数据结构与算法

人工智能与大数据学院

本节课内容安排

一、二叉树遍历操作应用举例

二、哈夫曼树的应用练习

三、学习通作业

二叉树——存储结构

二叉树的链式存储结构——二叉链表存储结构

二叉链表中每个结点包括3个域：数据域、左孩子指针域和右孩子指针域。左、右孩子指针域分别指示左右孩子结点的存储地址。每个结点的两个指针域分出了两个叉，因此该存储结构被形象地称为二叉链表。

lchild	data	rchild

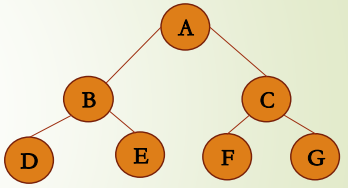
二叉链表结点的存储结构

```
typedef struct TreeNode {
    char val;
    struct TreeNode *left;
    struct TreeNode *right;
} TreeNode;
```

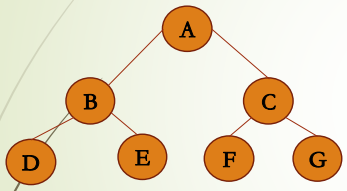
二叉树的遍历算法实现

手动创建一棵二叉树

```
TreeNode* root = createNode('A');
root->left = createNode('B');
root->left->left = createNode('D');
root->left->right = createNode('E');
root->right = createNode('C');
root->right->left = createNode('F');
root->right->right = createNode('G');
```



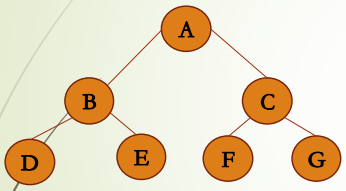
二叉树的遍历算法实现



先序遍历的结果是【A, B, D, E, C, F, G】

```
void preorderTraversal(TreeNode* root) {
    if (root == NULL) return;
    printf("%c ", root->val);
    preorderTraversal(root->left);
    preorderTraversal(root->right);
}
```

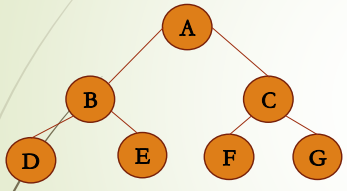
二叉树的遍历算法实现



中序遍历的结果是【D, B, E, A, F, C, G】

```
void inorderTraversal(TreeNode* root) {
    if (root == NULL) return;
    inorderTraversal(root->left);
    printf("%c ", root->val);
    inorderTraversal(root->right);
}
```

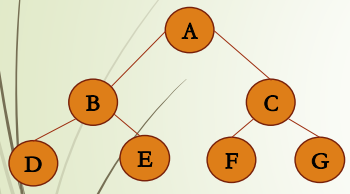
二叉树的遍历算法实现

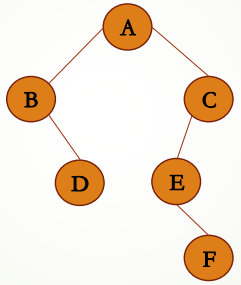


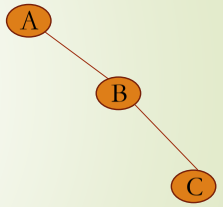
后序遍历的结果是【D, E, B, F, G, C, A】

```
void postorderTraversal(TreeNode* root) {
    if (root == NULL) return;
    postorderTraversal(root->left);
    postorderTraversal(root->right);
    printf("%c ", root->val);
}
```

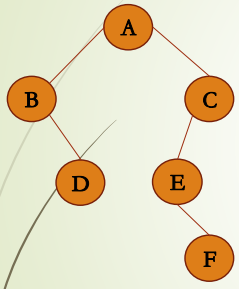
二叉树的遍历——计算树的深度





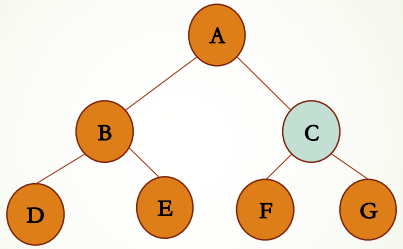


二叉树的遍历——计算树的深度

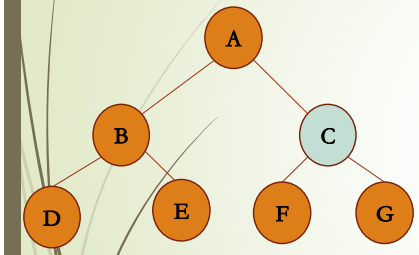



```
int getDepth(TreeNode* root){
    int l=0,r=0;
    if (root == NULL) return 0;
    l = getDepth(root->left);
    r = getDepth(root->right);
    return (l>r?l:r)+1;
}
```

