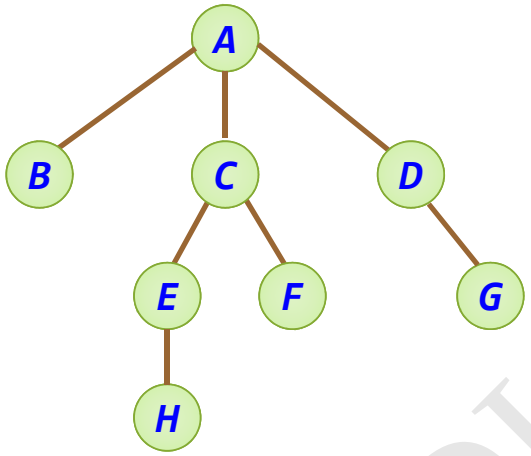
6.4 树和森林

6.4.1 树的存储结构

在大量的应用中，人们曾使用多种形式的存储结构来表示树。这里介绍3种常用的链表结构。

1. 双亲存储结构

这种存储结构是一种顺序存储结构，用一组连续空间存储树的所有结点，同时在每个结点中附设一个下标（伪指针）指示其双亲结点的位置。



下标	data	parent
0	A	-1
1	B	0
2	C	0
3	D	0
4	E	2
5	F	2
6	G	3
7	H	4

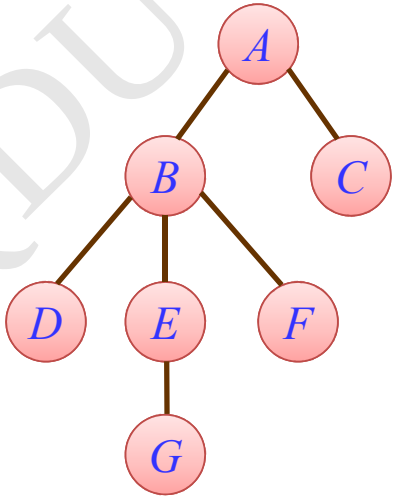
- 这种存储结构利用了每个结点（除根以外）只有惟一的双亲的性质。
- 但是，在这种表示法中，求结点的孩子时需要遍历整个结构。

双亲存储结构的类型声明如下：

```
// ------- 树的双亲表存储表示 -----------
#define MAX_TREE_SIZE  100
struct PTNode {
   TElemType data;
   int parent; // 双亲位置域
};
struct PTree {
   PTNode nodes[MAX_TREE_SIZE];
   int  r,  n;                    // 根的位置、结点数
};
```

