



广州大学



数据结构

Data Structure

GU

主讲人：王国军

计算机科学与网络工程学院

csgjwang@gzhu.edu.cn
<http://trust.gzhu.edu.cn/>

办公室：行政西楼前座532室

根据严蔚敏老师《数据结构》（C语言版）（第2版）制作，仅供广州大学计算机、软件工程、网络工程及相关专业2024级本科生和任课老师使用。

第六章 树和二叉树



6.1 树的定义和基本术语

6.2 二叉树的定义

6.3 二叉树的存储结构

6.4 二叉树的遍历

6.5 线索二叉树

6.6 树和森林的表示方法

6.7 树和森林的遍历

6.8 哈夫曼树与哈夫曼编码

6.1 树的定义和基本术语



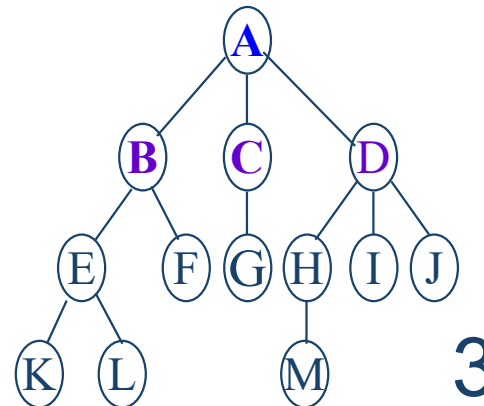
树是一类重要的非线性数据结构，是以分支关系定义的层次结构。

一、树的定义（递归定义）

1. 树(Tree)是 n ($n \geq 0$) 个结点的有限集 T 。

如果 $n = 0$ ，称为空树；否则：

- 有且仅有一个称为根(root)的结点。
- 其余结点可分为 m ($m \geq 0$) 个互不相交的有限集 $T_1, T_2, \dots, T_m$ ，其中每一个集合本身又是一棵树，称为根的子树(SubTree)。



