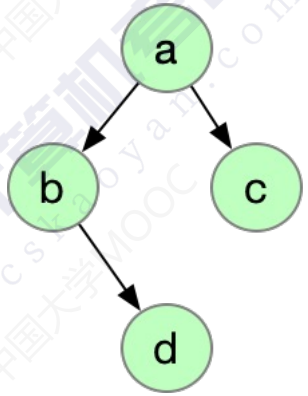
```
//初始化队列
void InitQueue(Queue &Q)
//判断队列是否为空
bool IsEmpty(Queue Q)
//新元素x入队
void EnQueue(Queue &Q, BiTNode * x)
//队头元素出队，并使用 x 返回队头元素
void DeQueue(Queue &Q, BiTNode * &x)
```

```
//层序遍历
void LevelOrder(BiTree T){
   Queue Q;
   InitQueue(Q);                  //初始化辅助队列
   BiTree p;
   EnQueue(Q,T);                  //将根结点入队

   while(!IsEmpty(Q)){            //队列不空则循环
      DeQueue(Q, p);              //队头结点出队
      visit(p);                   //访问队头元素
      if(p->lchild!=NULL)
         EnQueue(Q,p->lchild); //左孩子入队
      if(p->rchild!=NULL)
         EnQueue(Q,p->rchild);//右孩子入队
   }
}
```

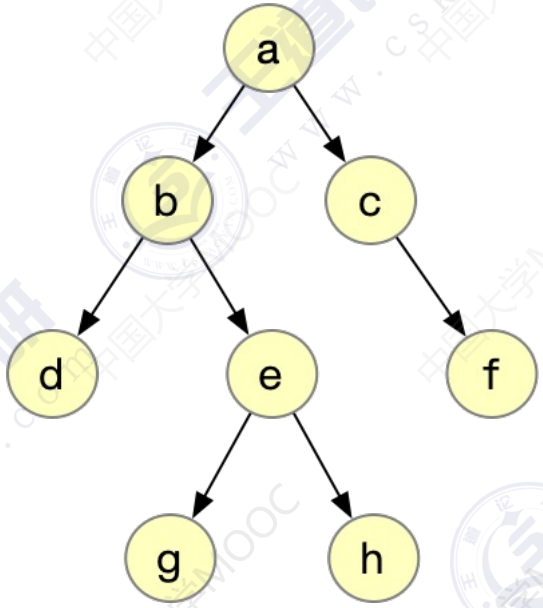
求二叉树的属性：求高度

假设链式存储的二叉树结点定义如下所示。请实现一个函数求树的高度。例如：二叉树旺财的高度=3，二叉树喵喵的高度=4。规定空二叉树高度=0。

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int data;                              //数据域
   struct BiTNode    *lchild,*rchild;     //左、右孩子指针
}BiTNode,*BiTree;
```



二叉树：旺财



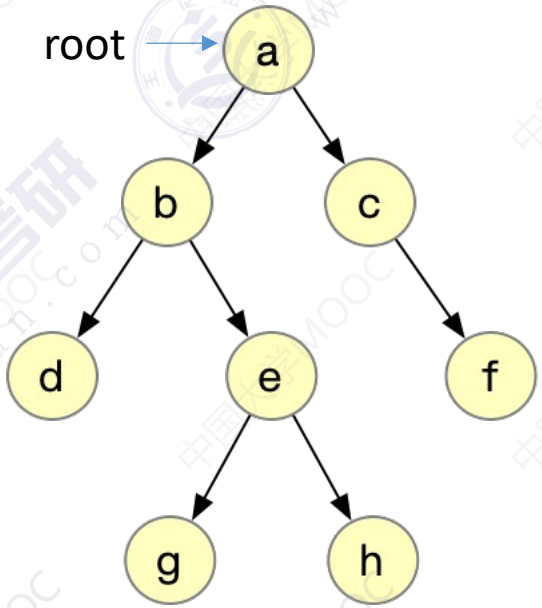
二叉树：喵喵

求二叉树的属性：求高度

假设链式存储的二叉树结点定义如下所示。请实现一个函数求树的高度。

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int data;                                  //数据域
   struct BiTNode    *lchild,*rchild;         //左、右孩子指针
}BiTNode,*BiTree;
```

```
/*求树的高度（方法一）*/
int height=0;          //用全局变量记录树的高度
void PreOrder (BiTree T, int n){
     if (T == NULL)
          return;
     if(n> height)     height=n;     //更新树的高度
     PreOrder (T->lchild, n + 1);     //遍历左子树
     PreOrder (T->rchild, n + 1);     //遍历右子树
}
```