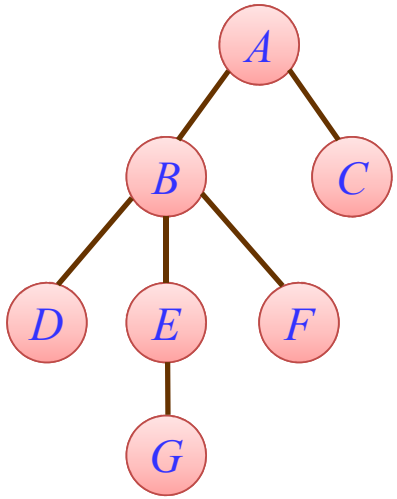
2．孩子表示法

由于树中每个结点可能有多棵子树，则可用多重链表，即每个结点有多个指针域，其中每个指针指向一棵子树的根结点，此时链表中的结点可以有如下两种结点格式。

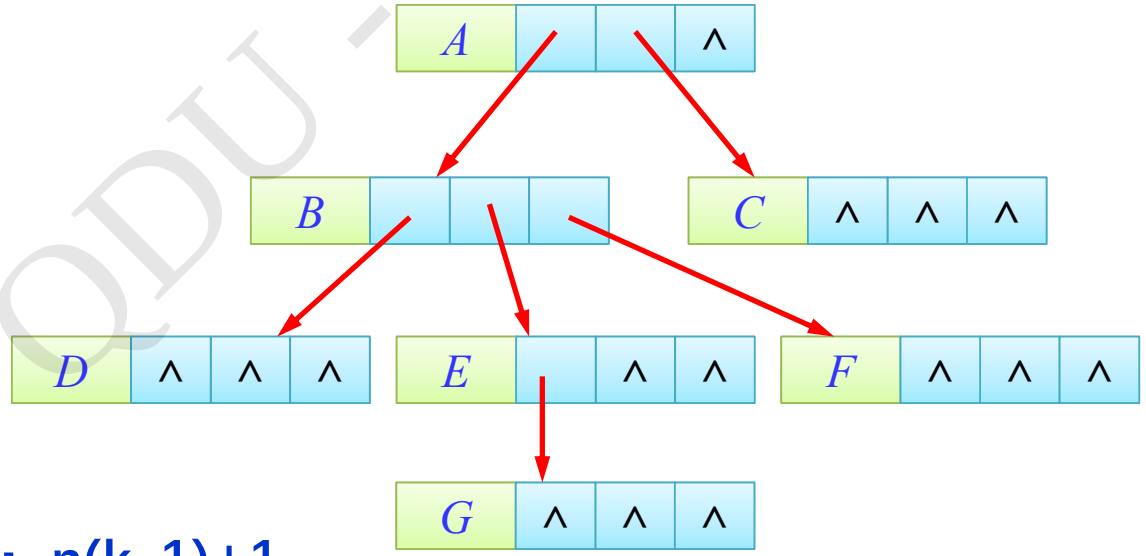


第一种方法：定长结点链表存储结构

孩子链表存储结构可按树的度（即树中所有结点度的最大值）设计结点的孩子结点指针域个数。



每个指针指向一颗子树

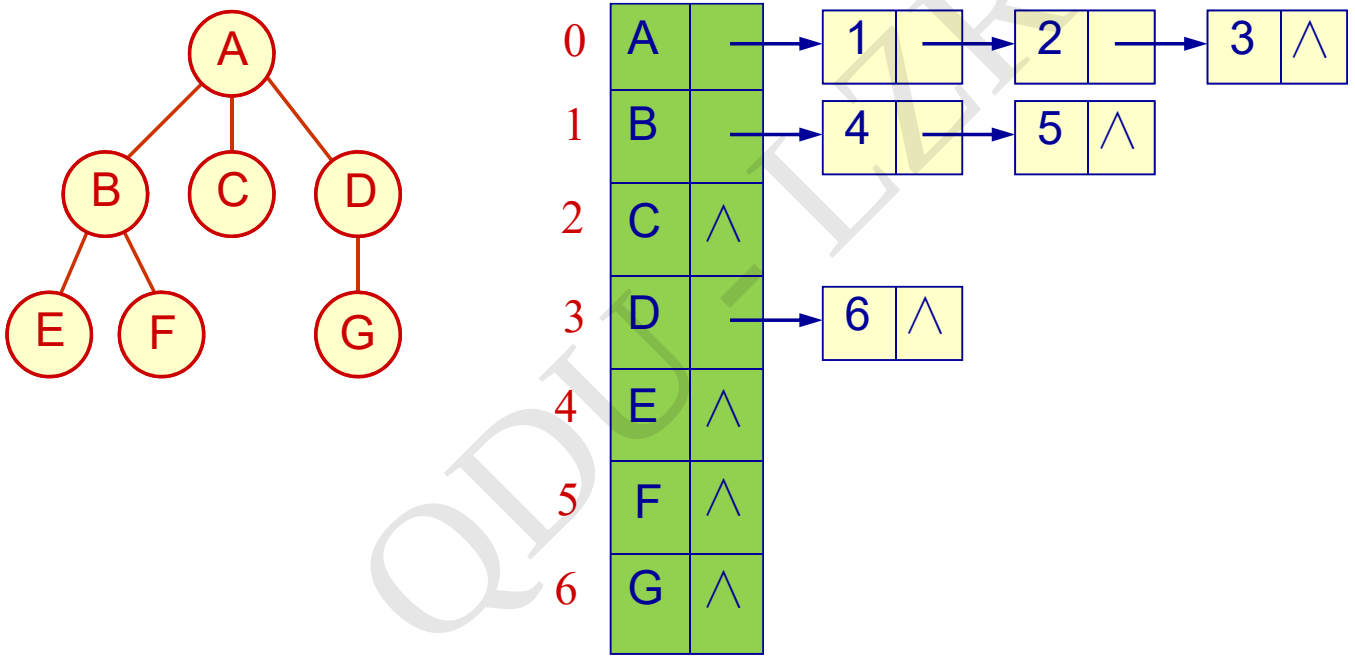


空指针域：n(k-1)+1

第二种方法：孩子链表存储结构

- 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(叶子的孩子链表为空表）。
- 而n个头指针又组成一个线性表，为了便于查找，可采用顺序存储结构。

第二种方法：孩子链表存储结构