2. 树的表示形式——是一棵倒立的树

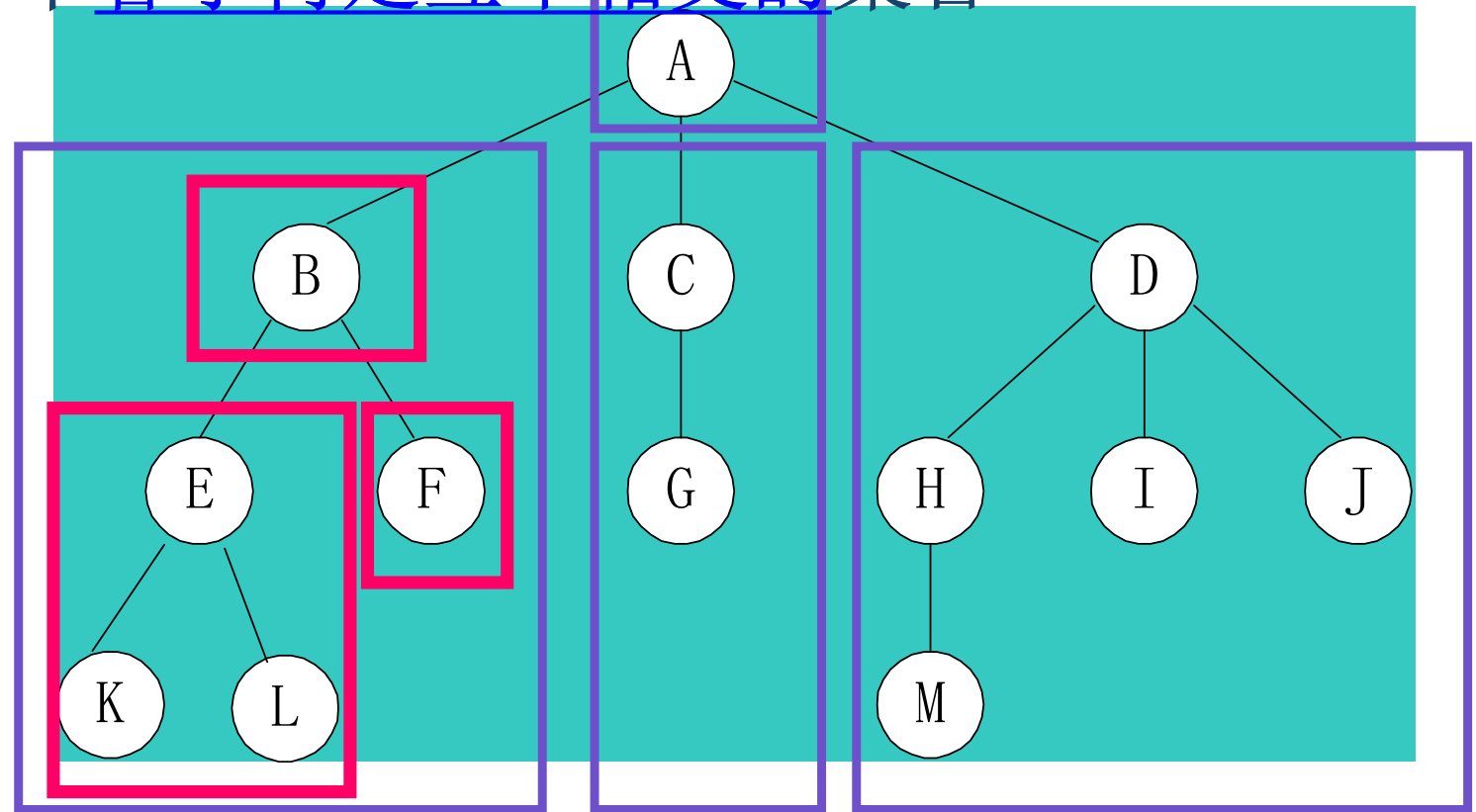


对于非空树，其特点：

- 树中至少有一个结点——根
- 树中各子树是互不相交的集合



只含有根
结点的树



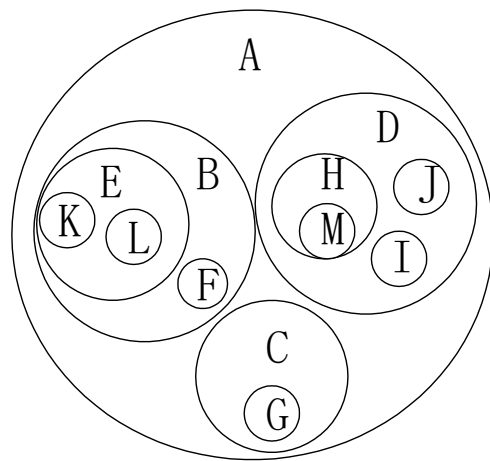
T_1

T_2

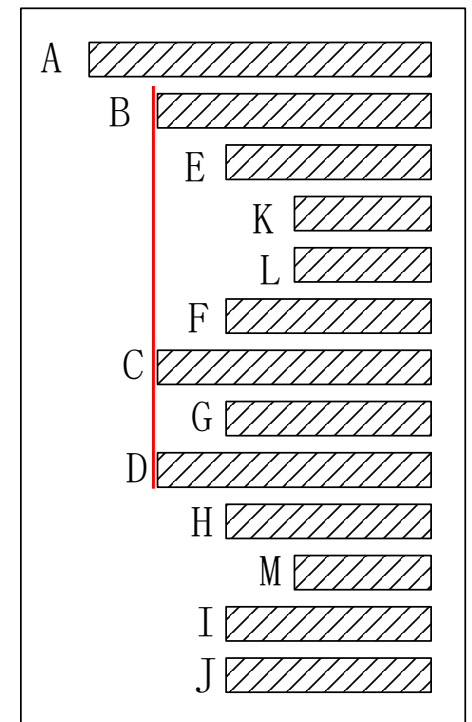
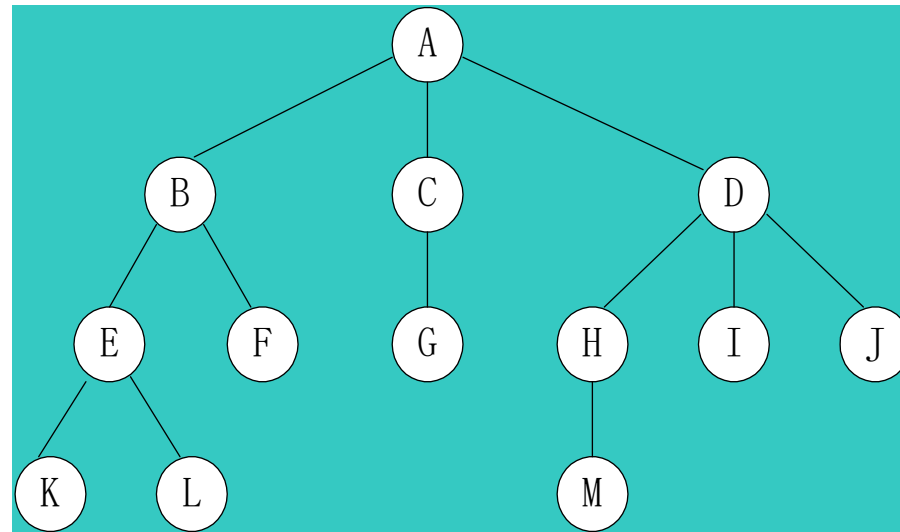
T_3

