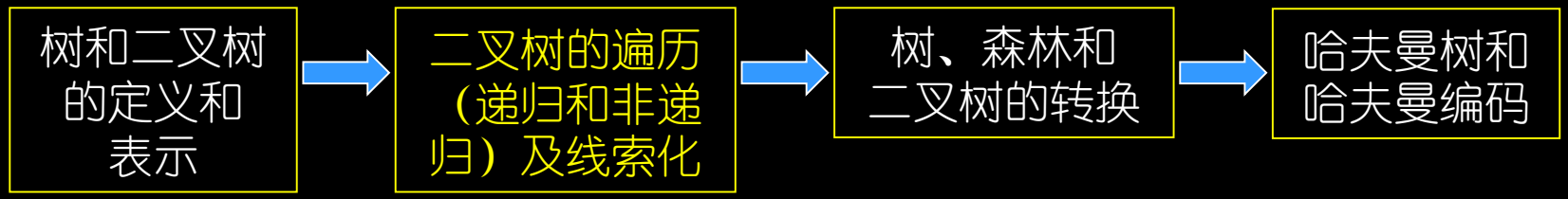
Chapter 06 Tree and binary tree
第六章 树和二叉树

本章学习的线索

- 主要线索



- 重点
  - 树和二叉树的定义及表示
  - 二叉树的遍历
  - 树、森林和二叉树的转换
  - 哈夫曼树和哈夫曼编码
- 难点
  - 二叉树的遍历及线索化

Contents

- Definition and notations of Tree and Forest
- Notations and representation of binary tree
- Binary tree traversal
- Threading binary tree
- Reconstruction of binary tree
- Transformation among Tree, Forest and binary tree
- Huffman tree and Huffman coding

6.1 Definition and notations of Tree

6.1.1 Definition of Tree

A tree is a set of n nodes. If n=0, it is a NULL tree. If n>0, then

1. It has a so-called “root” node which only has successors and without predecessor.

2. Except the root, the remainders can be partitioned into m (m>0) **un-overlapped** subsets $T_1, T_2, \ldots, T_m$, each of which is also a tree. They are called the sub-trees of the root.

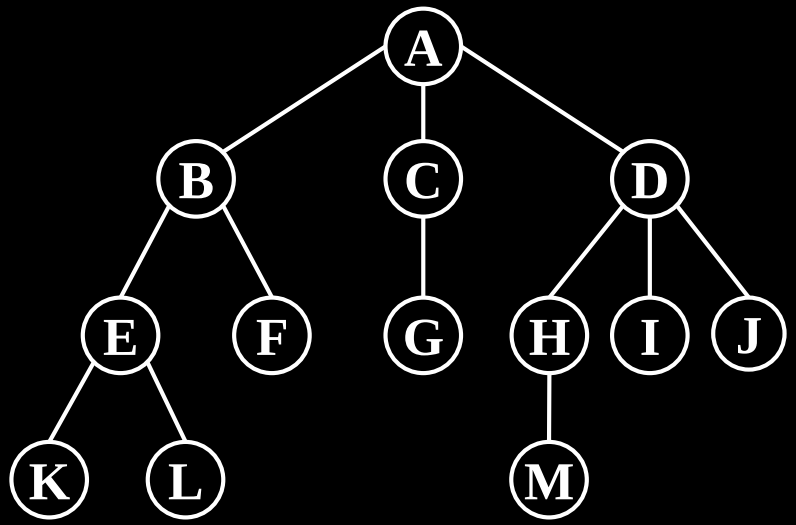
Note: The root of the tree has no predecessor and 0~m successors (sub-trees).

Representations of tree

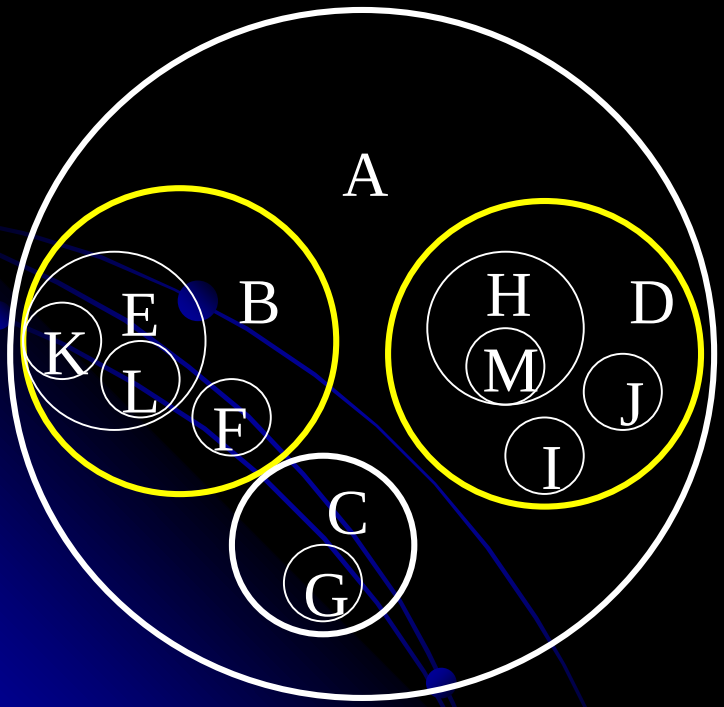
General list form

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

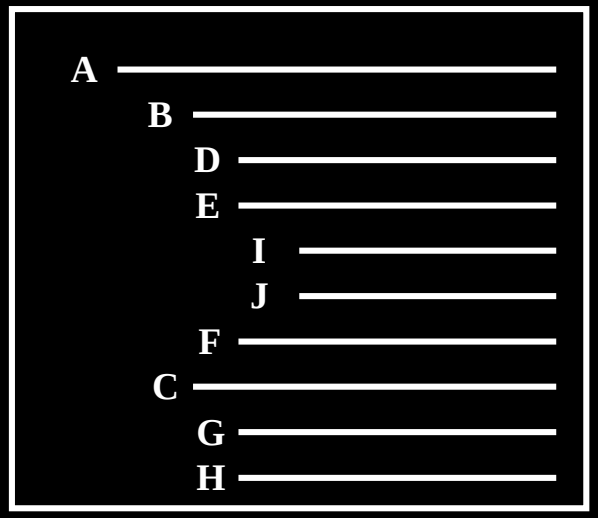
Tree form



Wen Graph



Retraction form



6.1.2 Notations

Node (结点)：包含一个**数据**元素及若干指向其子树的**分支**。

Leaf (树叶)、Branch node(分支结点)