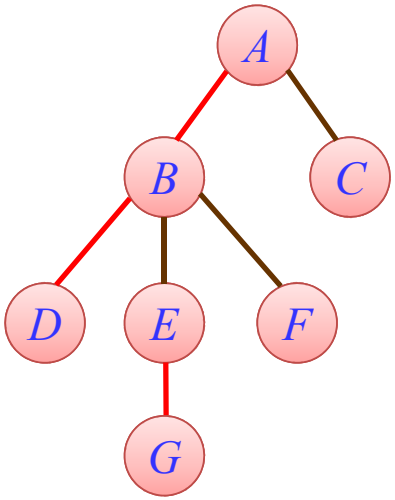
- 与双亲表示法相反， 孩子表示法便于那些涉及孩子的操作的实现，却不适用于 **PARENT(T,x)**操作。

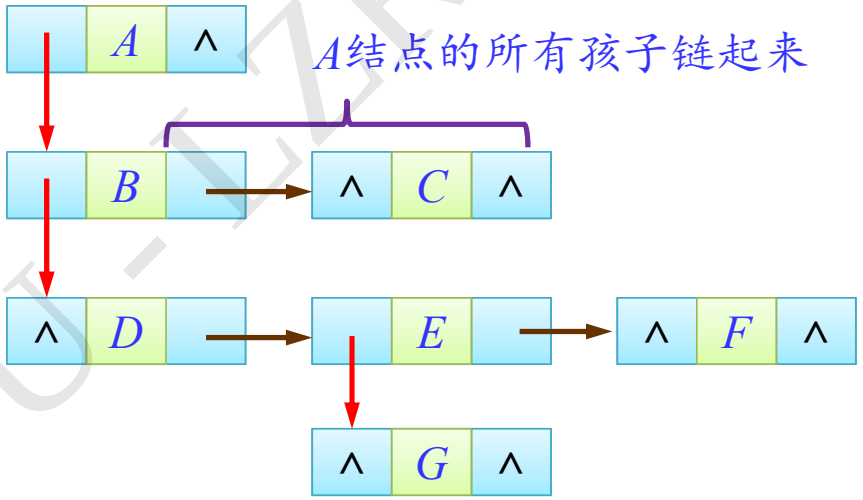
3．孩子兄弟表示法

- 又称二叉树表示法，或二叉链表表示法。
- 以二叉链表作树的存储结构。链表中结点的两个链域分别指向该结点的第一个孩子结点和下一个兄弟结点，分别命名为firstchild域和nextsibling域。

```
// ----- 树的二叉链表(孩子-兄弟)存储表示 ------
typedef struct CSNode {
    TElemType data;
    CSNode *firstchild, *nextsibling;
} CSNode, *CSTree;
```

3．孩子兄弟表示法





树的孩子兄弟链表存储结构

END