含有子树 T_1 、 T_2 和 T_3 的树

树的其它表示方式



嵌套集合



凹入表示法

$(A(B(E(K, L), F), C(G), D(H(M), I, J)))$

广义表

对比线性结构和树型结构的特点



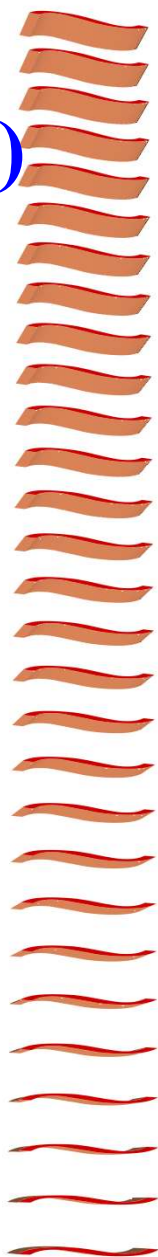
线性结构 (1:1)

$(a_1, a_2, a_3, \dots, a_{n-2}, a_{n-1}, a_n)$

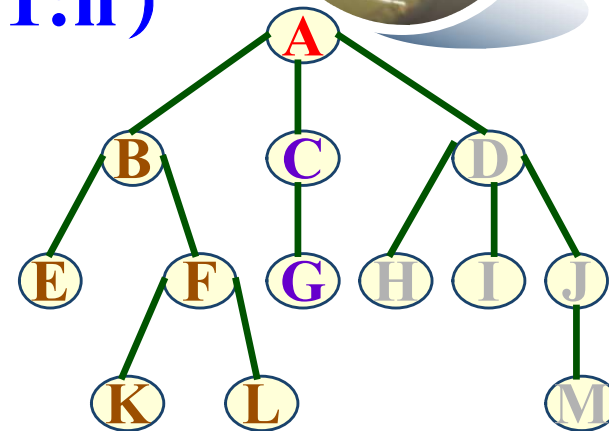
第一个数据元素
(无前驱)

最后一个数据元素
(无后继)

其它数据元素
(一个前驱、
一个后继)



树型结构 (1:n)



根结点
(无前驱)

多个叶子结点
(无后继)

其它数据元素
(一个前驱、
一个或多个后继)

