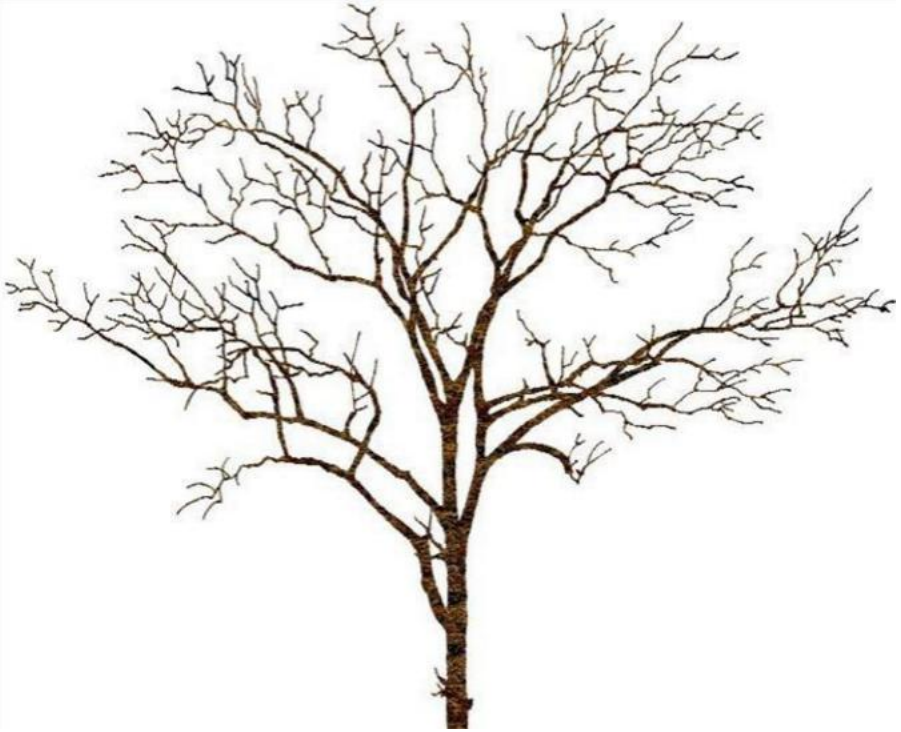
数据结构
Data Structure

2017年秋季学期

刘鹏远

树



树型结构是一类非常重要的**非线性结构**。一般来说，树型结构是**以分支(一对多)关系定义的层次结构**。

树在计算机领域中有着广泛的应用，例如在编译程序中，用树来表示源程序的语法结构；在数据库系统中，可用树来组织信息；在分析算法的行为时，可用树来描述其执行过程等等。

很多问题也可以转化为树的结构来求解与应用，如：

- ✓ 人机对弈（阿法狗AlphaGo）
- ✓ 字符编码
- ✓ 动态查找等

- 树相关基本概念
- 二叉树与树
- 二叉树遍历
- 应用
- 二叉树的计数(选修)
- 线索二叉树(选修)
- 树的遍历(选修)

树的基本概念

1 树的定义

树(Tree)是n(n≥0)个结点的有限集合T，若n=0时称为空树，否则：

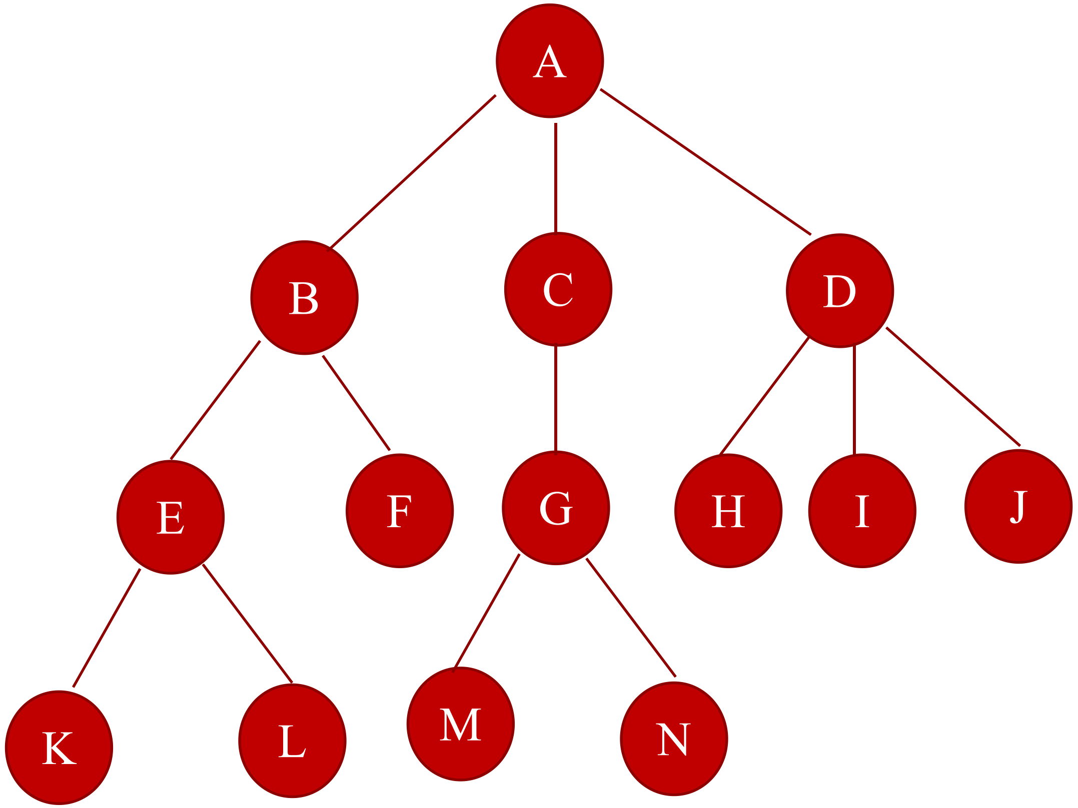
(1) 有且只有一个特殊的称为树的根(Root)结点；

(2) 若n>1时，其余的结点被分为m(m>0)个互不相交的子集T_1, T_2, T_3…T_m，其中每个子集本身又是一棵树，称其为根的子树(Subtree)。

这是树的递归定义，而只有一个结点的树必定仅由根组成，如图所示。思考：1、元素关系在定义里何处体现？2、为何允许n=0，定义空树？

树的表示

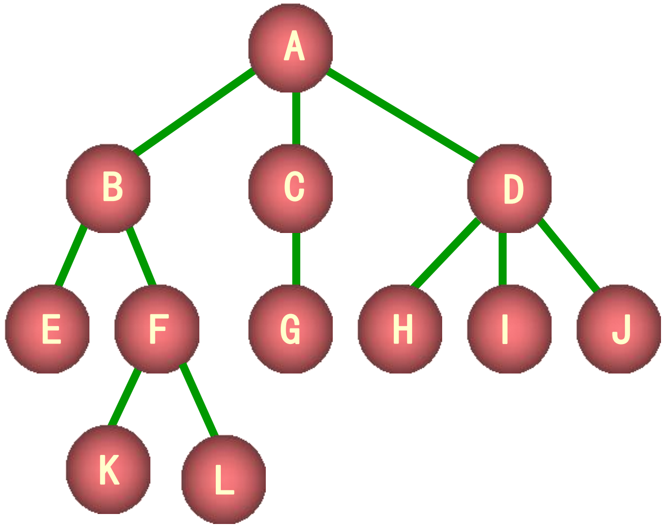
(a) 只有根结点

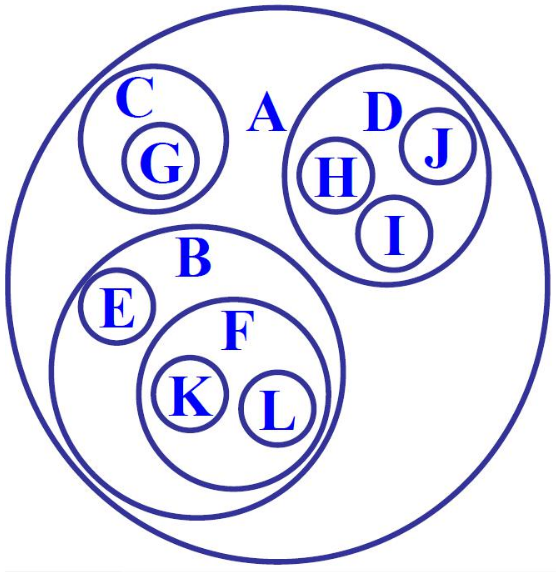


(b) 一般的树

➢树的其他表示

嵌套集合、广义表表示、凹入表示






```
A*****************
 B****************
  E***************
  F***************
   K**************
   L**************
 C****************
  G***************
 D****************
  H***************
  I***************
  J***************
```

{A,{B,{E, F,{K,L}}, C,{G}, D,{H, I, J}}}

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

基本术语

(1) 结点(node):