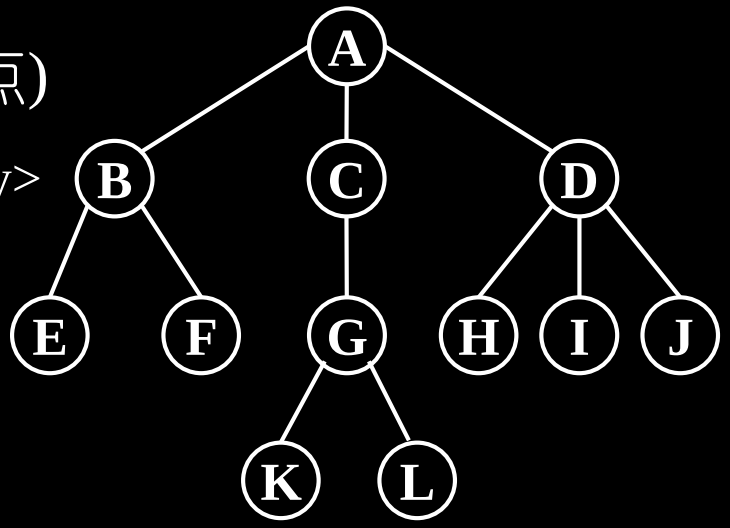
Parent node (父结点)、child node(子结点)

Edge (边)：父结点x到子结点y的有序对 <x,y>

Siblings (兄弟)：同一双亲的孩子

Ancestor (祖先)、Descendant (子孙)：

堂兄弟：双亲在同一层的结点



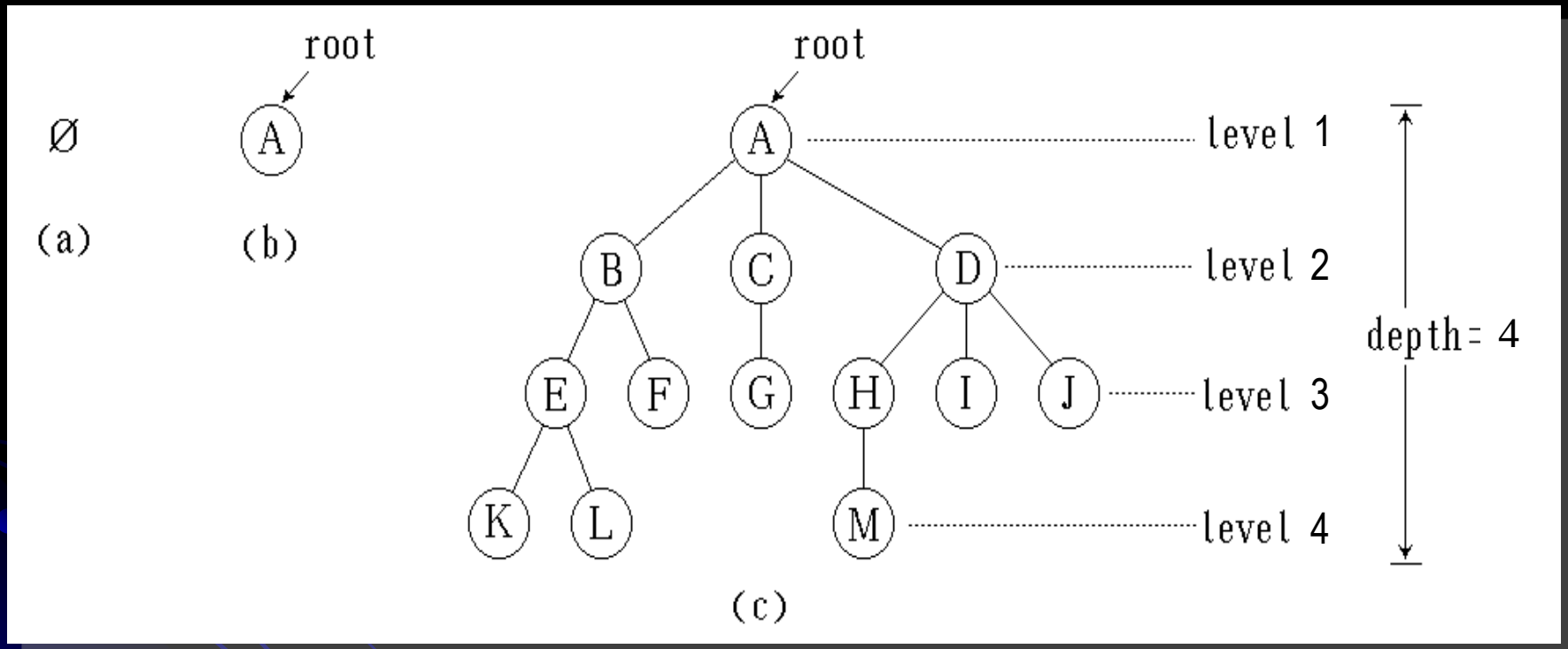
Layer (结点的层数、树的层数)：规定根的层数为1

Degree (结点的度、树的**度**)：结点拥有的子树数目

Depth (树的高度或**深度**)：树中结点的最大层次称为树的深度

Path and Path length (路径和路径长度)