树的三种存储结构



一、双亲表示法

二、孩子表示法

三、树的**二叉链表**（**孩子-兄弟**）存储表示法

一、双亲表示法



❖ 实现：定义结构数组存放树的结点，每个结点含两个域：

data	parent
------	--------

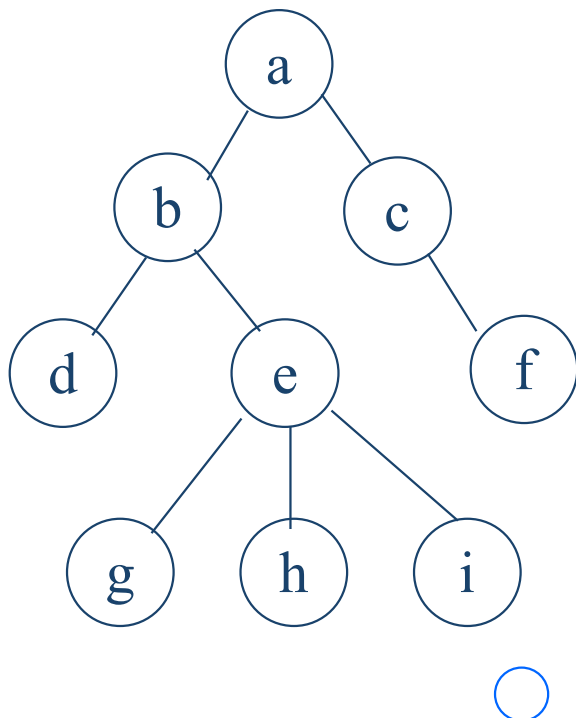
- 数据域（**data**）：存放结点本身信息
- 双亲域（**parent**）：指示本结点的双亲结点在数组中的位置

- 简单实现：

```
typedef struct node
{
    char data;
    int parent;
} PT;
PT tree[M];
```

■ 特点：找双亲容易，找孩子难

双亲表示法示意图



如何找某结点的
孩子结点？

	data	parent
0	a	9
1	b	0
2	b	1
3	c	1
4	d	2
5	e	2
6	f	3
7	g	5
8	h	5
9	i	5

0号单元不用或
用于存结点个数

该结点在数组中的位序与某个结点的parent相等

C语言类型描述



```
#define MAX_TREE_SIZE 100
```

```
typedef struct PTNode { //结点结构
```

```
    ElemType data;
```

```
    int parent; // 双亲位置域
```

```
} PTNode;
```

```
typedef struct { // 树结构
```

```
    PTNode nodes[MAX_TREE_SIZE];
```

```
    int n, r; // 结点个数和根结点的位置
```

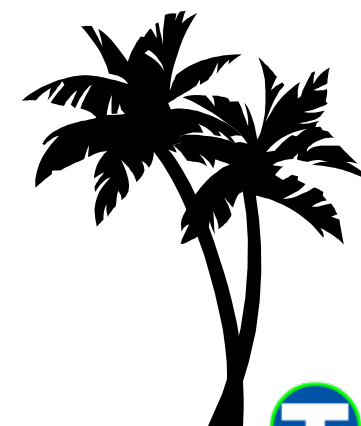
```
} PTree;
```

结点结构:

data	parent
------	--------



6.2 二叉树



6.2.1 二叉树的定义



一、二叉树定义

二叉树的递归定义：

其左、右子树又都是二叉树。

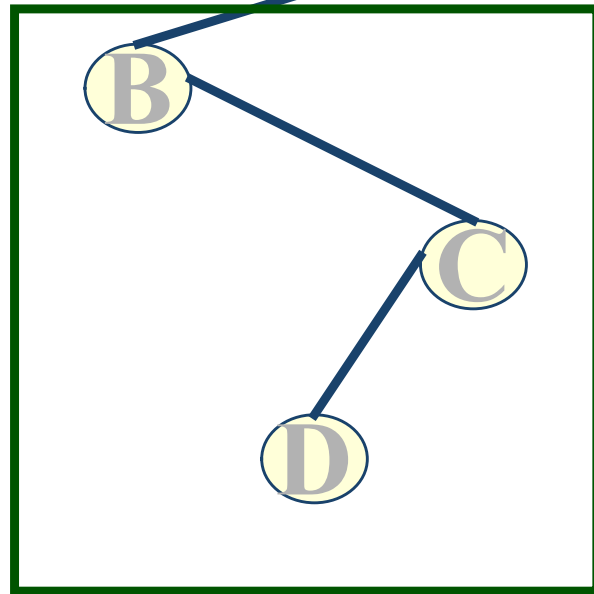
或：二叉

两棵分别称为左子树和右子树的、互不相交的二叉树组成

根结点

右子树

左子树

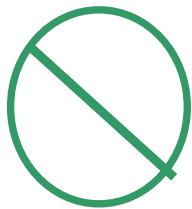


二、二叉树的五种基本形态



二叉树定义：度不大于2且子树有左右之分的树型结构。

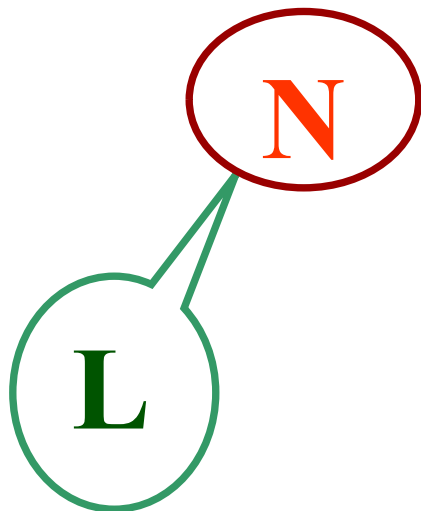
空树