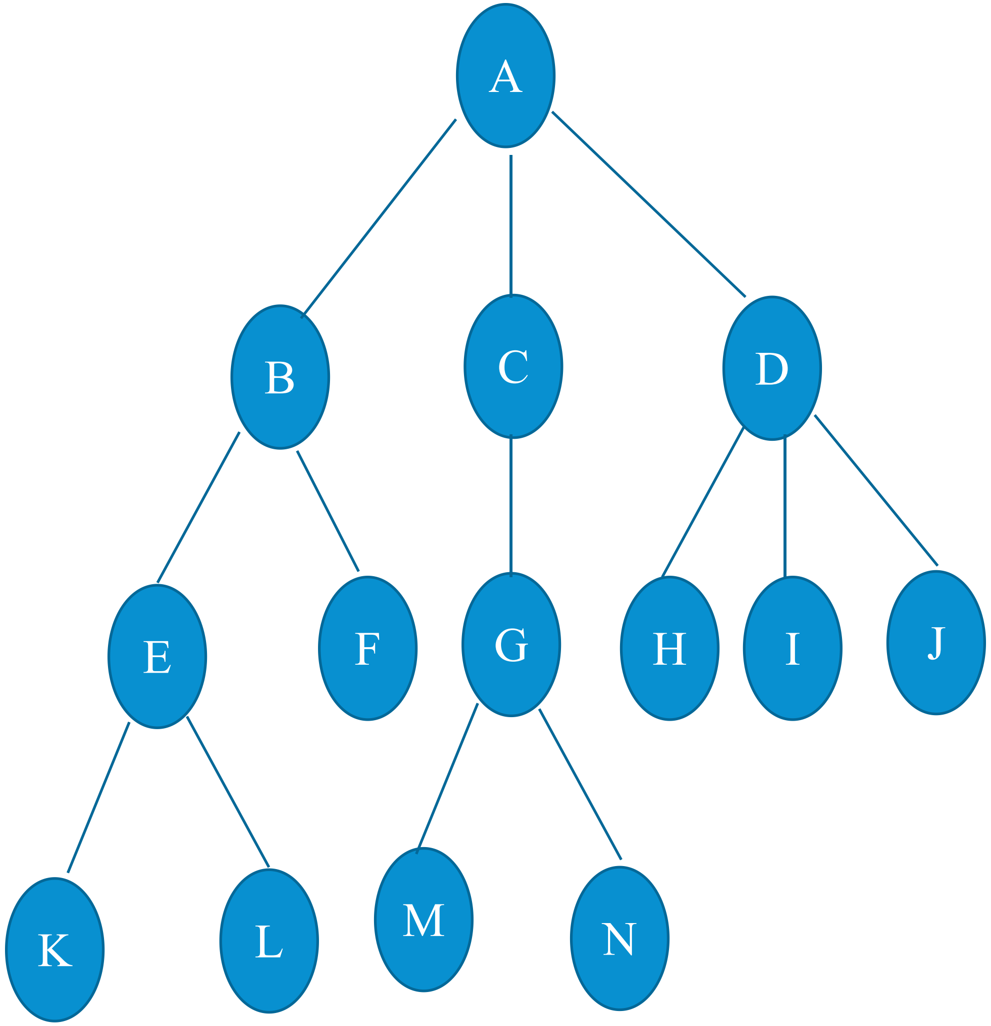
一个数据元素及其若干指向其子树的分支。

(2) 结点的度(degree)、树的度:

结点所拥有的子树的棵数称为结点的度。

树中结点度的最大值称为树的度。

(3) 叶子结点、非叶子结点:树中度为0的结点称为叶子结点(终端结点)。相对应地,度不为0的结点称为非叶子结点(非终端结点/分支结点)。除根结点外,分支结点又称为内部结点。



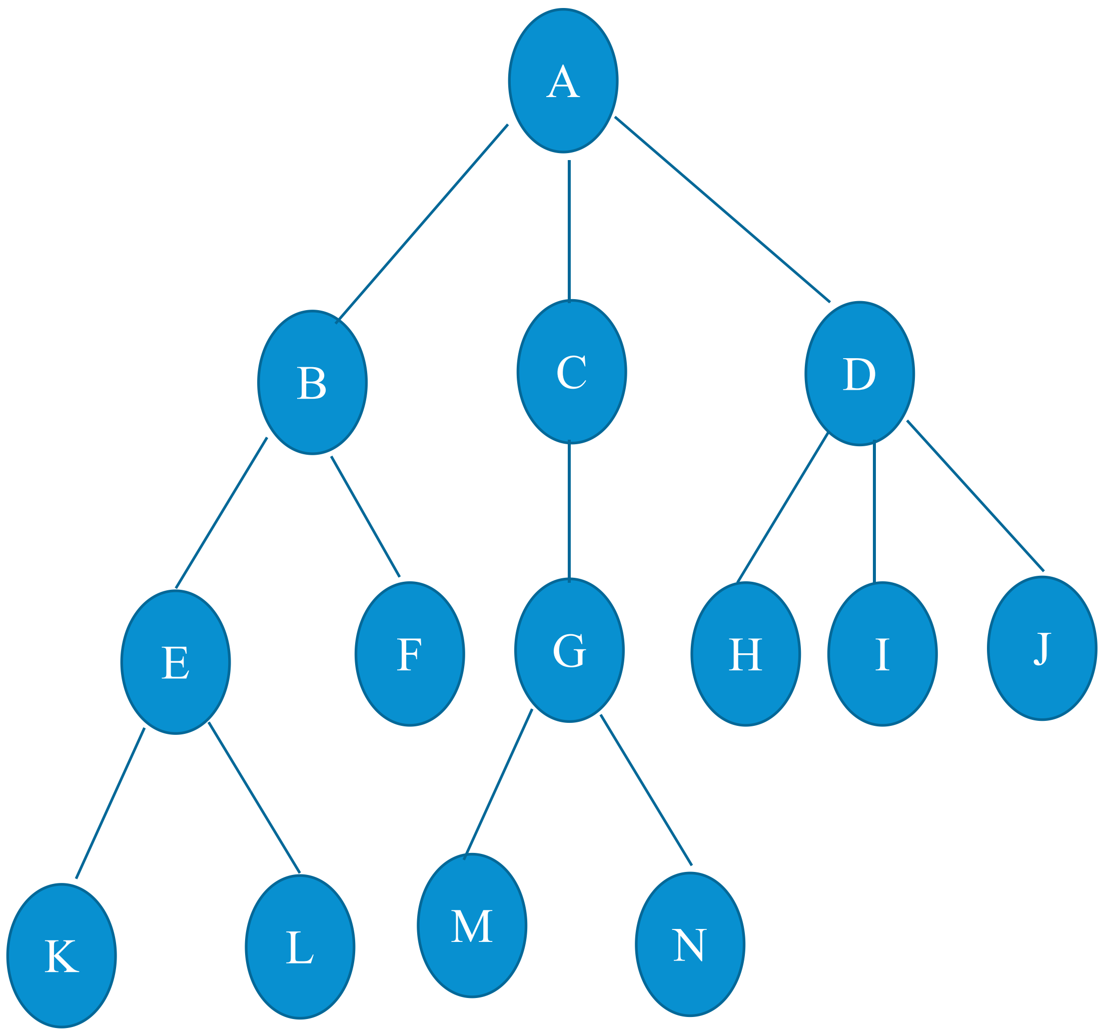
(b) 一般的树

图中结点H、I、J、K、L、M、N是叶子结点，而所有其它结点都是分支结点。

(4) 孩子结点、双亲结点、兄弟结点

一个结点的子树的根称为该结点的孩子结点(child)或子结点；相应地，该结点是其孩子结点的双亲结点(parent)或父结点。同**一双亲结点的所有子结点**互称为兄弟结点。

