《数据结构与算法》作业

22920212204392 黄勖

习题 3　树结构

3-1 设二叉树 T 中度为 1 的结点 11 个，度为 2 的结点 12 个，则二叉树 T 共有(　C　)个叶子结点。

(A) 11
(B) 12
(C) 13
(D) 36

3-2 设树 T 的度为 4，其中度为 1，2，3 和 4 的结点个数分别为 4，2，1，1，则 T 中的叶子数为(　D　)。

(A) 5
(B) 6
(C) 7
(D) 8

注：

一棵含有 n 个结点的树，有 n-1 个分支，即 n = 1*4 + 2*2 + 3*1 + 4*1 + 1 = 16;

又由于 n = n0 + n1 + n2 + n3 + n4 = n0 + 8;

n0 + 8 = 16，所有叶子结点个数为 8

3-3 已知一棵度为 k 的树中，有 n1 个度为 1 的结点，n2 个度为 2 的结点，…，nk 个度为 k 的结点。试计算该树的叶子结点数。

答：设该树中的叶子数为 n_0 个。该树中的总结点数为 n 个，则有：

$$n=n_0+n_1+n_2+\ldots+n_K \quad ①$$

$$n-1=0*n_0+1*n_1+2*n_2+\ldots+K*n_K \quad ②$$

联立①②方程组可得：

叶子数为：

$$n_0=1+0*n_1+1*n_2+2*n_3+\ldots+(K-1)*n_K$$

3-4 证明：如果二叉树 T 的叶子结点数为 n_0，度为 2 的结点数为 n_2，则 $n_0=n_2+1$。

证明：

假设二叉树的度为 0，1，2 的结点为 n_0,n_1,n_2,总节点数为 n

则有按照结点求和的

$$n = n_0 + n_1 + n_2 \quad ①$$

按照边求和得：

$$n = n_1 + 2 * n_2 + 1 \quad ②$$

所以 ② - ①可得

$$n_2 + 1 - n_0 = 0$$

即

$$n_0 = n_2 + 1 \quad ③$$

3-5 对于任意非空二叉树，要设计出其后序遍历的非递归算法而不使用栈结构，最适合的方法是对该二叉树采用(　B　)存储结构。

(A) 二叉链表
(B) 三叉链表
(C) 索引
(D) 顺序

注：三叉链表比二叉链表多一个指向父结点的指针

3-6 一棵二叉树的叶子结点在其先序、中序和后序序列中的相对位置(**C**)。

(A) 肯定发生变化
(B) 可能发生变化
(C) 不会发生变化
(D) 无法确定

3-7 设二叉树 T 按照二叉链表存储，则下列递归算法的主要功能是(**B**)。

```
int F(BiTree T)
{
    if (!T) return 0;
    x=F(T->Lchild);
    y=F(T->Rchild);
    if (y>x) x=y;
    return x+1;
}
```