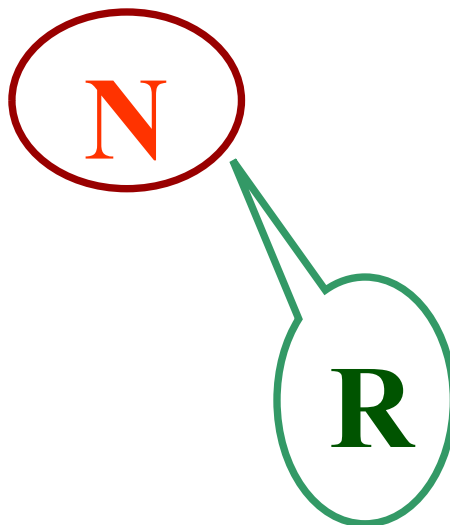
只含根结点



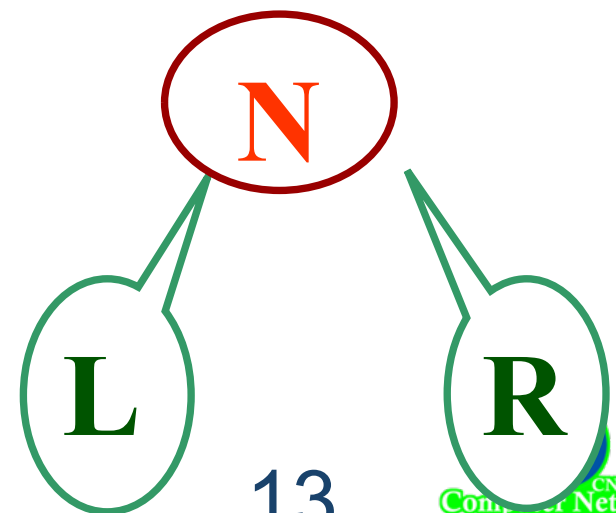
右子树为**空**树



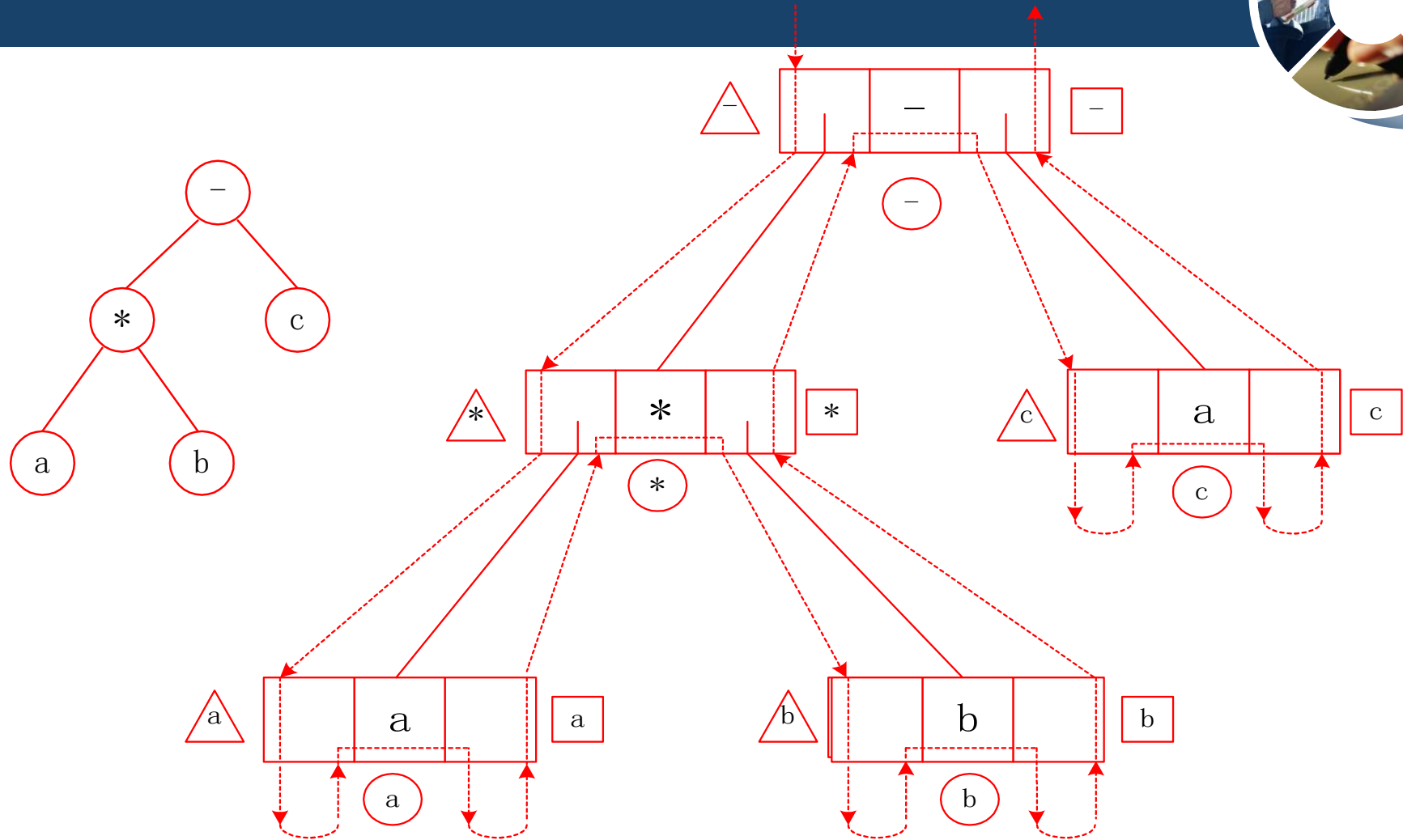
左子树为**空**树



左右子树
均不为空树



二叉树的遍历过程（先根、中根、后根）





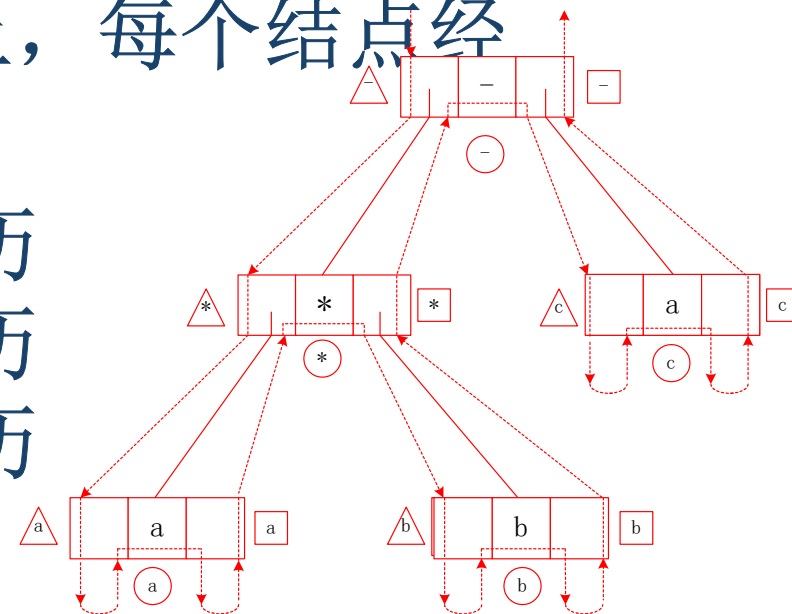
- ❖ 3种遍历算法不同之处仅在于访问根结点、遍历左子树、遍历右子树的先后关系不同。若不考虑Visit()语句，则三种遍历方法完全相同（访问路径是相同的，只是访问结点的时机不同）。