如图中结点B 、C、D是兄弟结点；结点E 、F是兄弟结点。



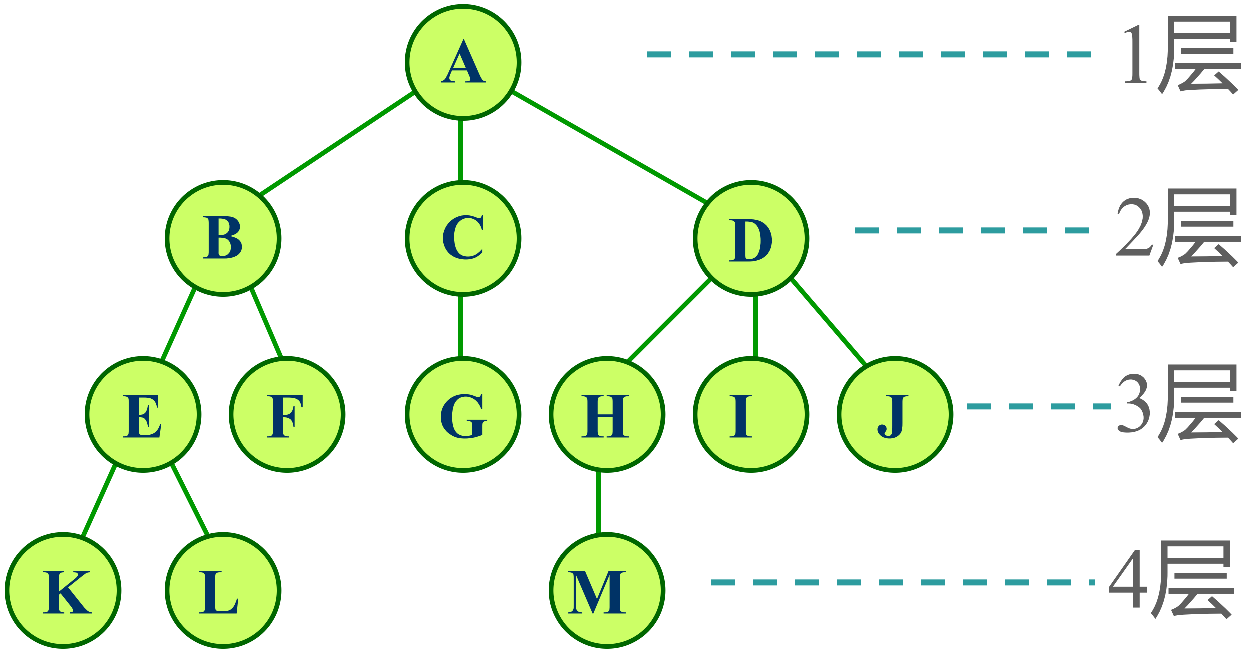
(b) 一般的树

(5) 层次、堂兄弟结点

规定树中根结点的层次为1，其余结点的层次等于其双亲结点的层次加1。树中节点最大层次称为数的深度或高度

若某结点在第 l (l≧1) 层，则其子结点在第 l+1层。

双亲结点在同一层上的所有结点互称为堂兄弟结点。如图中结点G与E、F、H、I、J。

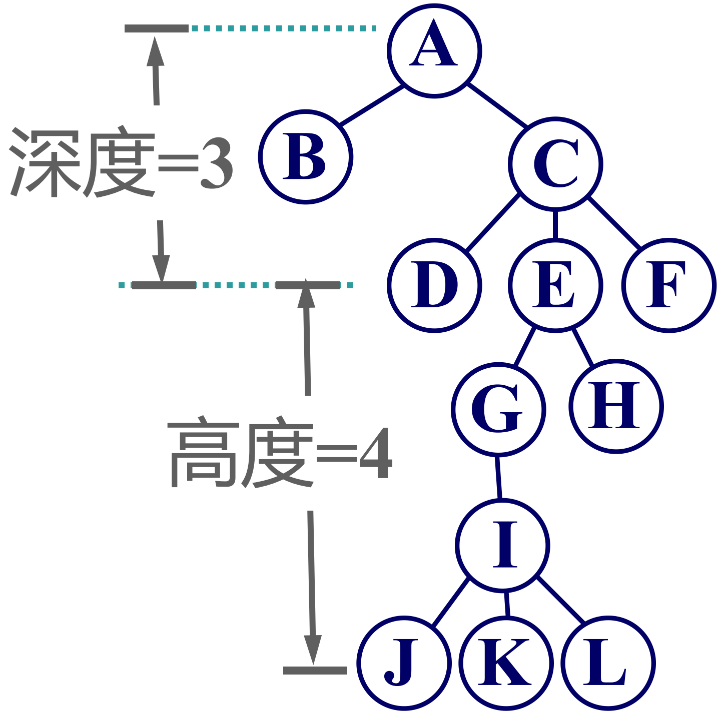


树高度与深度

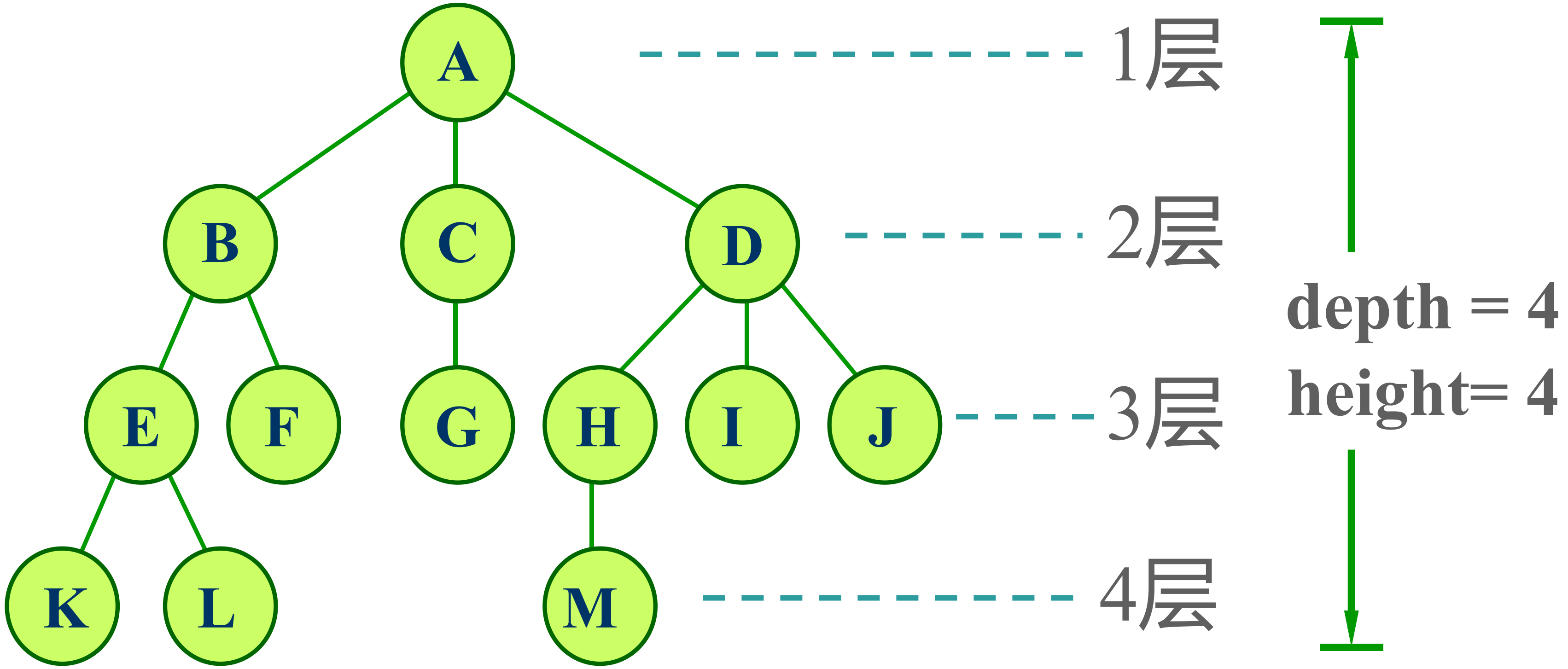
结点的深度和结点的高度是不同的。结点的深度即结点所处层次，是从根向下逐层计算的；（其实是树根）

结点的高度是从下向上逐层计算的：叶结点的高度为1，其他结点的高度是取它的所有孩子结点最大高度加一。

树的深度与高度相等。



树是分层结构，又是递归结构。除第一个根节点外，所有子树的根结点有且仅有一个直接前驱，但可以有 0 个或多个直接后继。考虑N个节点的树，其边的个数为？



有序无序树与森林

如将树中节点看成从左到右有次序的，即不能互换，则称该树为有序树，否则称为无序树。

有序树最左边子树的根称为第一个孩子，最右边的称为最后一个孩子。

森林（Forest）是m棵互不相交的树的集合。(m>=0)
树的每个节点的子树集合也是森林，仍旧可以递归定义。
还可以对子树进行编号。

ADT与操作

ADT见教材P118页，操作见教材119页

destory	删
deletechild	删
createtree	建
parent	查
leftchild	查
rightchild	查
insertchild	增
traversetree	遍历

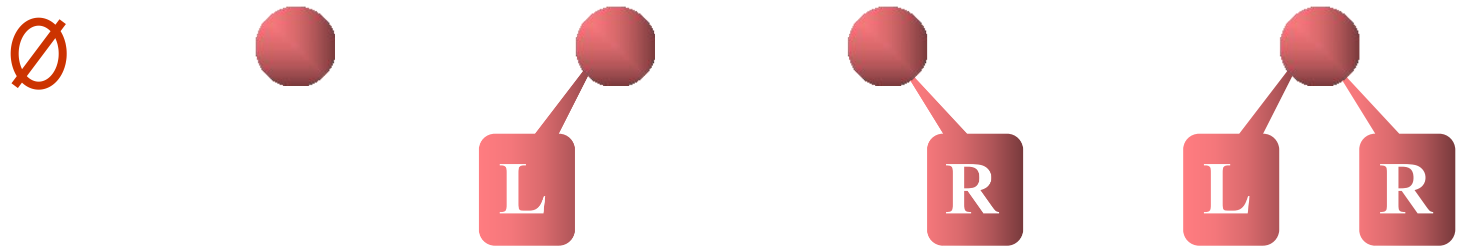
由简入繁，首先考虑度为二的树。
下面引入二叉树。

二叉树 (Binary Tree)

二叉树的定义

一棵二叉树是结点的一个有限集合，该集合或者为空，或者是由一个根结点加上两棵分别称为左子树和右子树的、互不相交的二叉树组成。

这个定义是递归的。



二叉树的五种不同形态

送分题（去年）：3个节点可能组成的2叉树的形态？

二叉树的性质1

若二叉树的层次从1开始，则在二叉树的第 i 层最多有 2^{i-1} 个结点。（ i≥1）

[证明]（这还用证明?^-^）

用数学归纳法

- $i = 1$时，根结点只有1个，$2^{1-1} = 2^0 = 1$；
- 若设 $i = k$ 时性质成立，即该层最多有 2^{k-1} 个结点，则当 $i = k+1$ 时，由于第 k 层每个结点最多可有 2 个孩子，第 $k+1$ 层最多结点个数可有 $2*2^{k-1} = 2^k$ 个，故性质成立。

二叉树的性质2

高度为 h 的二叉树最多有 2^h-1个结点。(h≥0)

[证明](另一种用数学归纳法的证明，课下自证)

用求等比级数前k项和的公式

高度为 h 的二叉树有 h 层，各层最多结点个数相加，得到等比级数，求和得：

$$2^0 + 2^1 + 2^2 + \ldots + 2^{h-1} = 1*(1-2^h)/1-2 = 2^h-1$$

空树的高度为 0，只有根结点的树的高度为 1。

注意树每层最多节点数与总最多节点数公式不要弄混

二叉树的性质2

深度	该层最多节点数	二叉树的最多节点总数
1	$2^0=1$	1
2	$2^1=2$	3
3	$2^2=4$	7
4	$2^3=8$	15
5	$2^4=16$	31
6	$2^5=32$	63
……	……	……

对任何一棵二叉树, 如果其叶结点有 n_0 个, 度为2的非叶结点有 n_2 个，则有 $n_0 = n_2 + 1$

证明1：

若设度为 1 的结点有 n_1 个，总结点个数为 n，总边数为 e，则根据二叉树的定义，$n = n_0+n_1+n_2$ $e = n-1= 2n_2+n_1$

$n_2 = n_0-1$ $n_0 = n_2+1$

证明2：用数学归纳法

- 推论：叶子节点总数每增加一个，则度为2的根节点总数必增1，反之也成立 思考：n1可以无限制吗？
- 送分题：可用于判断二叉树各类结点个数。

满二叉树与完全二叉树

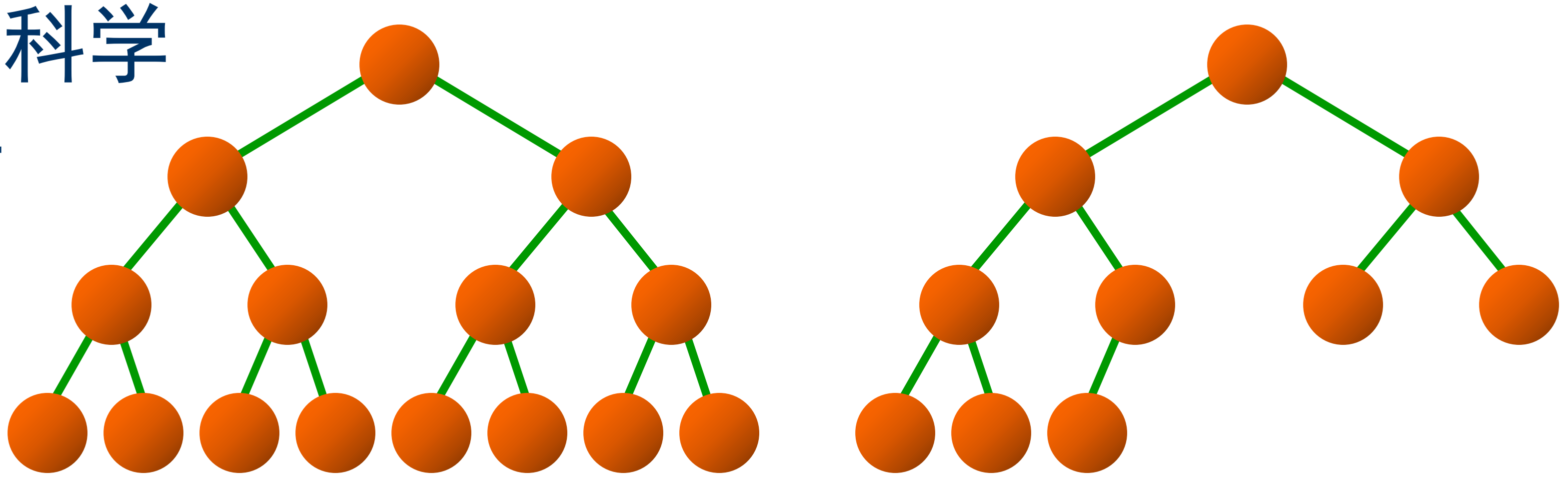
定义1 满二叉树 (Full Binary Tree) 深度为k，节点数为2^k-1的树

