树的存储结构

双亲表示法

假设以一组连续空间存储树的结点

```
//结点结构
typedef struct Node{
    ElementType     data;       //数据域
    int             parent;     //双亲位置域
} Node;

//树结构
typedef struct {
    Node            nodes[MAX_TREE_SIZE];       //一维数组存储结点
    int             root;                       //根的位置
    int             count;                      //结点数
} Tree;
```

数组下标	data	parent
0	R	-1
1	A	0
2	B	0
3	C	0
4	D	1
5	E	1
6	F	3
7	G	6
8	H	6
9	K	6

这种存储结构利用了每个结点（除根以外）只有唯一双亲的性质

PARENT(i)操作可以在常量时间内完成

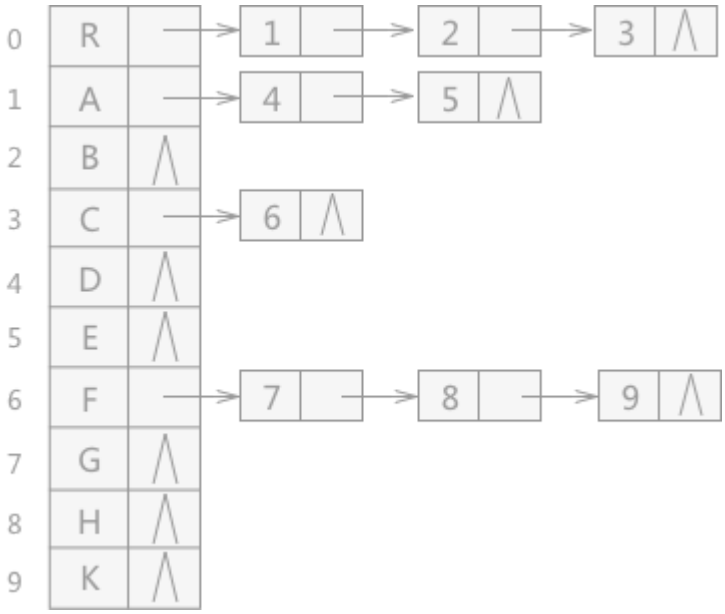
反复调用PARENT(i)操作，直到遇到无双亲的结点时，便找到了树的根，这就是ROOT(i)操作的执行过程

但是在这种结构中，求结点的孩子需要遍历整个结构

孩子表示法

把每个结点的孩子结点排列起来，看成是一个线性表，且以单链表作存储结构

则n个结点有n个孩子链表（叶子的孩子链表是空表）



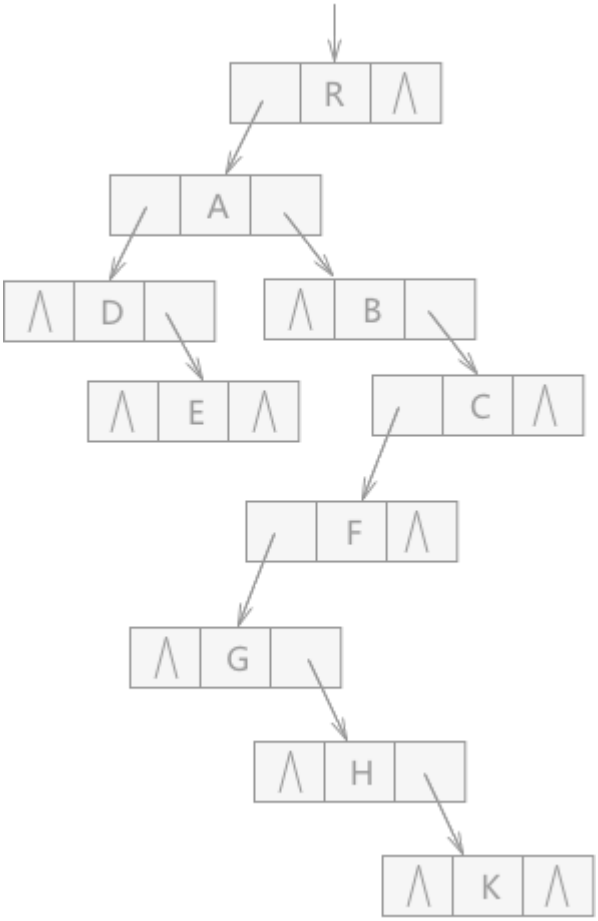

```
typedef struct Node {
    int             child;
    struct Node     *next;
} *ChildPointer;

typedef struct {
    ElementType     data;       //数据域
    ChildPointer    child;      //孩子链表指针
} CTBox;

typedef struct {
    CTBox           nodes[MAX_TREE_SIZE];
    int             root;       //根的位置
    int             count;      //结点数
} Tree;
```