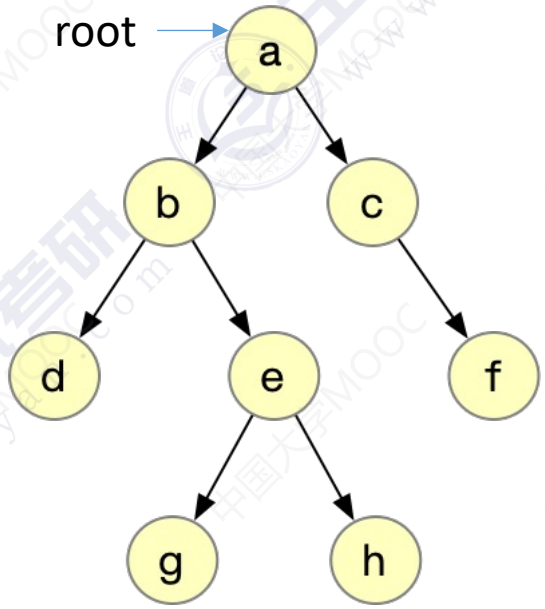
注：调用 PreOrder(root, 1)，即可求出树的高度，树高用全局变量height保存。

求二叉树的属性：求高度

假设链式存储的二叉树结点定义如下所示。请实现一个函数求树的高度。

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int data;                                  //数据域
   struct BiTNode    *lchild,*rchild;         //左、右孩子指针
}BiTNode,*BiTree;
```

```
/*求树的高度（方法二）*/
int PostOrder(BiTree T){
      if (T == NULL)
            return 0;
      int left = PostOrder(T->lchild);
      int right = PostOrder(T->rchild);
      //树的高度=Max(左子树高度，右子树高度)+1
      if (left>right)    return left+1;
      else               return right+1;
}
```

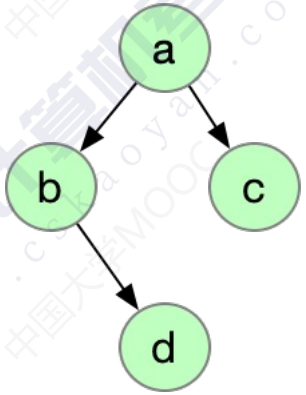


注：调用 PostOrder(root)，即可返回树深度

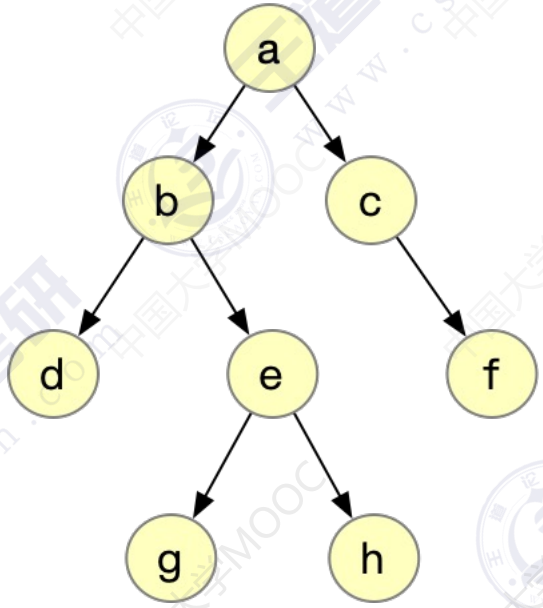
求二叉树的属性：求宽度

假设链式存储的二叉树结点定义如下所示。请实现代码求二叉树树的宽度。结点最多的一层中，结点数量即为二叉树的宽度。例如：二叉树旺财的宽度=2，二叉树喵喵的宽度=3。规定空二叉树宽度=0。

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int data;                              //数据域
   struct BiTNode    *lchild,*rchild;     //左、右孩子指针
}BiTNode,*BiTree;
```