(A) 交换二叉树 T 的左右子树
(B) 计算二叉树 T 的高度
(C) 计算二叉树 T 的叶子结点数
(D) 先遍历左子树，再遍历右子树

3-8 已知二叉树 T 的先序序列为 ABCDEF，中序序列为 CBAEDF, 则 T 的后序序列为(**A**)。

(A) CBEFDA
(B) FEDCBA
(C) CBEDFA
(D) 不确定

注：画图或按照 3-9 操作。

3-9 简述由先序序列和中序序列构造二叉树的基本操作方法。

答：

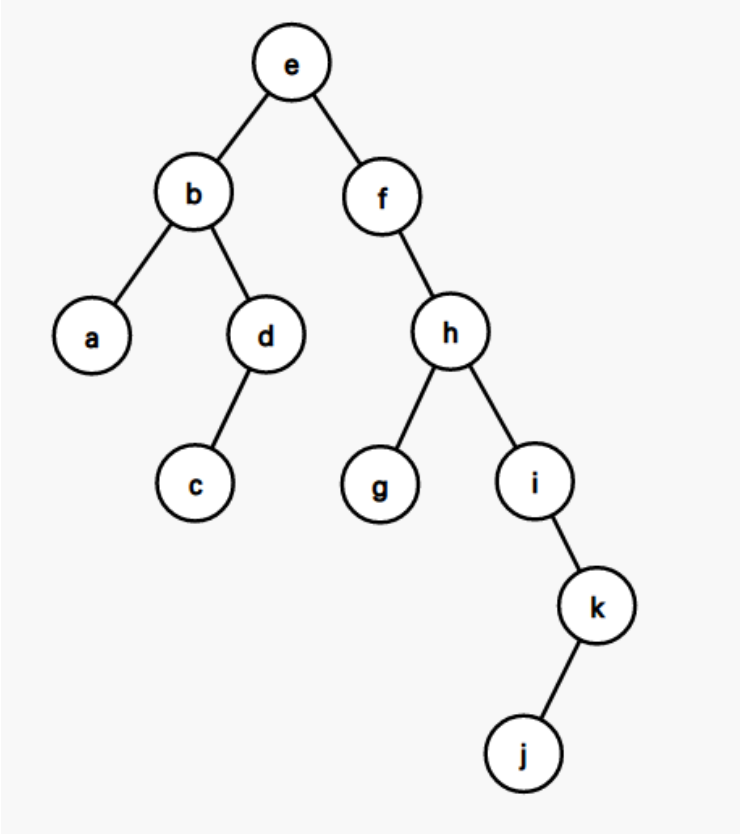
①取先序遍历序列的第一个值，用该值构造根结点，然后在中序遍历序列中查找与该元素相等的值，这样就可以把序列分为三部分：左子树（如果有）、根结点和右子树（如果有）。

②将两个序列都分成三部分，这样就分别形成了根结点的左子树和右子树的先序遍历和后序遍历的序列。

③重复①和②步骤，直至所有结点都处理完就可以完整构成一颗二叉树了。

3-10 已知二叉树的先序序列为 ebadcfhgjik，中序序列为 abcdefghijk，试画出该二叉树。

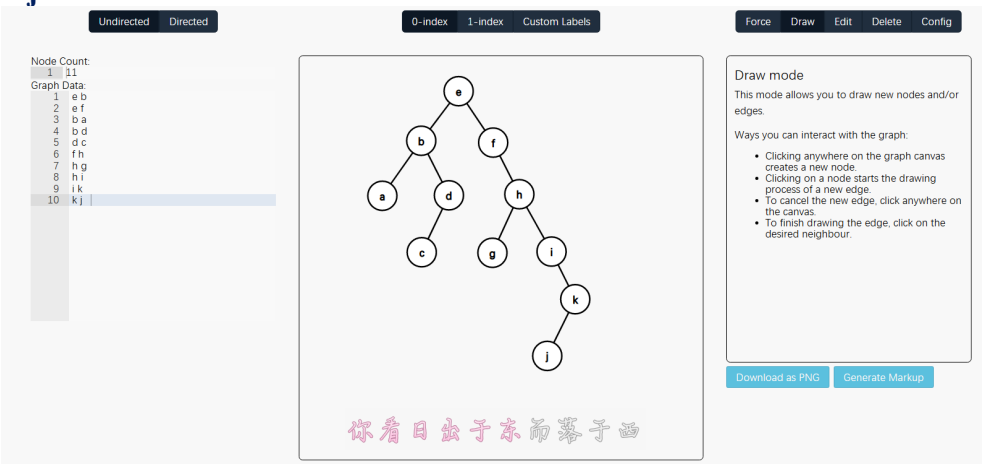
答：



注：画于 https://csacademy.com/app/graph_editor/

输入数据：

11
e b
e f
b a
b d
d c
f h
h g
h i
i k
k j



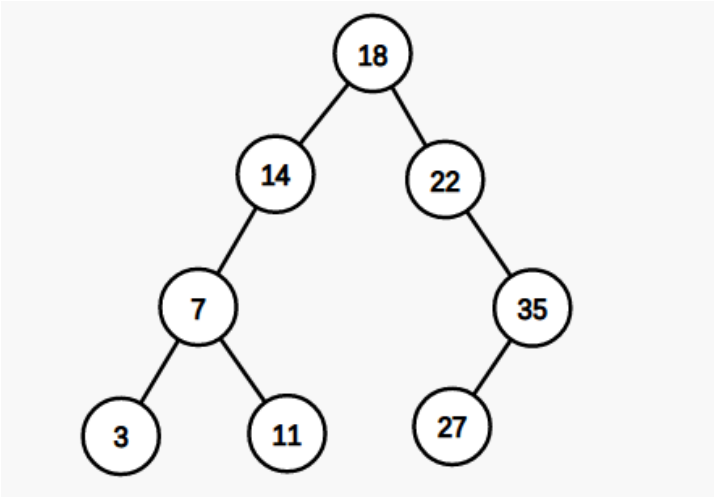
3-11 已知二叉树 T 的中序序列和后序序列分别为

(中序) 3, 7, 11, 14, 18, 22, 27, 35

(后序) 3, 11, 7, 14, 27, 35, 22, 18

试画出二叉树 T。

答：



输入数据：

8
18 14
18 22
14 7
7 3
7 11
22 35
35 27

3-12 已知二叉树 T 按照二叉链表存储，设计算法，计算 T 中叶子结点的数目。

答：

```
int Find(BiTree T) {
    if (!T) return 0;
    if (!T->Lchild && !T->Rchild) return 1;
    return F(T->Lchild)+F(T->Rchild);
}
```