Examples


root
Ø
A
(a)
(b)
root
A
B
C
D
E
F
G
H
I
J
K
L
M
level 1
level 2
level 3
level 4
depth = 4
(c)

recursive definition of Tree

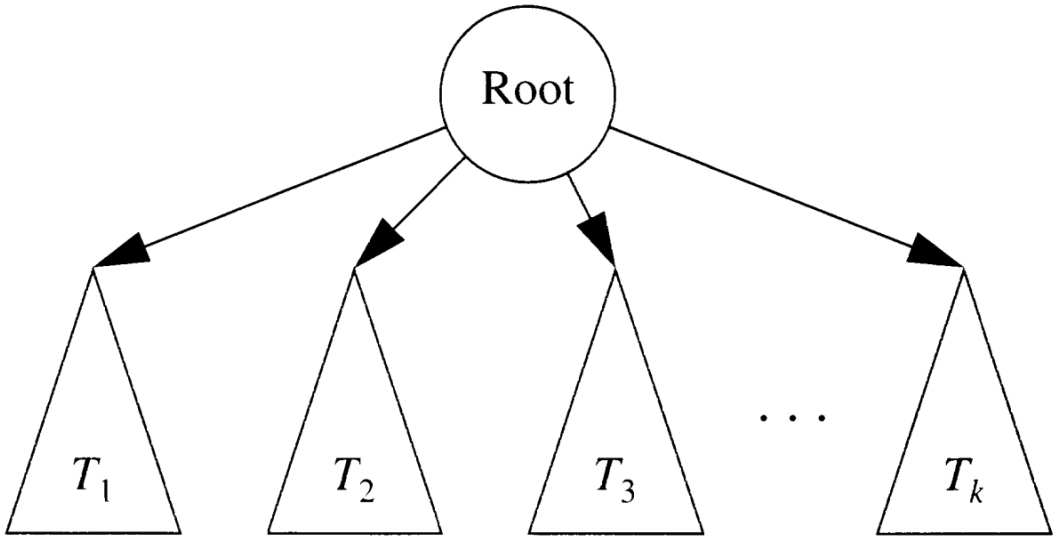
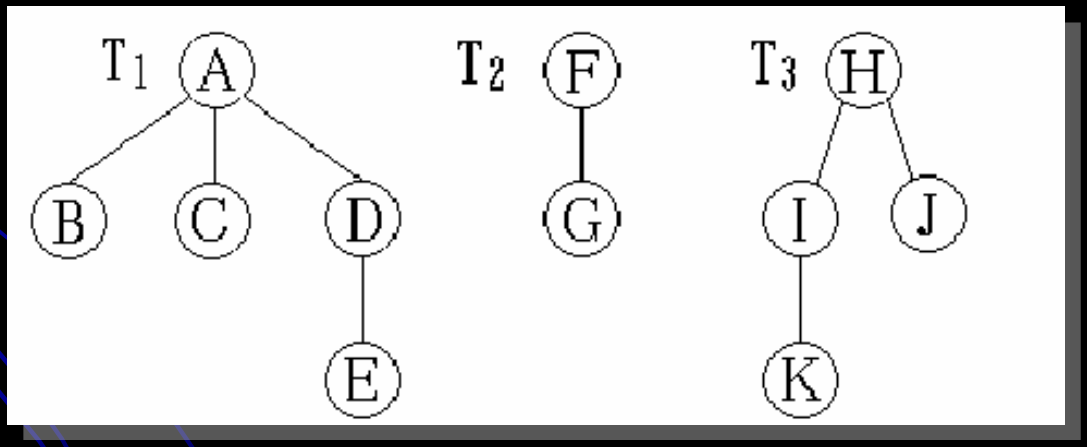


Figure 18.2 A tree viewed recursively.

6.1.3 Forest (Orchard)

A set of trees. 森林是m (m>=0)棵互不相交的树的集合。对树中每个结点而言，其子树的集合即为森林。

就逻辑结构而言，任何一棵树是一个二元组Tree=(root, F)，其中root称为树的根结点；F是m (m>=0)棵子树构成的森林，$F=(T_1, T_2,\ldots,T_m)$, 其中$T_i=(r_i, F_i)$称作根root的第i棵子树



6.1.4 Basic functions for the tree

树的基本运算通常有以下几种：

1. Initialization - 创建一棵空树；
2. IsEmpty - 判断某棵树是否为空；
3. GetRoot - 求树中的根结点，若为空树，则返回一特殊值；
4. GetParent - 求树中某个指定结点的父结点，当指定结点为根时，返回一特殊值；
5. **GetFirstChild** - 求树中某个指定结点的最左子结点，当指定结点为树叶时，它没有子女，则返回一特殊值；
6. **GetRightSibling** - 求树中某个指定结点的右兄弟结点，当指定结点没有右兄弟时，返回一特殊值；
7. Traversal - 树的遍历（周游），即按某种方式访问树中的所有结点，并使每个结点恰好被访问一次。

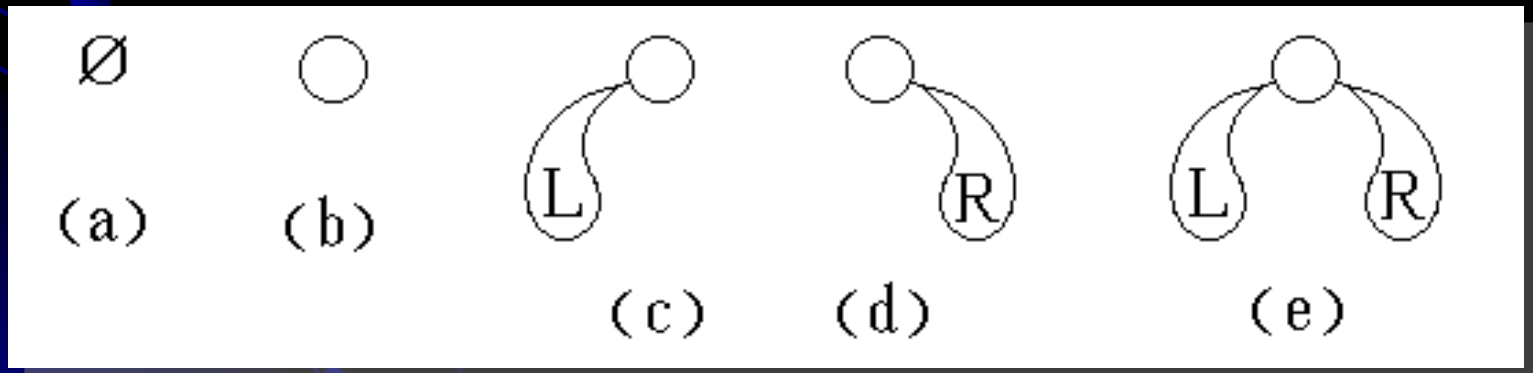
线性结构	树型结构
第一个数据元素 (无前驱)	根结点 (无前驱)
最后一个数据元素 (无后继)	多个叶子结点 (无后继)
其它数据元素 (一个前驱、 一个后继)	其它数据元素 (一个前驱、 多个后继)

