6.1 树的定义和基本术语

树（Tree）是n（n>=0）个结点的有限集。在任意一棵非空树中:1）有且仅有一个特定的称为根（Root）的结点；2）当n>1时，其余结点可分为m（m>0）个互不相交的有限集T_1，T_2，…，T_m，其中每一个集合本身又是一棵树，并且称为根的子树。

其抽象数据类型定义如下：

ADT Tree{

数据对象 D： D是具有相同特性的数据元素的集合。

数据关系 R：

若D为空集，则称为空树。若D仅含一个数据元素，则R为空集，否则R={H}，H是如下二元关系：

（1）在D中存在唯一的称为根的数据元素root；它在H下无前驱

（2）若D- {root} ≠Ø，则存在D- {root}的一个划分$D_1, D_2, \ldots, D_m$（m>0)，对任意j ≠ k，有$D_j \cap D_k = \varnothing$,且对任意的i，唯一存在数据元素$x_i \in D_i$,有<root，$x_i$>∈H；

(3) 对应于D- {root}的划分，H- {<root，x_1>，... <root，x_m>}有惟一的一个划分$H_1, \ldots H_m$（m>0），对任意j ≠ k，有$H_j \cap H_k = \varnothing$，且对任意的i，$H_i$是$D_i$上的关系，（$D_i$, {H_i})是一棵符合本定义的树，称为根root的子树。