孩子兄弟表示法

又称二叉树表示法，或二叉链表表示法

即以二叉链表作为树的存储结构

链表中结点的两个链域分别指向结点的第一个孩子和下一个兄弟结点，分别命名为firstChild和nextSibling域




```
typedef struct Node {
    ElementType         data;           //数据域
    struct Node         firstChild;     //指向第一个孩子结点
    struct Node         nextSibling;    //指向下一个兄弟结点
}
```

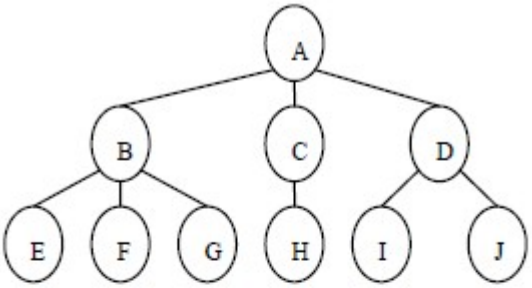
森林与二叉树的转换

由于二叉树和树都可以用二叉链表作为存储结构，则二叉链表作为媒介可以推导出树与二叉树的对应关系；

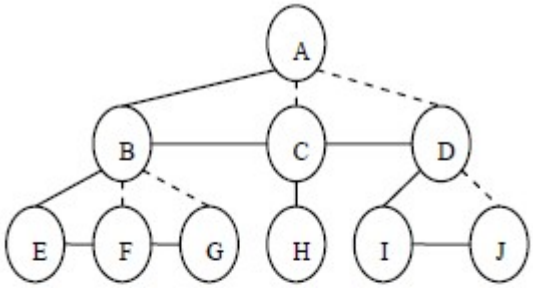
给定一棵树，可以找到唯一一颗二叉树与其对应

转换步骤

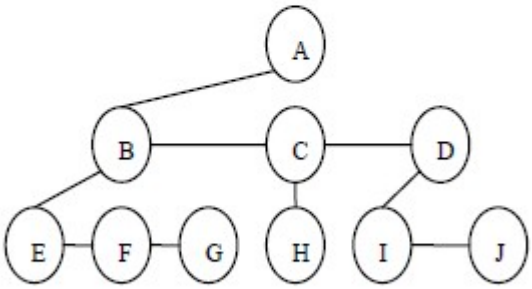
- 加线；在所有兄弟结点之间加一条连线
- 抹线；对树中每个结点，只保留他与第一个孩子结点之间的连线，删除它与其他孩子结点之间的连线；
- 旋转；以树的根节为轴心，将整棵树顺时针旋转一定角度，使之结构层次分明



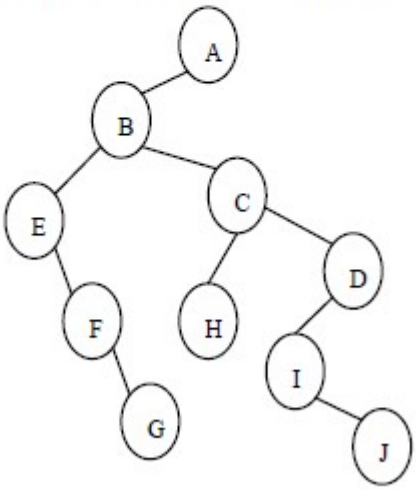
(a) 树



(b) 加线（虚线表示要删除的线）



(c) 抹线



(d) 旋转