6.2 Binary tree

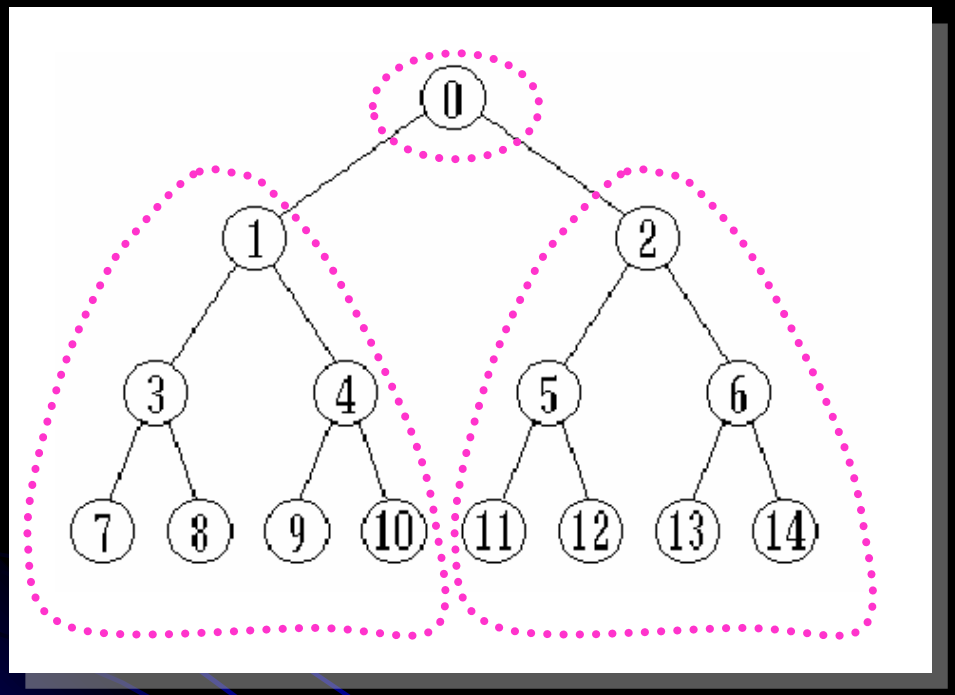
6.2.1 Basic notations

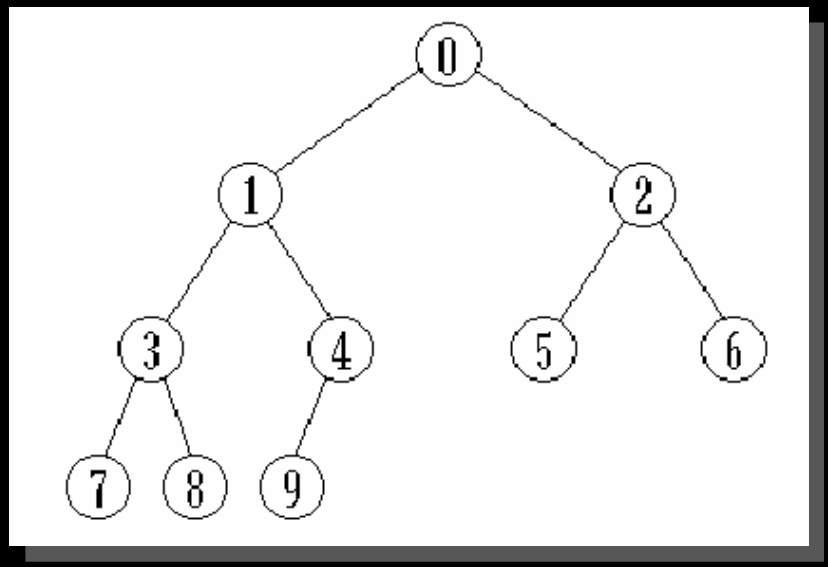
Binary Tree (二叉树)：它的每一个结点至多有两棵子树（也是二叉树），并且子树有 左右 之分，其次序不能随意颠倒。

二叉树的5种基本形态：



Examples of binary tree





Binary tree = (Root, Left sub-binary-tree, Right sub-binary-tree)

二叉树不是树的特殊情形，尽管树和二叉树的概念之间有许多类似，但它们是两个概念。树和二叉树之间最主要的差别是：

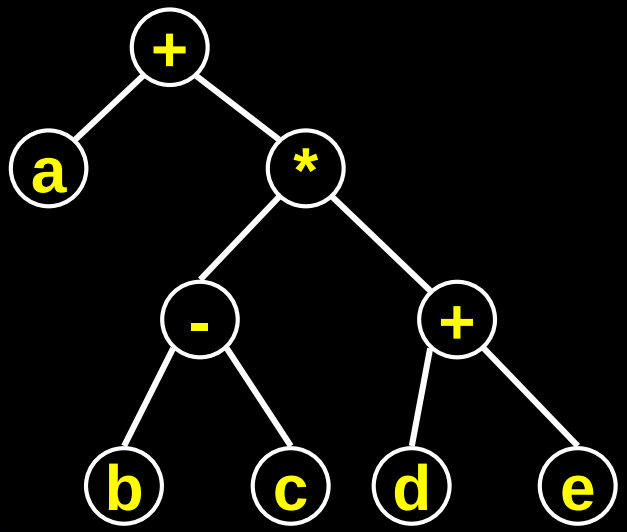
★ 二叉树中结点的子树要区分为**左子树和右子树**，即使在结点只有一棵子树的情况下也要明确指出该子树是左子树还是右子树。