二叉树：旺财



二叉树：喵喵

求二叉树的属性：求宽度

假设链式存储的二叉树结点定义如下所示。请实现代码求二叉树的宽度。

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int data;                                  //数据域
   struct BiTNode    *lchild,*rchild;         //左、右孩子指针
}BiTNode,*BiTree;
```

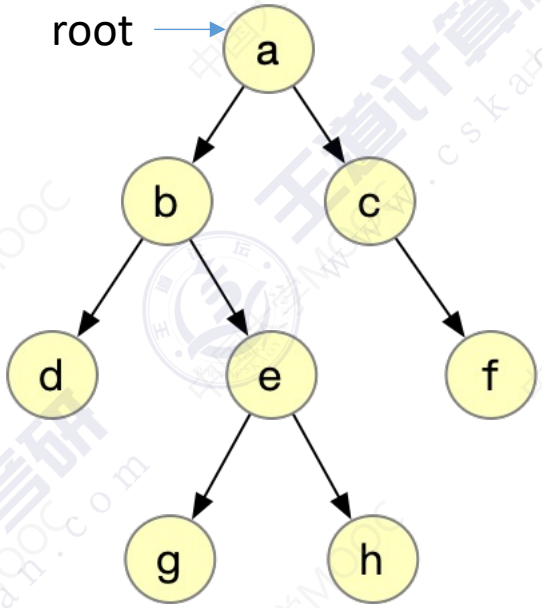
```
int width[MAX];  //用于统计各层的结点总数

//先序遍历，同时统计各层结点总数
void PreOrder (BiTree T, int level){
      if (T == NULL)  return;
      width[level]++;    //累加该层结点总数
      PreOrder(T->lchild, level + 1);     //遍历左子树
      PreOrder(T->rchild, level + 1);     //遍历右子树
}
```

```
//求树的宽度
void treeWidth (BiTree T) {
      for (int i=0; i<MAX; i++)
            width[i]=0;
      PreOrder(T, 0);  //先序遍历二叉树，统计各层结点总数
      int maxWidth = 0;      //找到最大宽度
      for (int i=0; i<MAX; i++){
            if(width[i]>maxWidth)
                  maxWidth =width[i];
      }
      printf("树的宽度是%d", maxWidth);
}
```

求二叉树的属性：求WPL

假设链式存储的二叉树结点定义如下所示。请实现一段代码，求二叉树的WPL（即：所有叶结点的带权路径长度之和）

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int weight;                          //权值
   struct BiTNode    *lchild,*rchild;   //左、右孩子指针
}BiTNode,*BiTree;
```

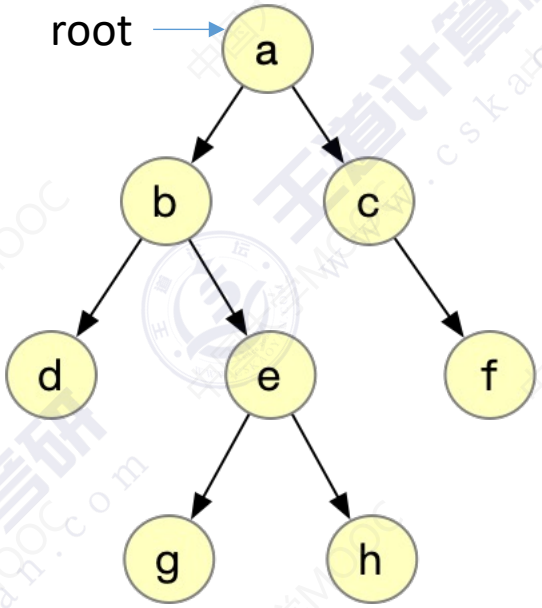


求二叉树的属性：求WPL

假设链式存储的二叉树结点定义如下所示。请实现一段代码，求二叉树的WPL（即：所有叶结点的带权路径长度之和）

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int weight;                          //权值
   struct BiTNode    *lchild,*rchild;   //左、右孩子指针
}BiTNode,*BiTree;
```