定义2 完全二叉树 (Complete Binary Tree)-定义多种

若设二叉树的高度为 h，则共有 h 层。除第 h 层外，其它各层 (1～h–1) 的结点数都达到最大个数，第 h 层从右向左连续缺若干结点，这就是完全二叉树。(已有节点与同层数的满二叉树一一连续对应的二叉树)

满 > 完全　　不够科学

翻译成"完备"较好



对于深度为k的完全二叉树，性质：

1） 前k-1层为满二叉树；
2） 第k层结点依次占据最左边的位置；
3） 一个结点有右孩子，则它必有左孩子；
4） 度为1的结点个数为0或1
5）叶子结点只可能在层次最大的两层上出现；（思考：叶子结点只在层次最大的两层上出现是否一定是完全二叉树？）
6）对任一结点，若其右分支下的子孙的最大层次为 l，则其左分支下的子孙的最大层次必为 l或 l+1。

二叉树的性质4

具有 n（$n \geq 0$）个结点的完全二叉树的高度为$\lceil \log_2(n+1) \rceil$

证明：设完全二叉树的高度为 h，则有

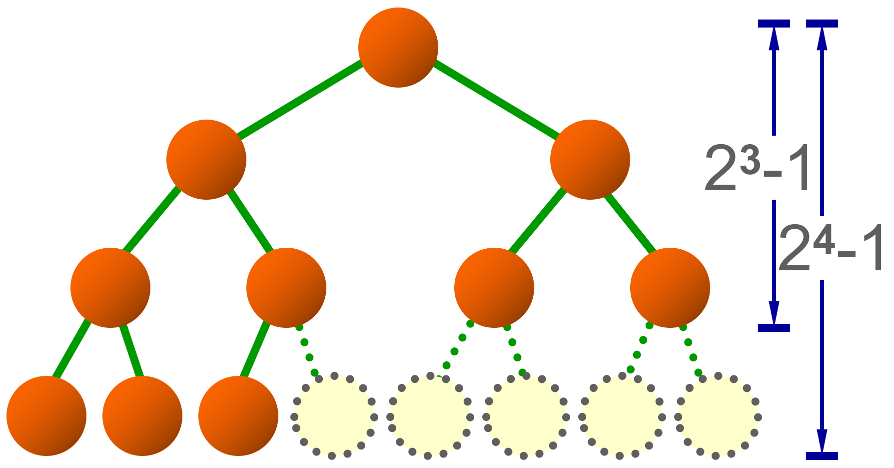
$$\underbrace{2^{h-1}-1}_{\text{上面}h-1\text{层结点数}} < n \leqslant \underbrace{2^h-1}_{\text{包括第}h\text{层的最大结点数}}$$

变形　　$2^{h-1} < n+1 \leqslant 2^h$

取对数

$$h-1 < \log_2(n+1) \leqslant h$$

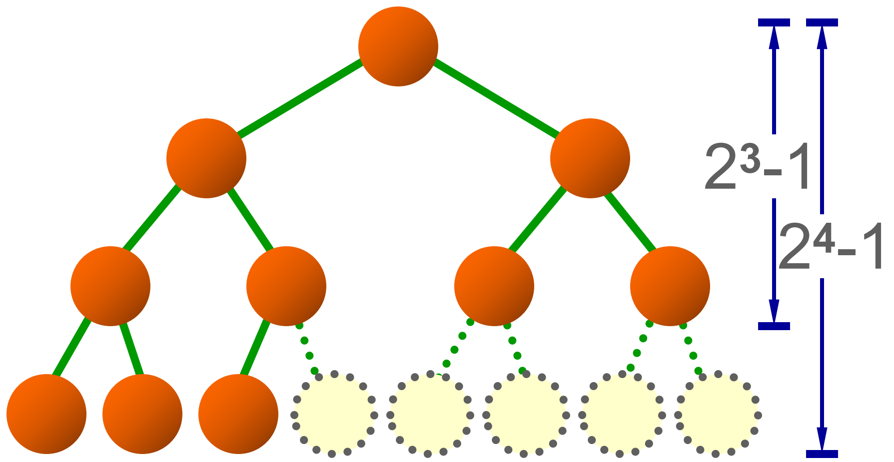
有　$h = \lceil \log_2(n+1) \rceil$



注意：求高度的另一公式为 $\lfloor \log_2 n \rfloor + 1$，此公式对于 $n = 0$ 不适用。

若设完全二叉树中叶结点有 n_0 个，则该二叉树总的结点数为 $n = 2n_0$，或 $n = 2n_0 - 1$。

若完全二叉树的结点数为奇数，没有度为1的结点；为偶数，有一个度为1的结点。

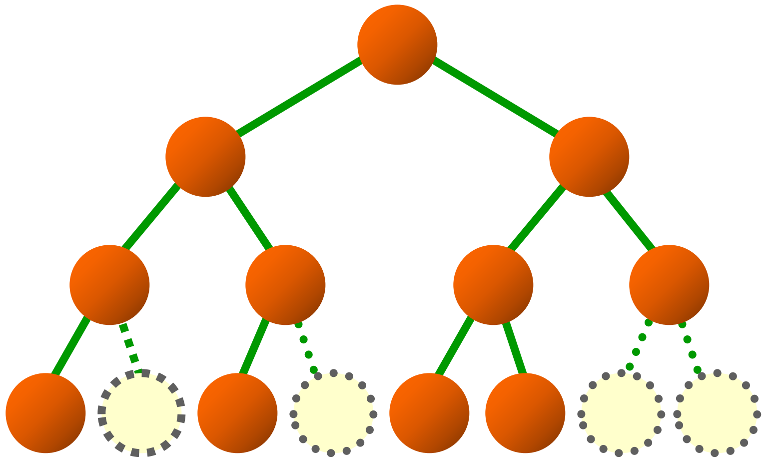


二叉平衡树：

h-1层都是满的，第h 层结点分布在各处。

任意两个叶节点的深度之差不超过1，即任意节点左右子树深度之差绝对值不超过1

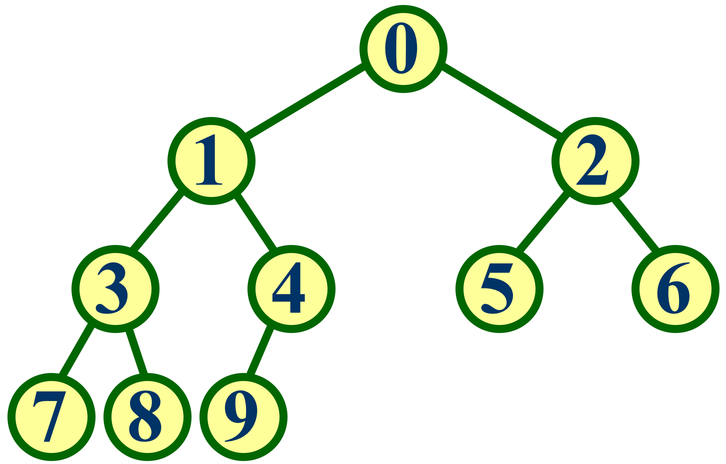
问题：完全二叉树是不是二叉平衡树？



二叉树的性质5

如将一棵有 n 个结点的**完全二叉树**自顶向下，同一层自左向右连续给结点编号: 0, 1, 2, …, n-1，则有以下关系：

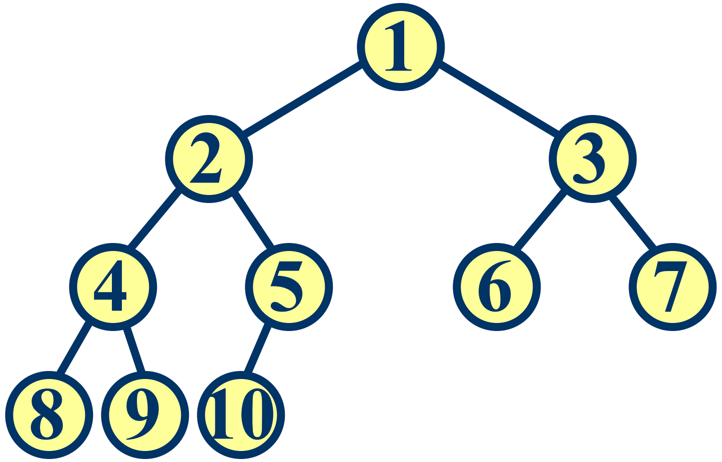
- 若$i = 0$, 则 i 无双亲；
  若$i > 0$, 则 i 的双亲为$\lfloor (i-1)/2 \rfloor$。
- 若$2*i+1 < n$, 则 i 的左孩子为 $2*i+1$；
  若$2*i+2 < n$, 则 i 的右孩子为$2*i+2$。
- 若 i 为偶数, 且i != 0, 则
  其左兄弟为i-1； 偶数--右
  若 i 为奇数, 且i != n-1,
  则其右兄弟为i+1。 奇数--左



二叉树的性质5

如果**完全二叉树**各层次结点从 1 开始编号：1, 2, 3, …, n,
则有以下关系：

- 若$i = 1$, 则 i 无双亲；
  若$i > 1$, 则 i 的双亲为$\lfloor i / 2 \rfloor$。
- 若$2*i <= n$, 则 i 的左孩子为 $2*i$；
  若$2*i+1 <= n$, 则 i 的右孩子为$2*i+1$
- 若 i 为奇数, 且i != 1,
  则其左兄弟为i-1；
  若 i 为偶数, 且i != n,
  则其右兄弟为i+1。