★

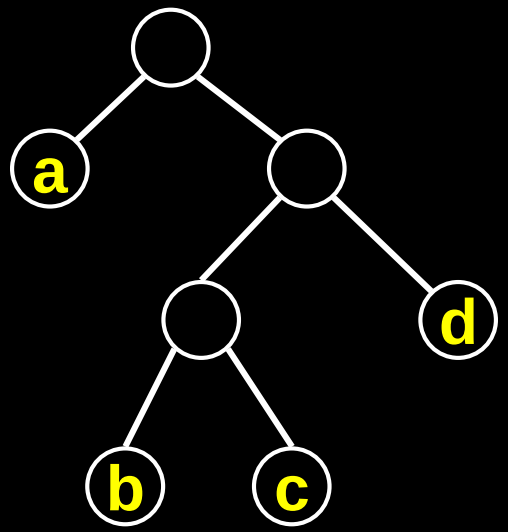
Full binary tree (满二叉树)：如果一棵二叉树的任何结点或者

完全二叉树：除最后一层外，每一层上的节点数均达到最大值；在最后一层上只缺少右边的若干结点。

Complete binary tree (完全二叉树)：如果一棵二叉树至多只有最下面的两层结点度数可以小于2（度为1的结点只能在倒数第二层），并且最下面一层的结点（叶子）都集中在该层最左边的若干位置上，则此二叉树称为“完全二叉树”。完全二叉树不一定是满二叉树。



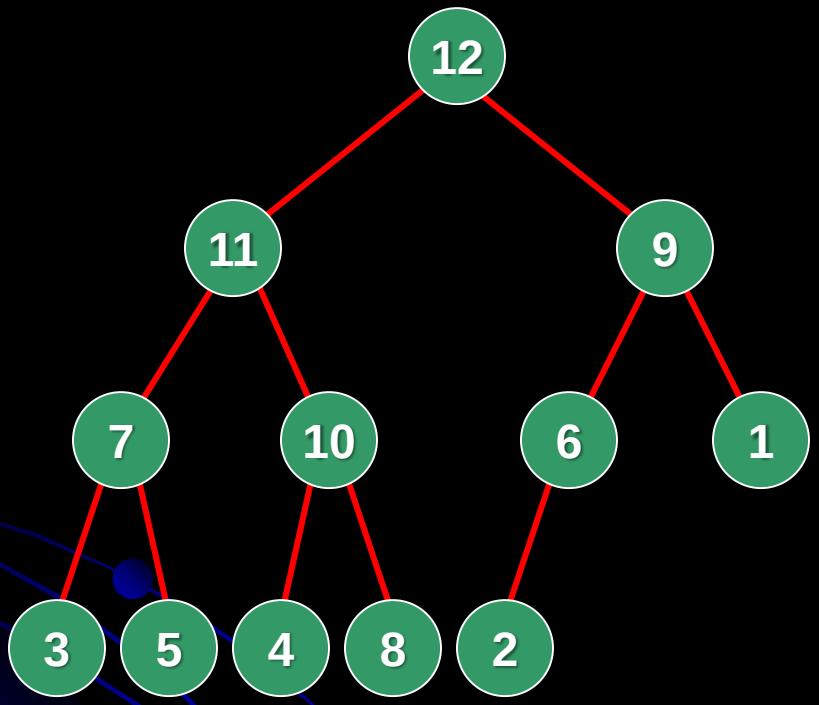
Expression tree



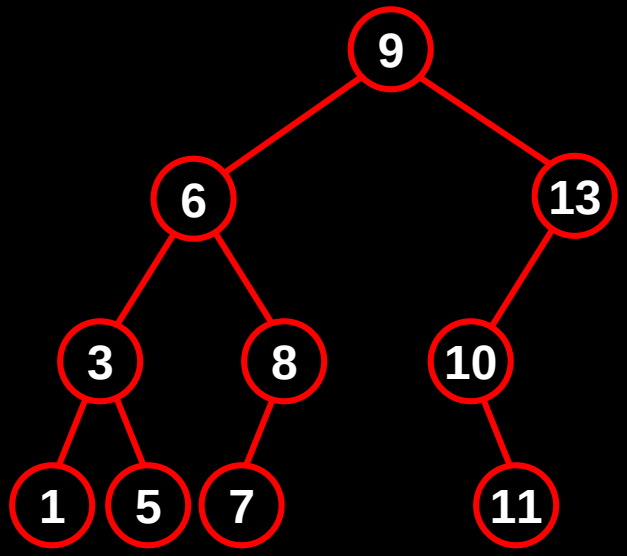
Huffman tree

a+(b-c)*(d+e)