```
int WPL=0;  //用于累加叶结点的带权路径长度

//先序遍历，同时统计各层结点总数
void PreOrder (BiTree T, int level){
      if (T == NULL)  return;
      if(T->lchild==NULL && T->rchild==NULL)
            WPL += level*T->weight;      //累加叶结点带权路径长度
      PreOrder(T->lchild, level + 1);    //遍历左子树
      PreOrder(T->rchild, level + 1);    //遍历右子树
}
```



注：调用 PreOrder(root, 0)，即可求出二叉树的WPL，结果用全局变量 WPL 保存。

特殊树形判定：二叉排序树？

假设链式存储的二叉树结点定义如下所示。实现代码判断一棵二叉树是否为二叉排序树。

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
    int data;                                   //数据域
    struct BiTNode    *lchild,*rchild;          //左、右孩子指针
}BiTNode,*BiTree;
```

特殊树形判定：二叉排序树？

假设链式存储的二叉树结点定义如下所示。实现代码判断一棵二叉树是否为二叉排序树。

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
    int data;                                   //数据域
    struct BiTNode    *lchild,*rchild;          //左、右孩子指针
}BiTNode,*BiTree;
```

错误做法：遍历每个结点，判断是否满足 左<根<右

正确且简单的做法：中序遍历二叉树，判断是否能是一个升序序列。

特殊树形判定：二叉排序树？

假设链式存储的二叉树结点定义如下所示。实现代码判断一棵二叉树是否为二叉排序树。

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int data;                                  //数据域
   struct BiTNode    *lchild,*rchild;         //左、右孩子指针
}BiTNode,*BiTree;
```

错误做法：遍历每个结点，判断是否满足 左<根<右

正确且简单的做法：中序遍历二叉树，判断是否能是一个升序序列。

```
int temp=MIN_INT;  //记录当前遍历到的最小值
bool isBST=true;       //是否为二叉排序树？

void InOrder(BiTree T){
      if (T == NULL)  return;
      InOrder(T->lchild);
      if (T->data >= temp)       temp=T->data;
      else                  isBST=false;
      InOrder(T->rchild);
}
```

注：调用 InOrder(root)，执行结束后若 isBST=true 则是二叉排序树，若 isBST=false 则不是二叉排序树

特殊树形判定：是否平衡？

假设链式存储的二叉树结点定义如下所示。对于一棵二叉树，如果任何一个结点的左子树、右子树高度之差的绝对值不超过1，则该二叉树平衡。规定空二叉树的高度是0，且空二叉树也是平衡的。请实现代码判断一棵二叉树是否平衡。

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
    int data;                                   //数据域
    struct BiTNode    *lchild,*rchild;          //左、右孩子指针
}BiTNode,*BiTree;
```

特殊树形判定：是否平衡？

假设链式存储的二叉树结点定义如下所示。对于一棵二叉树，如果任何一个结点的左子树、右子树高度之差的绝对值不超过1，则该二叉树平衡。规定空二叉树的高度是0，且空二叉树也是平衡的。请实现代码判断一棵二叉树是否平衡。

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int data;                                  //数据域
   struct BiTNode    *lchild,*rchild;         //左、右孩子指针
}BiTNode,*BiTree;
```