3-13 已知二叉树 T 按照二叉链表存储，设计算法，交换 T 的左子树和右子树。

答：

```
typedef struct BiTNode {
    int data;
    struct BiTNode *lchild, *rchlid;
}BiTNode,*BiTree;
bool ChangLR(BiTree T)
{
    if (T->lchild == NULL && T->rchlid == NULL)
    {
        return false;
    }
```

```
    else     //交换当前左右子树
    {
        BiTree temp;
        temp = T->lchild;
        T->lchild = T->rchlid;
        T->rchlid = temp;
    }
    ChangLR(T->lchild);
    ChangLR(T->rchlid);
}
```

3-14 先序后继线索化算法是根据二叉链表建立先序后继线索二叉链表，其基本原则是在前驱空指针域中写入后继线索，即将右子树的(C)指针写入左子树的最后一个叶子结点右指针域。

(A) 线索
(B) 根结点
(C) 前驱结点
(D) 后继结点

3-15 设计算法，在先序线索二叉树中，查找给定结点 p 在先序序列中的后继。

答：

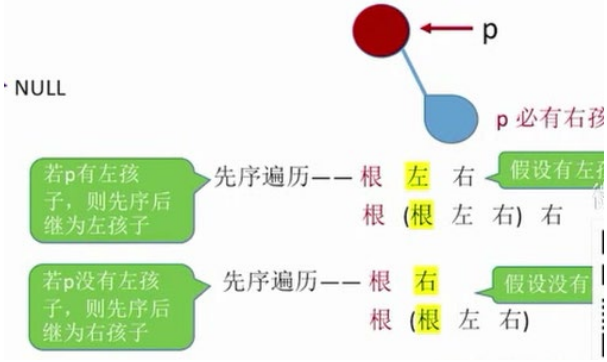
线索二叉树：根据某次遍历，在二叉树中的相关空指针域都写入线索(后继线索或前驱线索)，即成为线索二叉树。

先序后继：先序遍历中得到的后继。

在先序线索二叉树中找到指定结点*p 的先序后继 next

若 p->rtag==1（被线索化了），则 next=p->rchild（直接就是前驱）

若 p->rtag==0（没有被线索化），必定是有右孩子的




```
/*先序线索二叉树找后继*/
ThreadNode *find(ThreadNode *p){
    if(p->RTag == 1){
        p = p->rchild;
    }else{
        if(p->lchild){
            p = p->lchild;
        }else{
            p = p->rchild;
        }
    }
```

```
    if(p->data)
        return p;
    else
        return NULL; //最后一个节点无后继
}
```

3-16 对 n (n≥2)个权值均不相同的字符构造哈夫曼树 T，不正确的叙述(**A**)。

(A) T 一定是一棵完全二叉树

(B) T 中一定没有度为 1 的结点

(C) T 中两个权值最小的结点一定是兄弟结点

(D) T 中任一分支结点的权值一定不小于下一层任一结点的权值

3-17 设计一个求结点 x 在二叉树中的双亲结点算法。

答：

```
typedef struct node {
    datatype data;
    struct node *lchild,*rchild;
} bitree;
bitree *q[20];
int r=0,f=0,flag=0;
void preorder(bitree *bt, char x) { //记录先序遍历
    if (bt!=0 && flag==0)
        if (bt->data==x) {
            flag=1;
            return;
        } else {
            r=(r+1)% 20;
            q[r]=bt;
            preorder(bt->lchild,x);
            preorder(bt->rchild,x);
        }
}
void parent(bitree *bt,char x) { //求双亲结点
    int i;
    preorder(bt,x);
    for(i=f+1; i<=r; i++) if (q[i]->lchild->data==x || q[i]->rchild->data) break;
    if (flag==0) printf("not found x\n");
    else if (i<=r) printf("%c",bt->data);
    else printf("not parent");
}
```