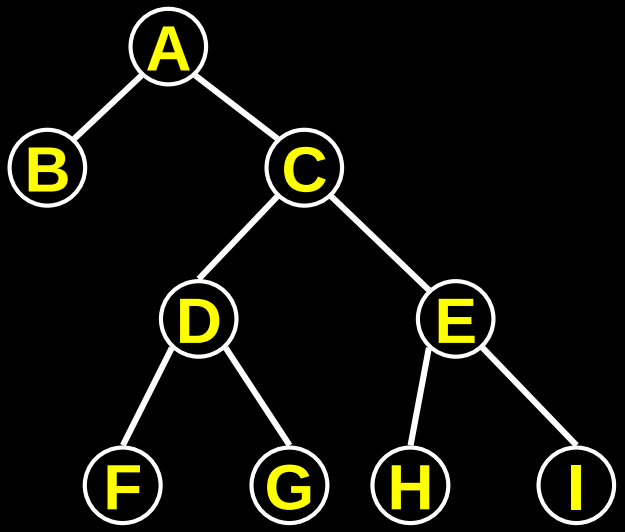
a: 0
b: 100
c: 101
d: 11



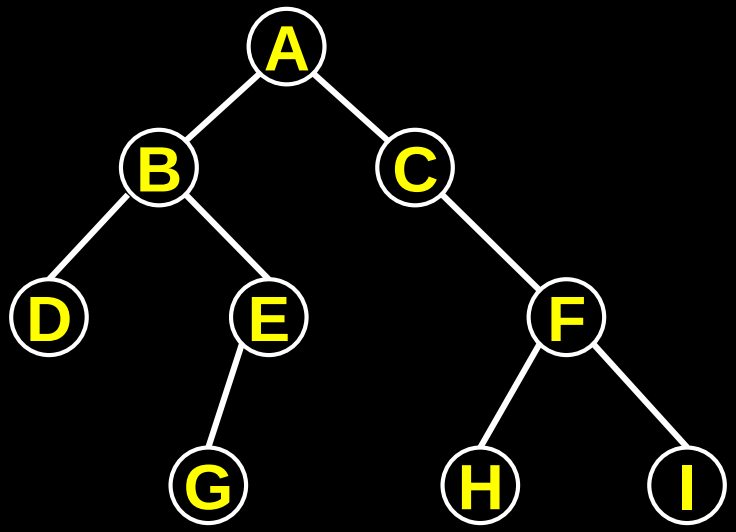
"Maximum" Heap



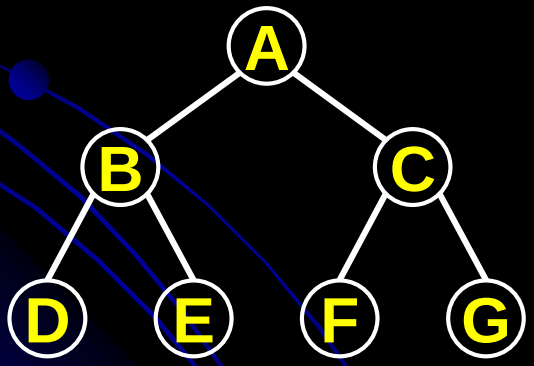
Binary Search Tree



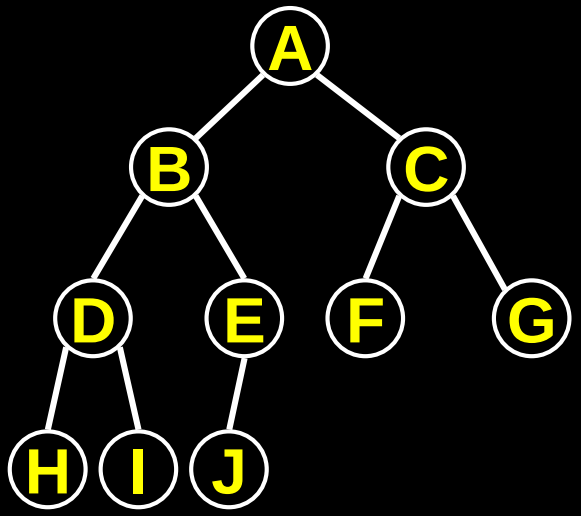
Generic binary tree 1



Generic binary tree 2



Full binary tree



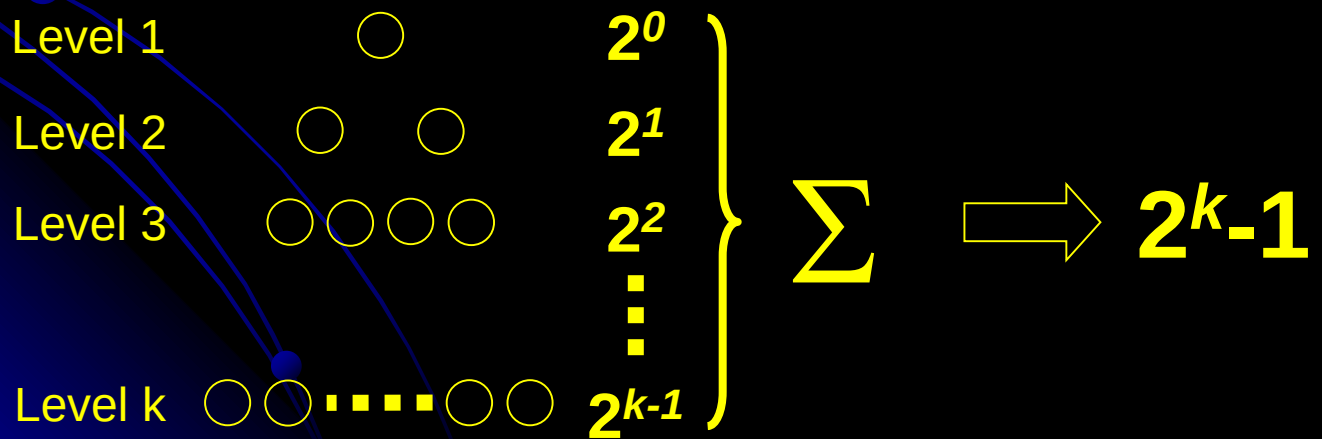
Complete binary tree

6.2.2 Characteristics of binary tree

(1) Suppose that binary tree is leveled from 1, there are at most 2^{i-1} nodes at the level i ($i \geq 1$).
在二叉树的第i层上至多有2^{i-1}个结点（$i \geq 1$）。

可以用归纳法证明。

(2) If the height of binary tree is k ($k \geq 1$), the number of nodes is at most 2^k-1.
深度为k的二叉树至多有2^k-1个结点（k>=1）。