基本操作：

查找类：

Root(T) // 求树的根结点

Value(T, cur_e) // 求当前结点的元素值

Parent(T, cur_e) // 求当前结点的双亲结点

LeftChild(T, cur_e) // 求当前结点的最左孩子

RightSibling(T, cur_e) // 求当前结点的右兄弟

TreeEmpty(T) // 判定树是否为空树

TreeDepth(T) // 求树的深度

TraverseTree(T, Visit()) // 遍历

插入类：

InitTree(&T) // 初始化置空树

CreateTree(&T, definition) // 按定义构造树

Assign(T, cur_e, value) // 给当前结点赋值

InsertChild(&T, &p, i, c)
// 将以c为根的树插入为结点p的第i棵子树

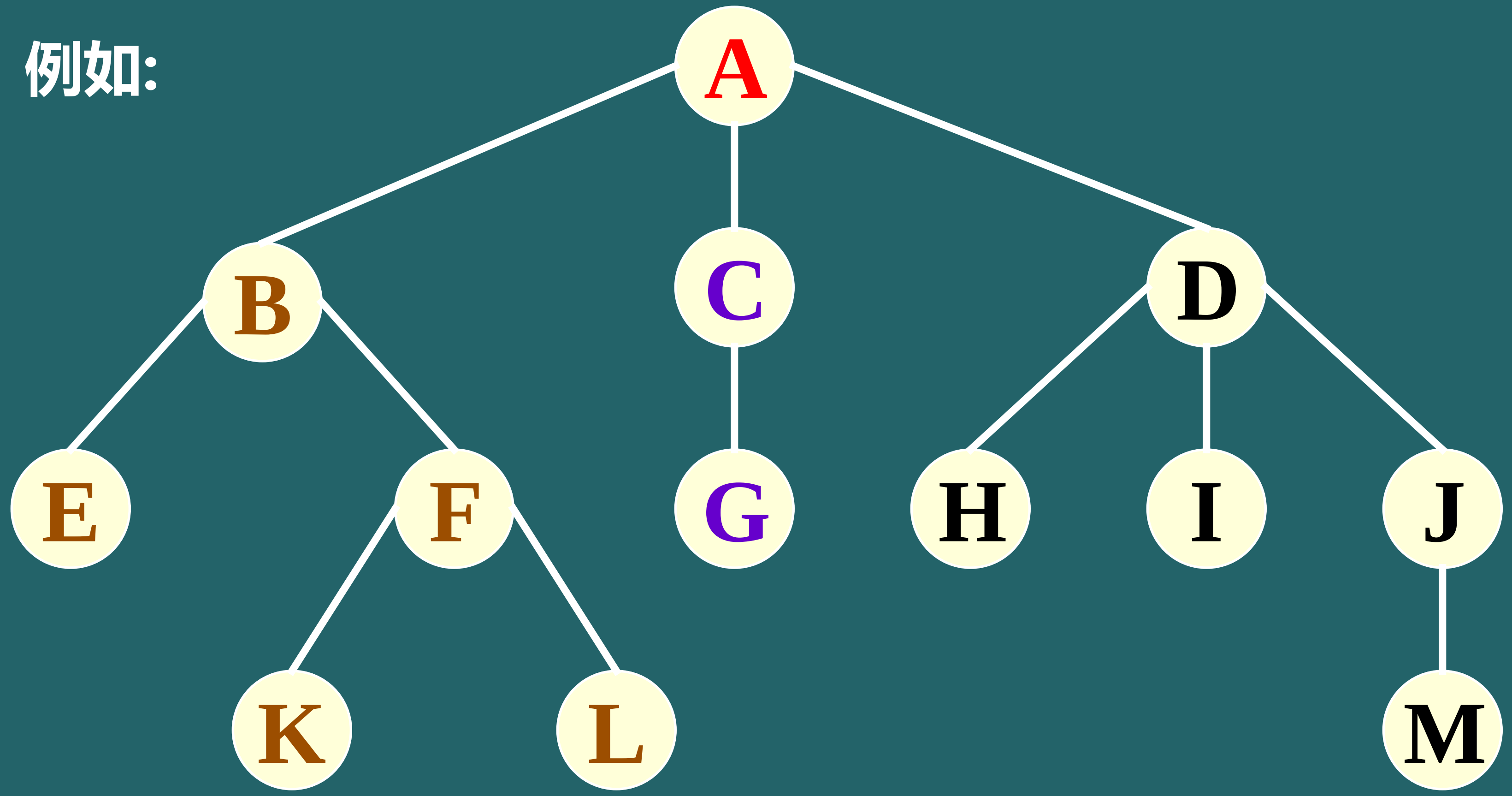
删除类：

ClearTree(&T) // 将树清空

DestroyTree(&T) // 销毁树的结构

DeleteChild(&T, &p, i)
    // 删除结点p的第i棵子树

例如:



$$\underset{\text{树根}}{A}\left(\underbrace{B(E, F(K, L))}_{T_1}, \underbrace{C(G)}_{T_2}, \underbrace{D(H, I, J(M))}_{T_3} \right)$$