二叉树的遍历——计算树中指定结点x所在的层数

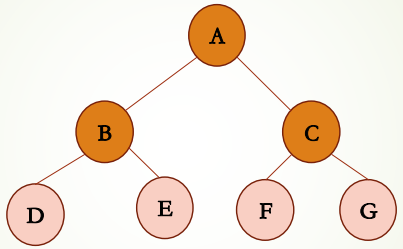


二叉树的遍历——计算树中指定结点x所在的层数

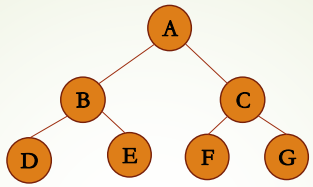



```
int getLevel(TreeNode* root, char x, int current_lev){
    int llev, rlev;
    if (root == NULL) {
        return -1;
    }
    if (root->val == x){
        return current_lev;
    }
    llev = getLevel(root->left, x, current_lev+1);
    if(llev!=-1) return llev;
    rlev = getLevel(root->right, x, current_lev+1);
    return rlev;
}
```

二叉树的遍历——计算二叉树中叶子结点的数量



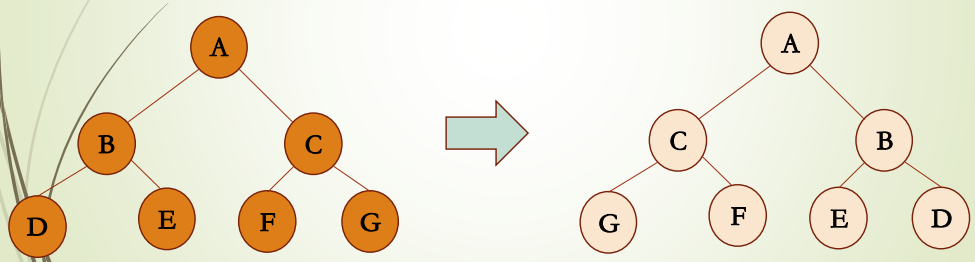
二叉树的遍历——计算二叉树中叶子结点的数量




```
int countLeaf(TreeNode* root){
   int m;
   if (root == NULL) return 0;
   if(root->left == NULL && root->right == NULL)
      m = 1;
   else
      m = countLeaf(root->left) + countLeaf(root->right);
   return m;
}
```

二叉树操作练习1：

以二叉链表为存储结构，编定算法将二叉树中所有结点的左右子树相互交换



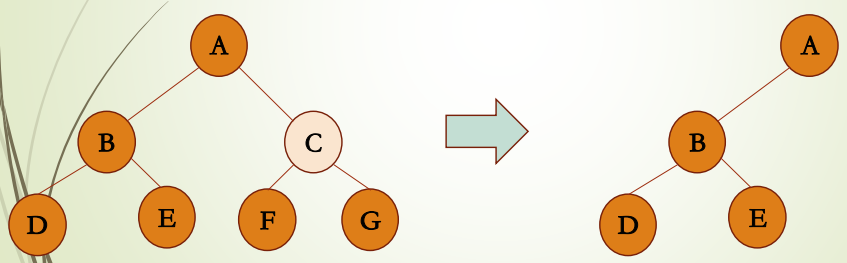
二叉树操作练习1：

以二叉链表为存储结构，编定算法将二叉树中所有结点的左右子树相互交换

```
void swapLeftRright(TreeNode* root){
   if (root == NULL) return;
   TreeNode* tmp = root->left;
   root->left = root->right;
   root->right = tmp;
   swapLeftRright(root->left);
   swapLeftRright(root->right);
}
```

二叉树操作练习2：

以二叉链表为存储结构，编定算法找到二叉树中值为x的结点，删去以它为根的子树，并释放相应的空间



二叉树操作练习2:

以二叉链表为存储结构，编定算法找到二叉树中值为x的结点，删去以它为根的子树，并释放相应的空间

```
void delSubtree(TreeNode* x){
   if (x == NULL) return;
   delSubtree(x->left);
   delSubtree(x->right);
   free(x);
}
```

完成学习通作业