从虚线的出发点到终点的路径上，每个结点经过3次。

第1次经过时访问 = 先序遍历

第2次经过时访问 = 中序遍历

第3次经过时访问 = 后序遍历



- 三种遍历算法均是递归算法：
 - 二叉树的定义本身就是递归的；
 - 递归和栈密切联系：递归过程实际就是对栈的操作过程
 - 可以直接通过对栈的操作，来把递归算法写为非递归算法

二叉树的建立(先序方式)



```
int CreateBiTree(BiTree &T) {  
    scanf(&ch); //cin>>ch;  
    if (ch=='#') T = NULL;  
    else {  
        if (! (T = (BiTNode *)malloc(sizeof(BiTNode))))  
            exit(OVERFLOW);  
        T->data = ch;           // 生成根结点  
        CreateBiTree(T->lchild); // 构造左子树  
        CreateBiTree(T->rchild); // 构造右子树  
    }  
    return OK;  
} // CreateBiTree
```



第6章 测验题



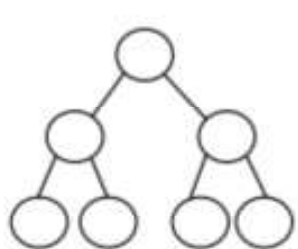
树最适合用来表示（ ）。

- ☐ A 有序数据元素
- ☐ B 无序数据元素
- ☒ C 元素之间具有分支层次关系的数据
- ☐ D 元素之间无联系的数据

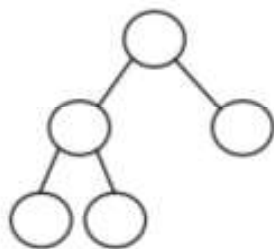
提交



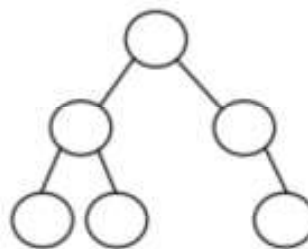
如图所示的4棵二叉树中，（ ）不是完全二叉树。



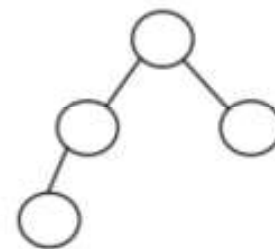
A



B.



C.



D.

☐ A

☐ B

☒ C

☐ D

提交



对于任意一棵二叉树T，其叶子节点个数 n_0 ，度为1的节点个数 n_1 以及度为2的节点个数 n_2 之间的关系为（ ）。

A $n_2 = n_1 + n_0$

B $n_2 = n_0 + 1$

C $n_0 = n_2 + 1$

D $n_0 = n_1 + 1$

提交



假定在一棵二叉树中，双分支结点数为 15，单分支结点数为 30，则叶结点数为（ ）个。

A 15

B 16

C 17

D 47

提交



若一棵完全二叉树有 768 个结点，则该二叉树中叶结点的个数是（ ）。

A 257

B 258

C 384

D 385

提交



二叉树的前序遍历序列中，任意一个结点均处在其子女结点的前面。

☒ A 对

☐ B 错

提交



一棵非空的二叉树的先序遍历序列与后序遍历序列正好相反，则该二叉树一定满足（）。

- ☐ A 所有的结点均无左孩子
- ☐ B 所有的节点均无右孩子
- ☒ C 只有一个叶子结点
- ☐ D 是任意一棵二叉树

提交



二叉树的先序（中序、后序）遍历结果是唯一的。

☒ A 对

☐ B 错

提交



若一棵二叉树的前序遍历序列和后序遍历序列分别是 1,2,3,4 和 4,3,2,1, 则该二叉树的中序遍历序列不可能是 ()。

- ☐ A 1,2,3,4
- ☐ B 2,3,4,1
- ☒ C 3,2,4,1
- ☐ D 4,3,2,1

提交



已知完全二叉树的第 7 层有 10 个叶结点，则整个二叉树的结点数最多是 [填空1] 。

作答



已知完全二叉树的第 8 层有 2 个叶结点，则整个二叉树的结点数最多是 [填空1] 。

作答



在进行哈夫曼编码时，以下哪种逻辑结构能够方便对数据进行组织与管理（ ）。

- ☐ A 线性表
- ☒ B 树
- ☐ C 图
- ☐ D 集合

提交