6.2.2 Characteristics of binary tree

(3) In binary tree, if the number of leaves is n_0, and the number of nodes with left and right children is n_2, then

$$n_0 = n_2 + 1$$

对任何一棵二叉树T，如果其终端结点数为n_0，度为2的结点数为n_2，则$n_0=n_2+1$。

证明： 设n_1为二叉树中度为1的结点数。则有，

$$n=n_0+n_1+n_2$$

再看二叉树中的分支数。除了根结点外，其余结点都有一个分支指向它，设B为分支总数，则$n=B+1$。由于这些分支都是由分支为1或2的结点发出的，因此有：

$$B=n_1+2n_2$$

于是有： $n=n_1+2n_2+1$

$$n_1+2n_2+1=n_0+n_1+n_2 \rightarrow n_0=n_2+1$$

(4) The depth of the complete binary tree with n nodes is

$$\lfloor \log_2 n \rfloor + 1$$

具有n个结点的完全二叉树的深度为

$$\lfloor \log_2 n \rfloor + 1$$

证明：假设完全二叉树的深度为k，则根据性质2和完全二叉树的定义有

$2^{k-1}-1 < n <= 2^k-1$　　或　　$2^{k-1} <= n < 2^k$

于是

$$k-1 <= \log_2 n < k$$

因k是整数，所以　　$k = \lfloor \log_2 n \rfloor + 1$

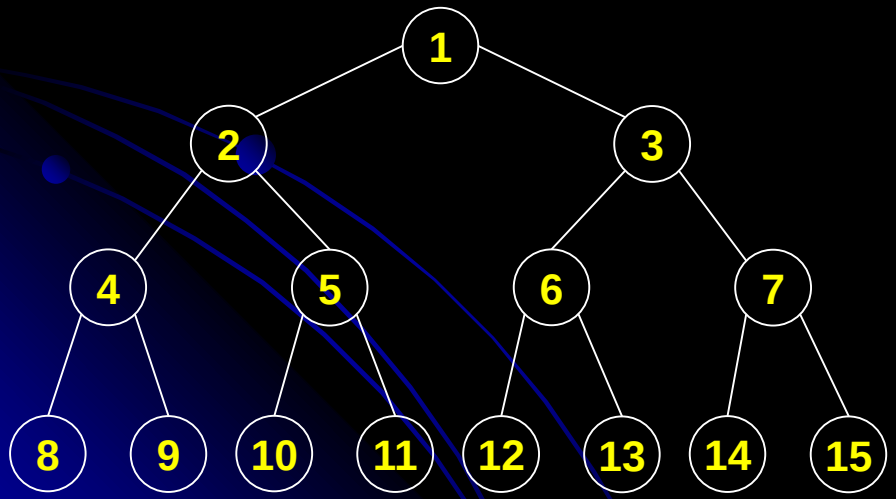
(5) 如果对一棵有n个结点的完全二叉树按层序从1开始编号，则对任一结点 i （$1<=i<=n$），有：

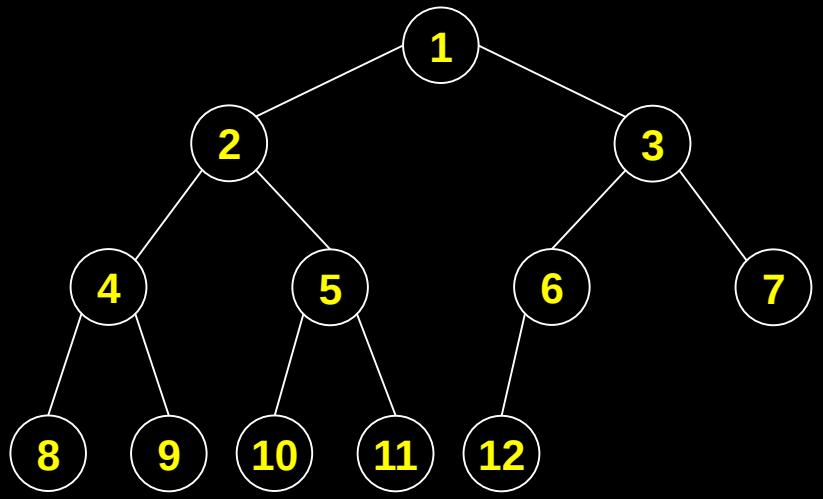
a) 如果i=1, 则结点i是二叉树的根, 无双亲; 如果i>1, 则其双亲结点是$\lfloor i/2 \rfloor$;

b) 如果2i>n, 则结点i无左孩子；否则其左孩子是结点2i;

c) 如果2i+1>n, 则结点i无右孩子；否则其右孩子是结点2i+1.