二叉树的性质，简单，但比较重要

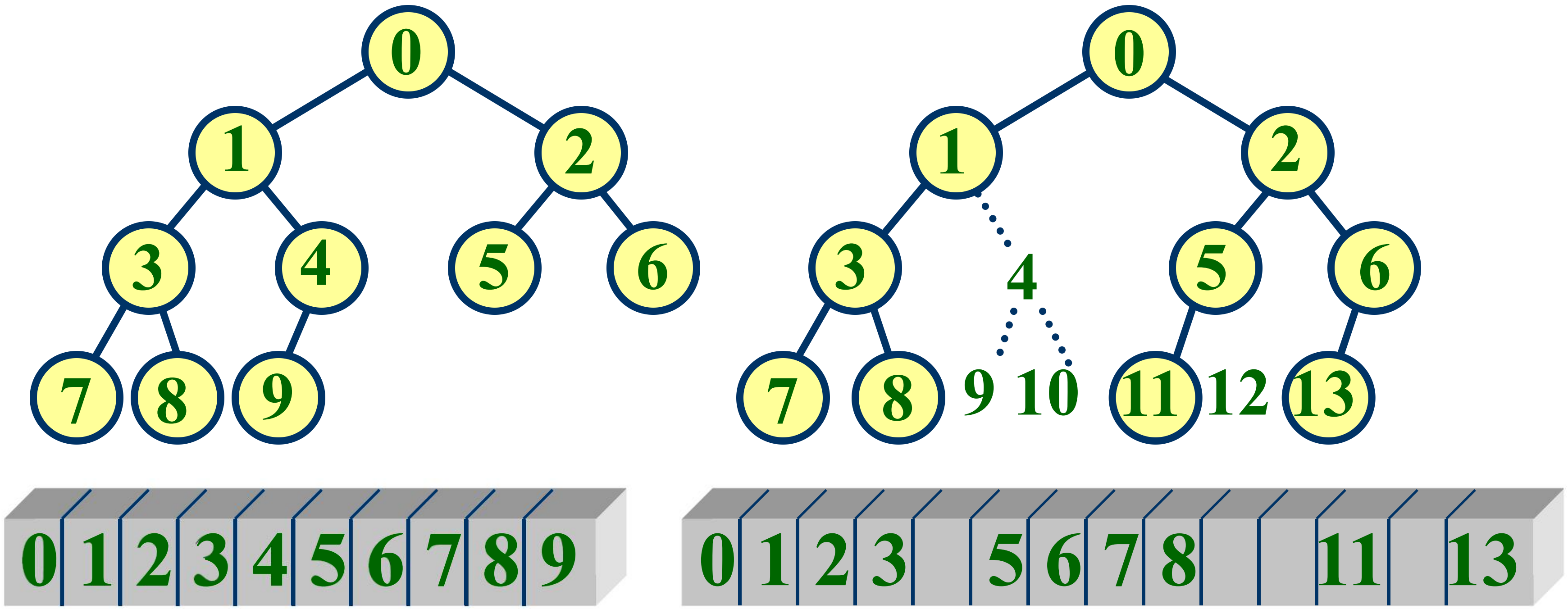
基本是送分题，分数不多，但啥考试都考！

性质较多，记不住，如何破？

性质从哪里来？各类性质，基本就是数数。
记住几个基本概念，其余的画出对应的二叉树，数数即可。

二叉树的存储表示

二叉树的顺序表示



完全二叉树　　一般二叉树

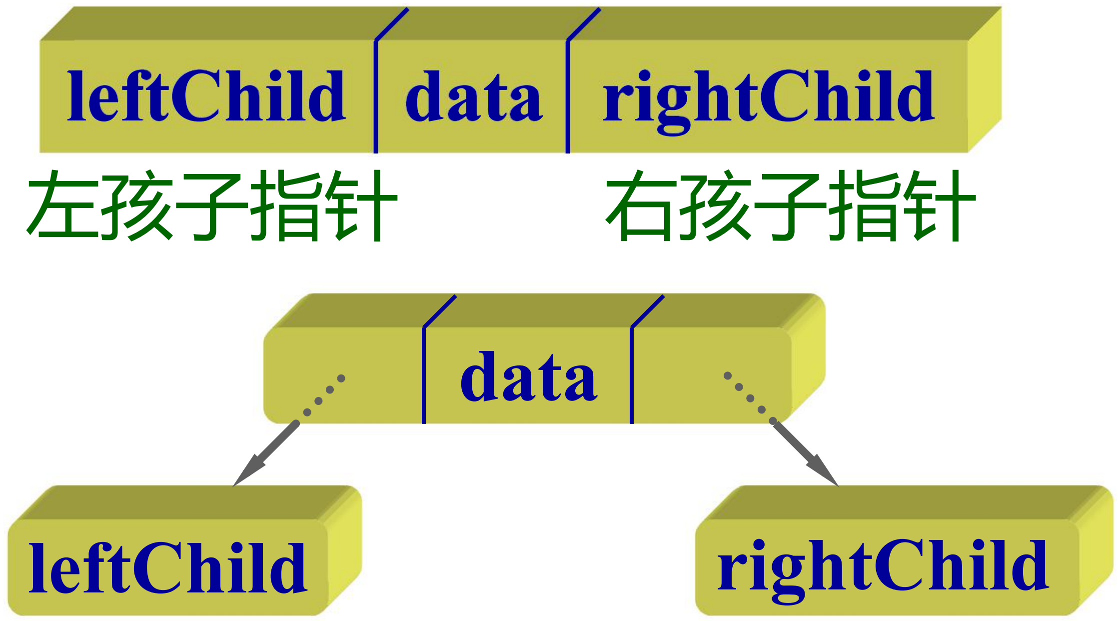
优点？缺点？

极端情形：只有右单支的二叉树

- 对于完全二叉树，因结点编号连续，数据存储密集，适于用顺序表示；
- 斜树--->退化成线性结构（计算）
- 因此，一般的二叉树用链式表示

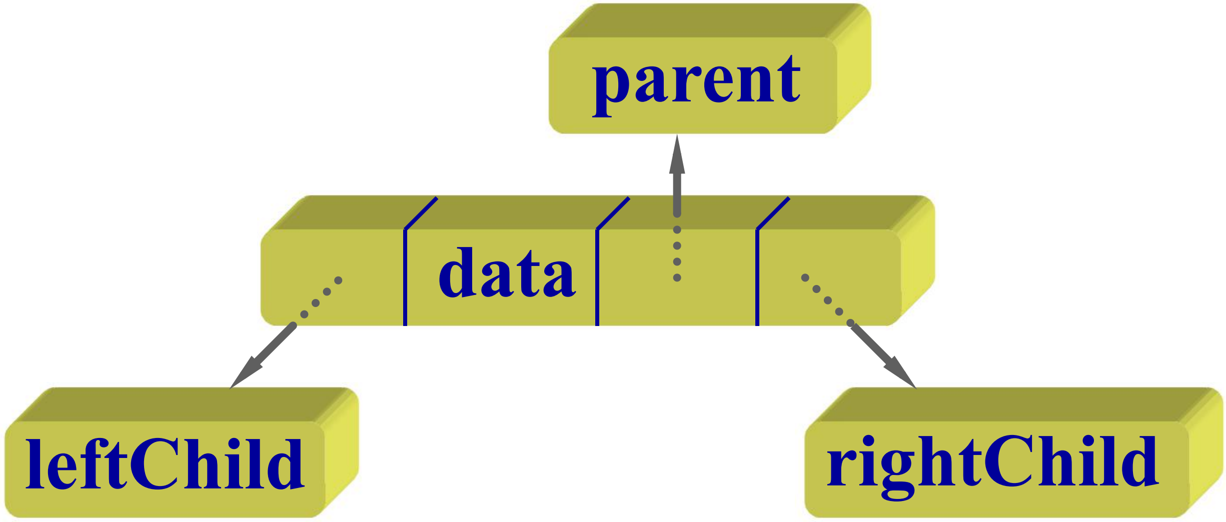
二叉树的二叉链表表示

使用二叉链表，找孩子的时间复杂度为O(1)，找双亲的时间复杂度最坏为O(i)，其中，i 是该结点编号。

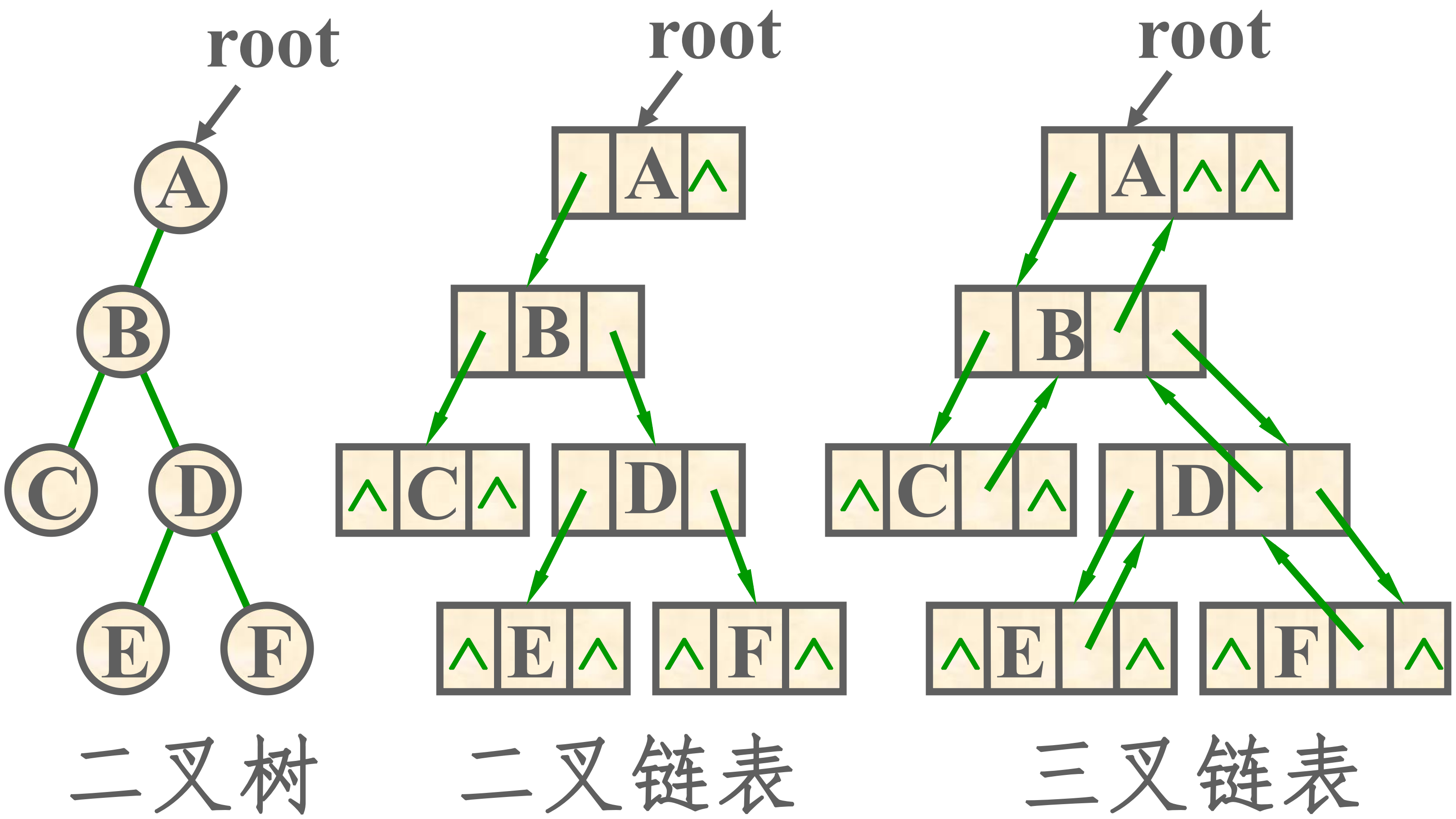


二叉树的三叉链表表示

使用三叉链表，找孩子、双亲的时间都是O(1)。



二叉树链表表示的示例



树的存储表示

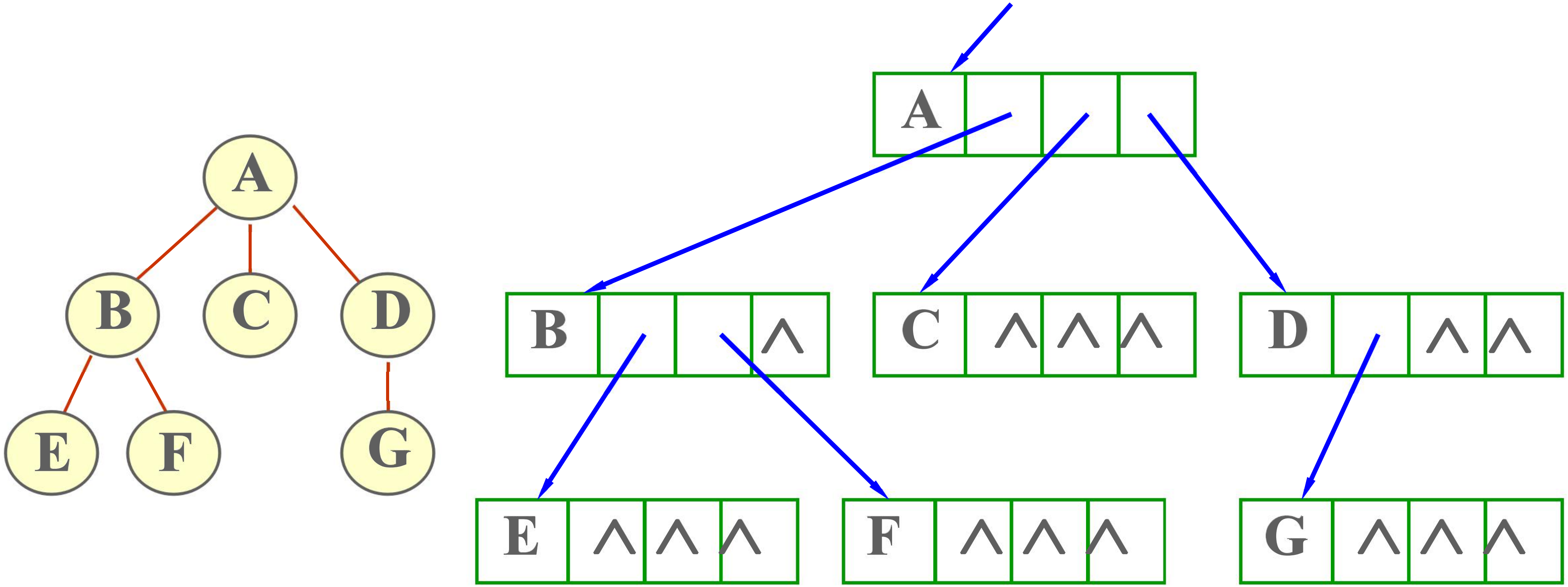
树的存储表示之**孩子指针表示**

仿照二叉树链式存储，树的链式存储的一个合理的想法是：在结点中存放指向每一个孩子结点的指针。

问题：由于各个结点的孩子数不同，每个结点设置数目不等的指针，将很难管理。

方案：设置等长的结点，每个结点包含的指针个数相等，等于树的度（degree）。