注：调用 PostOrder(root)，执行结束后，若全局变量 isBalance=true，说明二叉树平衡；若 isBalance=false，说明二叉树不平衡

```
bool isBalance=true;              //二叉树是否平衡
int PostOrder(BiTree T){
      if (T == NULL)
            return 0;
      int left = PostOder(T->lchild);
      int right = PostOder(T->rchild);

      if (left-right>1)              isBalance=false;
      if (left-right<-1)             isBalance=false;

      //树的深度=Max(左子树深度，右子树深度)+1
      if (left>right)   return left+1;
      else              return right+1;
}
```

特殊树形判定：完全二叉树？

假设链式存储的二叉树结点定义如下所示。请实现代码判断一棵二叉树是否为完全二叉树

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int data;                              //数据域
   struct BiTNode    *lchild,*rchild;     //左、右孩子指针
}BiTNode,*BiTree;
```

注：判定“完全二叉树”，也可能用“简答”的形式考应用题

特殊树形判定：完全二叉树？

假设链式存储的二叉树结点定义如下所示。请实现代码判断一棵二叉树是否为完全二叉树

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int data;                                  //数据域
   struct BiTNode    *lchild,*rchild;         //左、右孩子指针
}BiTNode,*BiTree;
```

注：可定义数据结构队列 Queue 存储二叉树结点。并且可使用以下基本操作处理队列。

```
//初始化队列
void InitQueue(Queue &Q)
//判断队列是否为空
bool IsEmpty(Queue Q)
//新元素x入队
void EnQueue(Queue &Q, BiTNode * x)
//队头元素出队，并使用 x 返回队头元素
void DeQueue(Queue &Q, BiTNode * &x)
```

特殊树形判定：完全二叉树？

假设链式存储的二叉树结点定义如下所示。请实现代码判断一棵二叉树是否为完全二叉树

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int data;                              //数据域
   struct BiTNode    *lchild,*rchild;     //左、右孩子指针
}BiTNode,*BiTree;
```

注：可定义数据结构队列 Queue 存储二叉树结点。并且可使用以下基本操作处理队列。

```
//初始化队列
void InitQueue(Queue &Q)
//判断队列是否为空
bool IsEmpty(Queue Q)
//新元素x入队
void EnQueue(Queue &Q, BiTNode * x)
//队头元素出队，并使用 x 返回队头元素
void DeQueue(Queue &Q, BiTNode * &x)
```

```
//层序遍历
void LevelOrder(BiTree T){
   Queue Q;
   InitQueue(Q);                  //初始化辅助队列
   BiTree p;
   EnQueue(Q,T);                  //将根结点入队

   while(!IsEmpty(Q)){            //队列不空则循环
      DeQueue(Q, p);              //队头结点出队
      visit(p);                   //访问队头元素
      if(p->lchild!=NULL)
         EnQueue(Q,p->lchild); //左孩子入队
      if(p->rchild!=NULL)
         EnQueue(Q,p->rchild);//右孩子入队
   }
}
```

特殊树形判定：完全二叉树？

```
bool isComplete=true;          //是否为完全二叉树
bool flag=false;      //flag=true，表示层序遍历时出现过叶子或只有左孩子的分支节点

void visit(BiTNode * p){
      if(p->lchild==NULL && p->rchild==NULL)      flag = true;
      if(p->lchild==NULL && p->rchild!=NULL)      isComplete=false;  //不是完全二叉树
      if(p->lchild!=NULL && p->rchild==NULL){
            if (flag)    isComplete=false;  //不是完全二叉树
            flag = true;
      }
      if(p->lchild!=NULL && p->rchild!=NULL)
            if (flag)    isComplete=false;  //不是完全二叉树
}
```

特殊树形判定：完全二叉树？

假设链式存储的二叉树结点定义如下所示。请实现代码判断一棵二叉树是否为完全二叉树

```
//二叉树的结点定义（链式存储）
typedef struct BiTNode{
   int data;                              //数据域
   struct BiTNode    *lchild,*rchild;     //左、右孩子指针
}BiTNode,*BiTree;
```

注：判定“完全二叉树”，也可能用“简答”的形式考应用题。

eg:
（1）请给出二叉树结点的数据结构定义
（2）请回答，如何判断一棵非空二叉树是否是完全二叉树？