有向树：

（1）有确定的根；

（2）树根和子树根之间为有向关系。

有序树：

子树之间存在确定的次序关系。

无序树：

子树之间不存在确定的次序关系。

基本术语

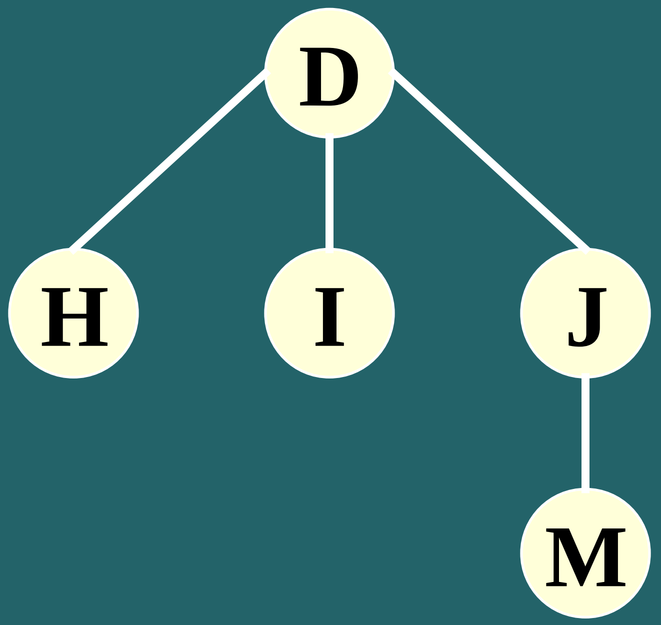
结点: 数据元素及若干指向其子树的分支

结点的度: 结点拥有的子树的数目

树的度: 树中所有结点的度的最大值

叶子结点: 度为零的结点

分支结点: 度不为零的结点



(从根到结点的)**路径**：

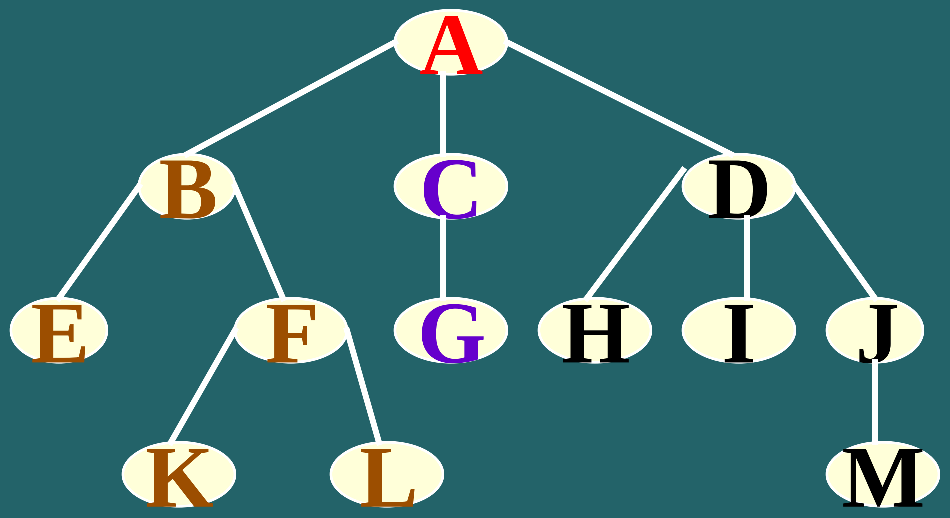
由从**根**到该结点所经分支和结点构成

例如：

孩子结点、**双亲**结点

兄弟结点、堂兄弟

祖先结点、**子孙**结点



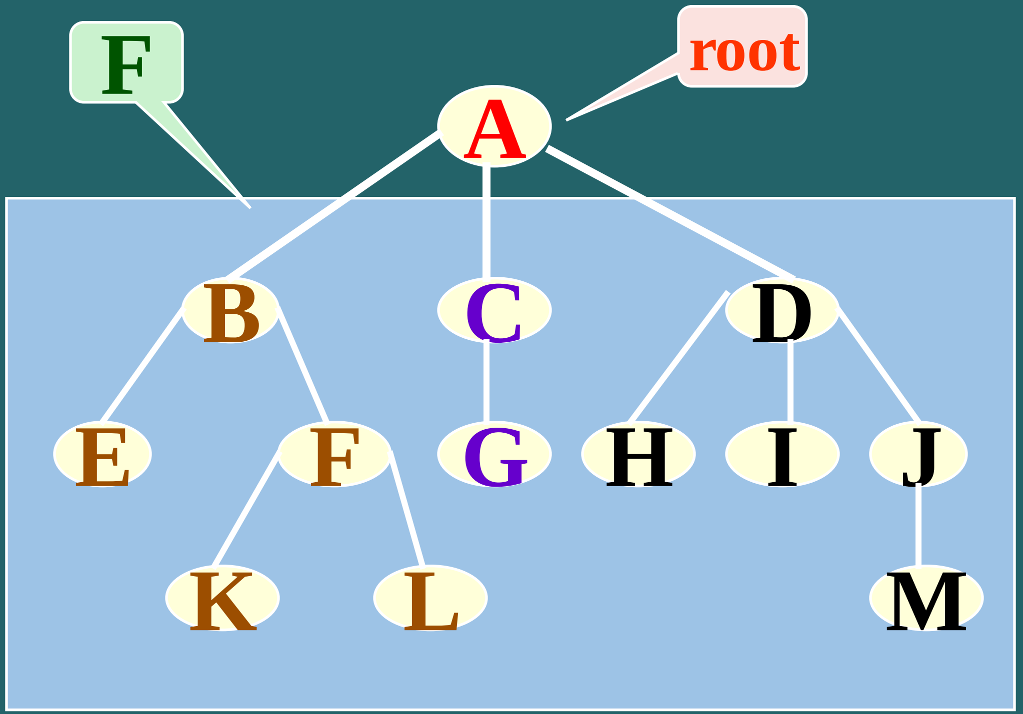
结点的层次: 假设根结点的层次为1，根的孩子为第二层。

第l 层的结点的子树根结点的层次为l+1

树的深度(高度)： 树中叶子结点所在的最大层次

森林：

是m（m≥0）棵互不相交的树的集合。



任何一棵非空树是一个二元组

Tree =（root，F）

其中：root 被称为根结点

F 被称为子树森林