问题：这保证结点有足够的指针指向它的所有孩子结点。但可能产生很多空闲指针，造成存储浪费。

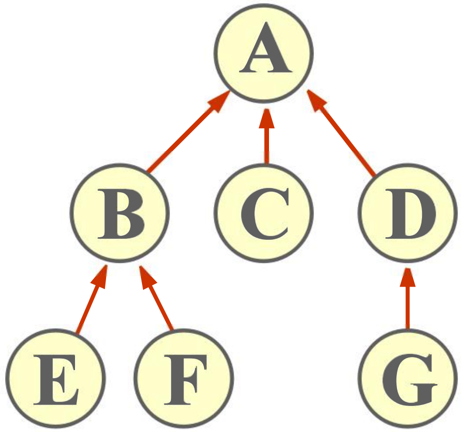


等数量的链域

空链域$2n+1$个

data	child_1	child_2	child_3	…….	child_d

树的存储表示之**双亲表示**



	0	1	2	3	4	5	6
data	A	B	C	D	E	F	G
parent	−1	0	0	0	1	1	3

（并查集应用也利用这种表示）

思考：是何种存储方式？

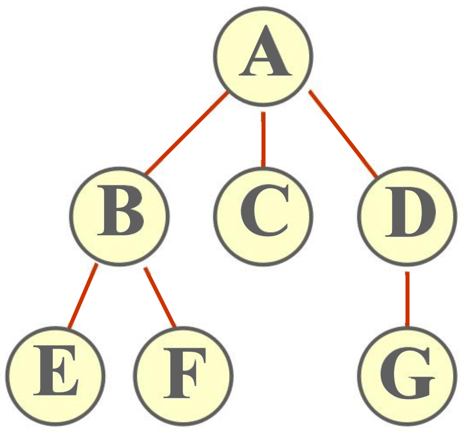
为了操作实现的方便，有时会规定结点的存放顺序。例如，可以按树的层次次序安排所有结点。

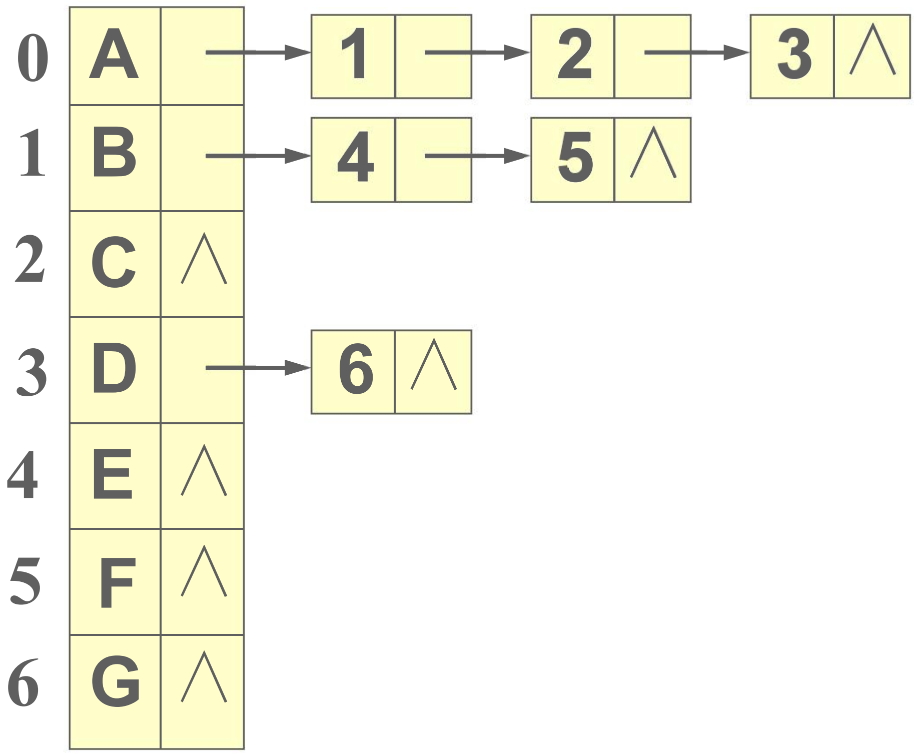
缺点：求某节点的孩子需要遍历整个结构(找该节点及内容)

优点？

树的存储表示---孩子链表表示

- 无序树情形链表中各结点顺序任意，有序树必须自左向右链接各个孩子结点。





- 类似图的邻接表表示
- 缺点：找双亲耗时
- 优点？

扩展：带双亲的孩子链表(P136)

树的存储表示---孩子-兄弟表示

这个比较有创意。也称为树的二叉树表示。结点构造为：

data	firstChild	nextSibling

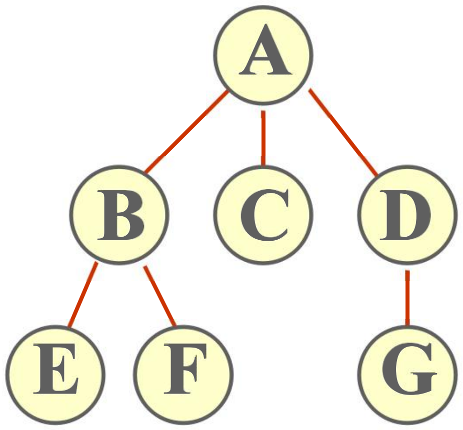
firstChild 指向该结点的第一个孩子结点。无序树时，可任意指定一个结点为第一个孩子。