证明：数学归纳法。





6.2.3 Basic functions for binary tree

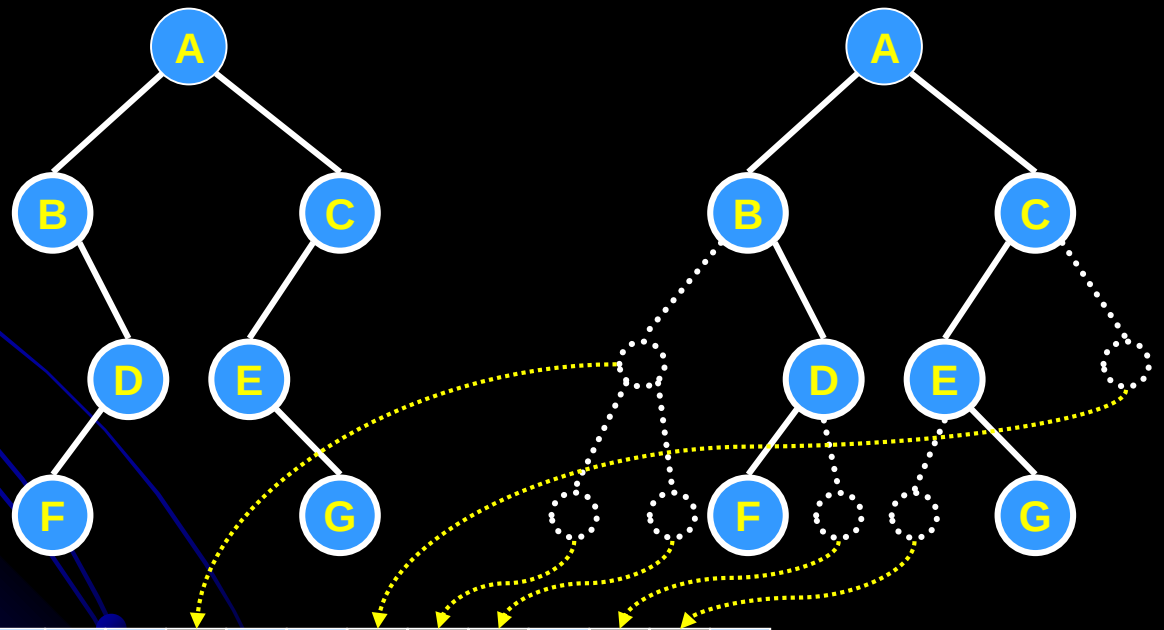
二叉树的基本运算通常有以下几种：

1. Initialization - 创建一棵空二叉树；
2. IsEmpty - 判断某棵二叉树是否为空；
3. GetRoot -求二叉树中的根结点，若为空二叉树，则返回一特殊值；
4. GetParent - 求二叉树中某个指定结点的父结点，当指定结点为根时，返回一特殊值；
5. GetLeftChild - 求二叉树中某个指定结点的左子女结点，当指定结点没有左子女时，返回一特殊值；
6. GetRightChild - 求二叉树中某个指定结点的右子女结点，当指定结点没有右子女时，返回一特殊值；
7. Traversal - 二叉树的遍历，即按某种方式访问二叉树中的所有结点，并使每个结点仅仅被访问一次。

6.3 Storage of Binary Tree

6.3.1 Sequential form

用一组**地址连续**的存储单元依次自上而下、自左而右存储完全二叉树的结点。对于一般树，则应将其每个结点与完全二叉树上的结点相对照，存储在一维数组的相应分量中。



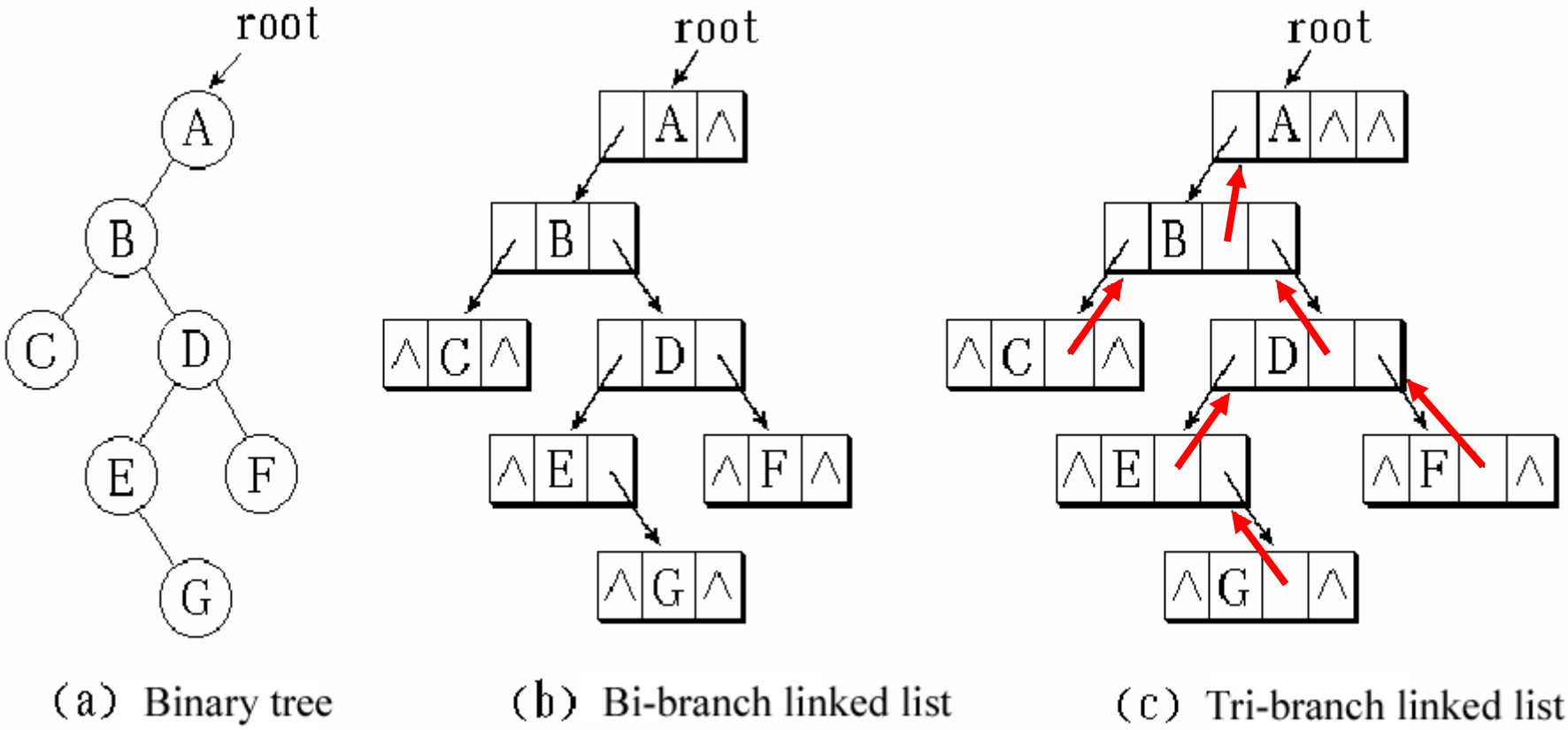
```
//Declaration
#define  MAXNODE  100     /* 定义二叉树中结点的最大个数 */
typedef struct SeqBTree       /* 顺序二叉树类型定义 */
{
      DataType  nodelist[MAXNODE+1];
      int  n;  /* 改造成完全二叉树后，结点的个数 */
}SeqBTree, *PSeqBTree;
```

6.3.2 Linked form

由二叉树的定义可知，二叉树的结点由一个数据元素和两个分支构成，因此在表示时，至少需要包含三个域：数据域和左、右指针域。如果想能够找到父结点，则可以增加一个指向父结点的指针域。利用这两种结点结构所得的二叉树存储结构分别称为二叉链表和三叉链表。

```
//Declaration
typedef struct BinTreeNode
{
    DataType  info;
    struct BinTreeNode  *lchild;
    struct BinTreeNode  *rchild;
}BinTreeNode, *PBinTreeNode, BinTree, *PBinTree;
```

Example



(a) Binary tree (b) Bi-branch linked list (c) Tri-branch linked list