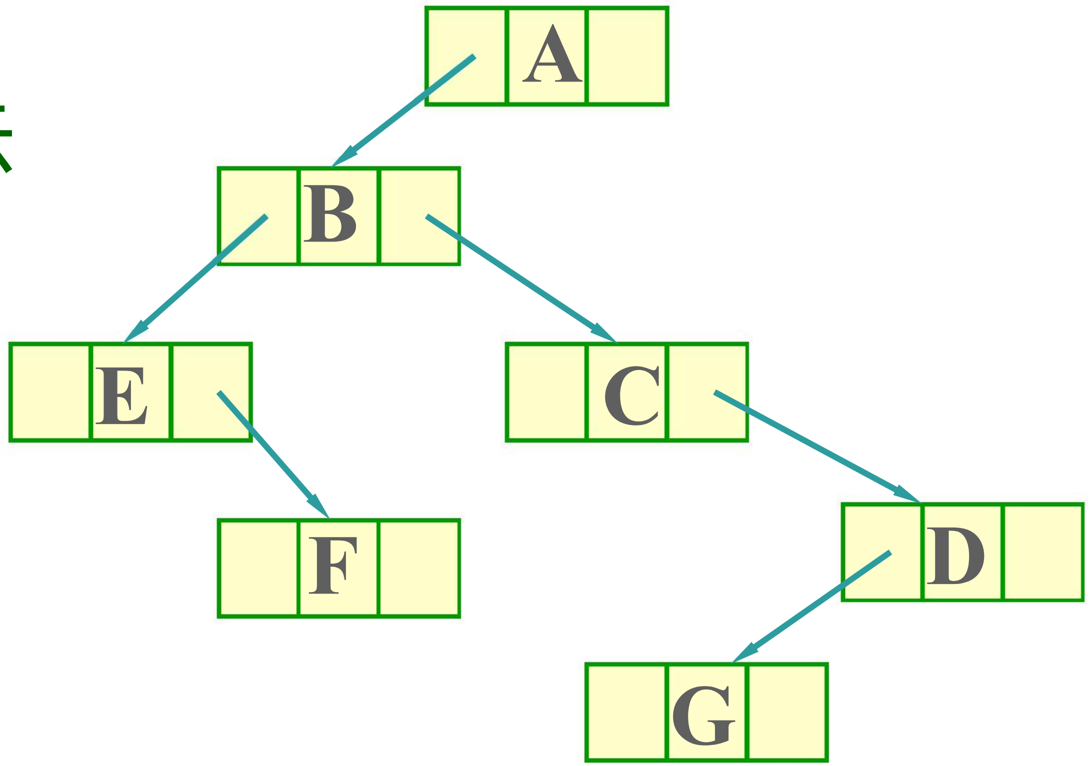
nextSibling 指向该结点的下一个兄弟。任一结点在存储时总是有顺序的。

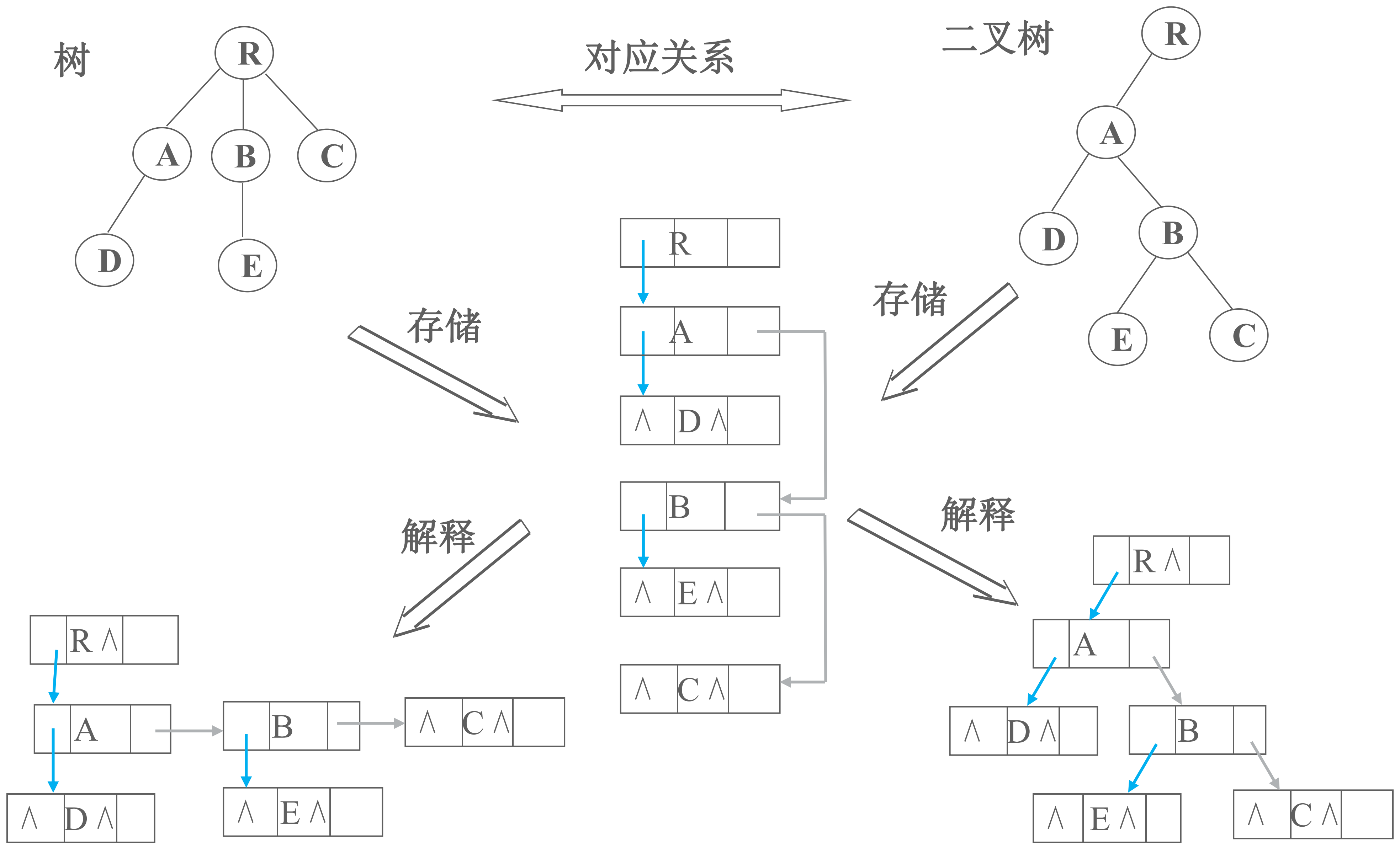
若想找某结点的所有孩子，可先找firstChild,再反复用 nextSibling 沿链扫描。

data	firstChild	nextSibling

树的左孩子－
右兄弟表示







树与二叉树的对应关系

二叉树与树之间的转换

对于一般的树，可以方便地转换成一棵唯一的二叉树与之对应。将树转换成二叉树在“孩子兄弟表示法”中已给出，其详细步骤是：

(1) **加虚线**。在树的每层按从“左至右”的顺序在兄弟结点之间加虚线相连。

(2) **去连线**。除最左的第一个子结点外，父结点与所有其它子结点的连线都去掉。

(3) **旋转**。将树顺时针旋转45^0，原有的实线均左斜。

(4) **整型**。将旋转后树中的所有虚线改为实线，并均向右斜。

由前可知，树与二叉树直接可以一一对应

任意一棵树，利用孩子-兄弟表示，均可以转化为一棵二叉树！森林也可如此（见P138）

所有的树都可视为二叉树

所有的颜色都可分为两种：X色与非X色

所有的分类问题都可视为二分类问题

一切可归为0/非0，从此世界清静了许多.......

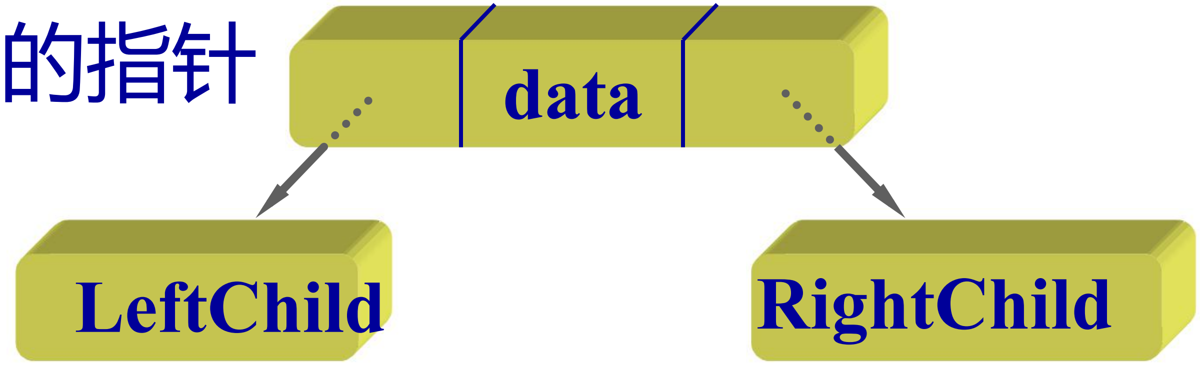
数据结构考试可分为过/不过...^-^

二叉树二叉链表表示与定义

```
typedef struct Node {              //树结点定义
    ElemType data;                 //结点数据域
    struct Node *LeftChild, *RightChild;  //孩子指针域
} BTNode, *BinTree;
```

//树定义

//与链表类似，用指向树根的指针

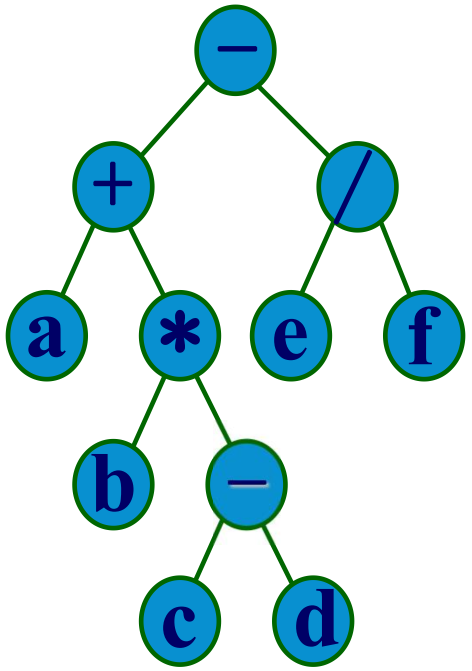


二叉树遍历

遍历就是按某种次序访问树中的结点，要求每个结点仅被访问一次。

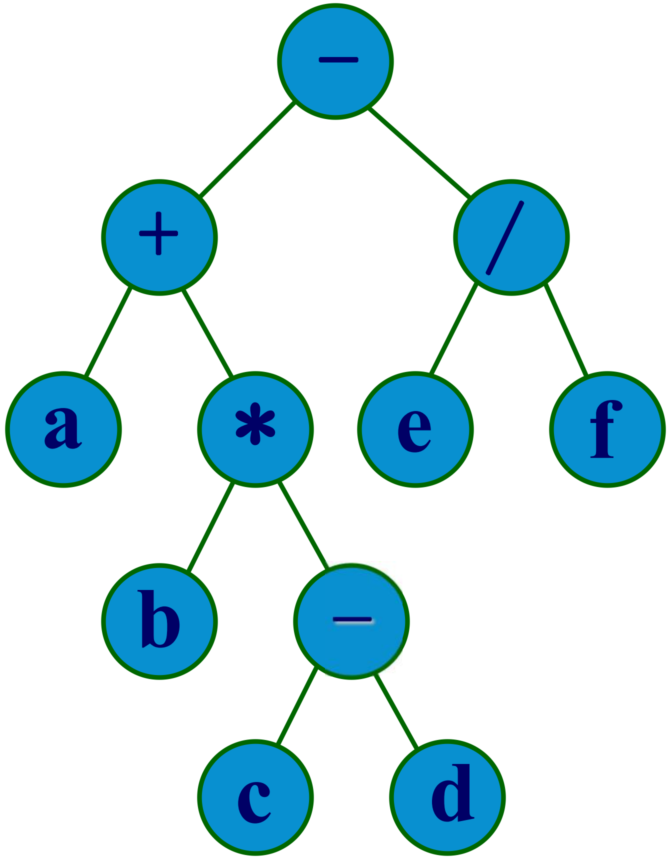
二叉树上的结点能排列在一个线性序列上。

这样可以使树状结构线性化。



二叉树的层次遍历

是从根结点开始遍历，按层次次序“**自上而下，从左至右**”访问树中的各结点。



- + / a * e f b - c d

二叉树的先中后序遍历

设：

访问根结点记作 V（或D），vertex（V）

遍历根的左子树记作 L,

遍历根的右子树记作 R,

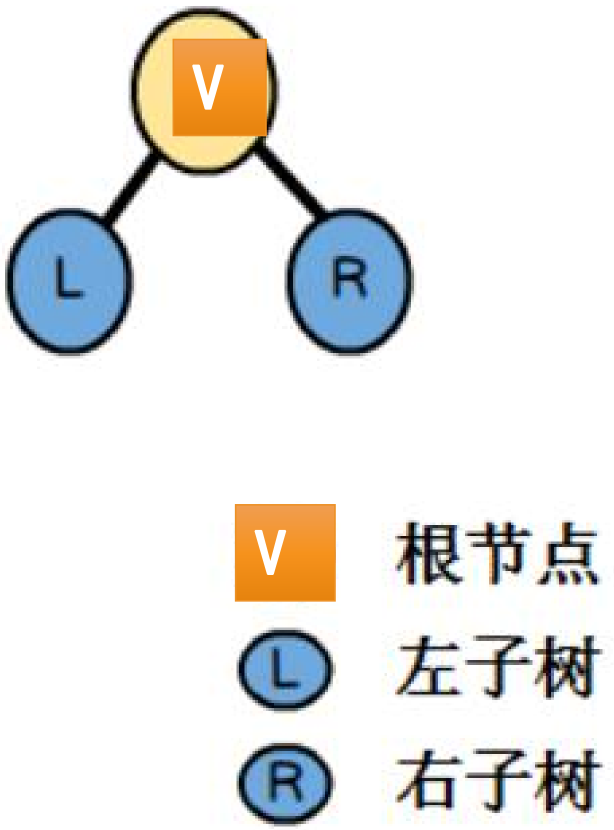
如根据访问根的先后，则可能的遍历次序有

前/先序　VLR（默认）　　镜像　　VRL

中序　LVR（默认）　　镜像　　RVL

后序　LRV（默认）　　镜像　　RLV

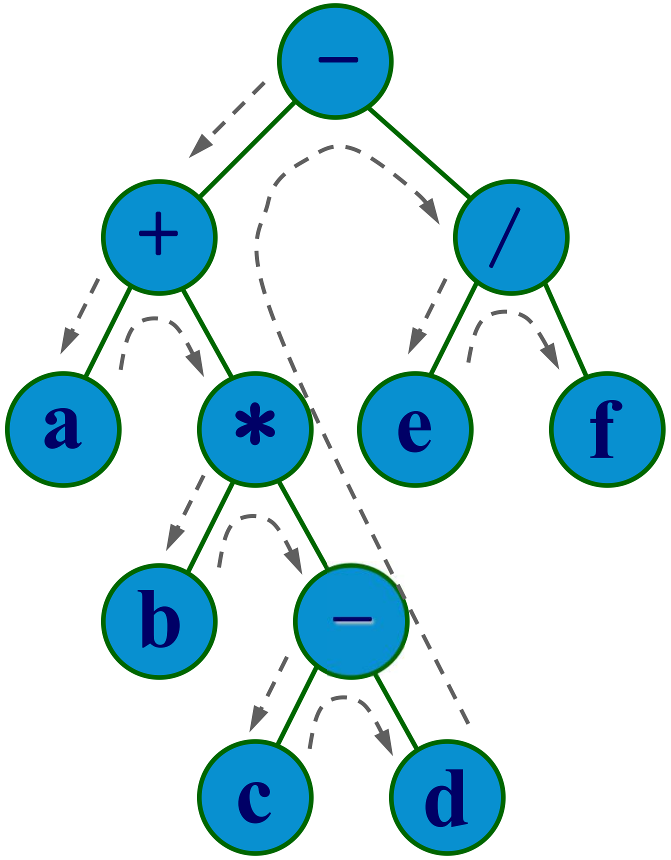
（或称为先根，中根，后根遍历）



前/先序遍历 (Preorder Traversal)

前序遍历二叉树算法的框架是：

- 若二叉树为空，则空操作；
- 否则
  - 访问根结点 (V)；
  - 前序遍历左子树 (L)；
  - 前序遍历右子树 (R)。



遍历结果

－ + a * b － c d / e f

（同学们自己遍历一遍）

中序遍历 (Inorder Traversal)

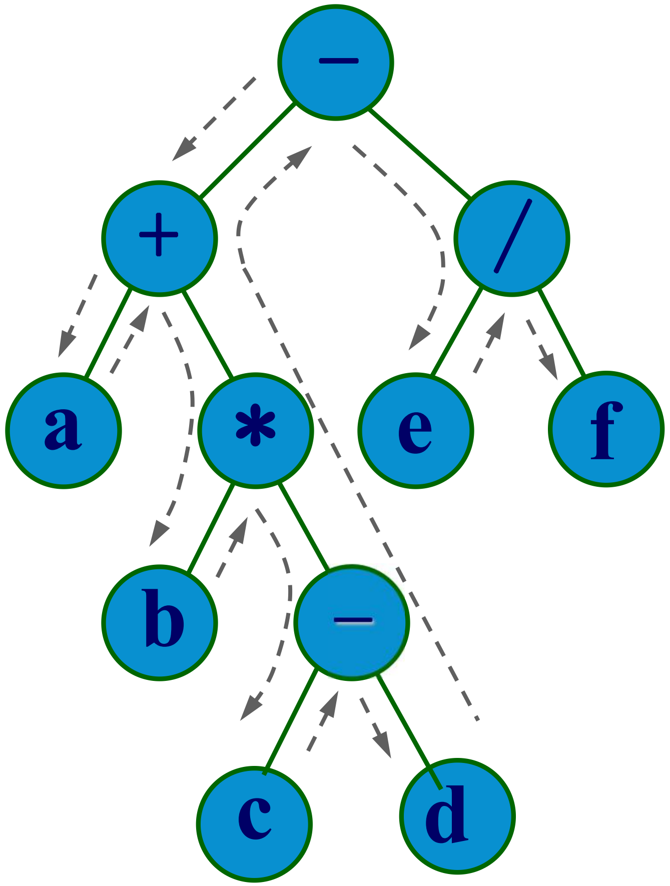
中序遍历二叉树算法的框架是：

- 若二叉树为空，则空操作；
- 否则
  - 中序遍历左子树 (L)；
  - 访问根结点 (V)；
  - 中序遍历右子树 (R)。

  遍历结果

  a + b * c - d - e / f

  （同学们自己遍历一遍）



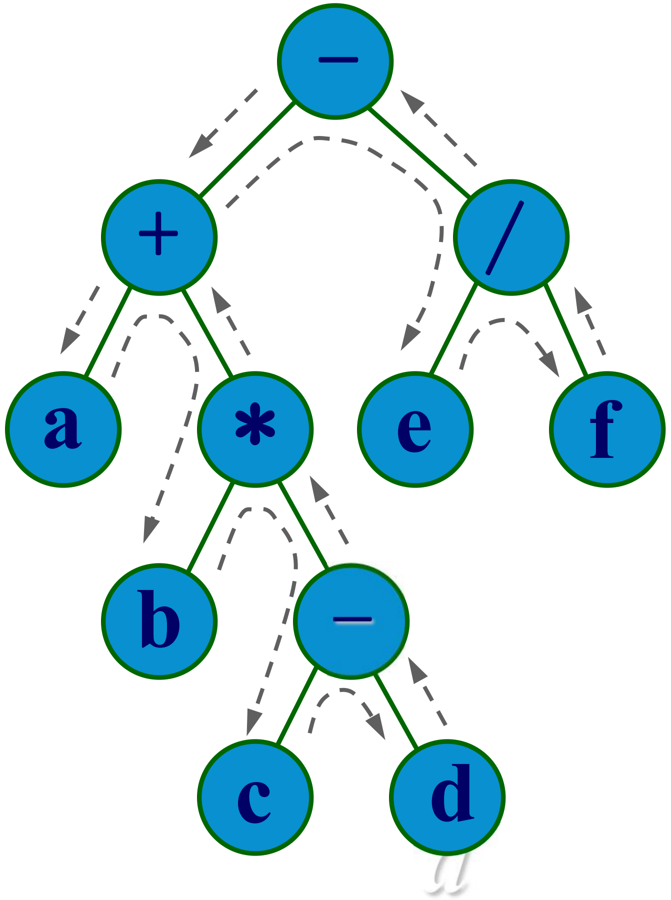
后序遍历 (Postorder Traversal)

后序遍历二叉树算法的框架是：
- 若二叉树为空，则空操作；
- 否则
  - 后序遍历左子树 (L)；
  - 后序遍历右子树 (R)；
  - 访问根结点 (V)。

遍历结果

a b c d – * + e f / –

（同学们自己遍历一遍）



周二进度：

二叉树的递归遍历实现

表达式树

基于二叉树递归的创建，销毁、求高度、求节点个数等