7.1.5 树的基本运算

树的运算主要分为三大类：

- 查找满足某种特定关系的节点，如查找当前节点的双亲节点等；
- 插入或删除某个节点，如在树的当前节点上插入一个新节点或删除当前节点的第i个孩子节点等；
- 遍历树中每个节点。

树的遍历

树的遍历运算是指按某种方式访问树中的每一个节点且每一个节点只被访问一次。

主要的遍历方法：

- 先根遍历：

  若树不空，则先访问根节点，然后依次先根遍历各棵子树。

- 后根遍历：

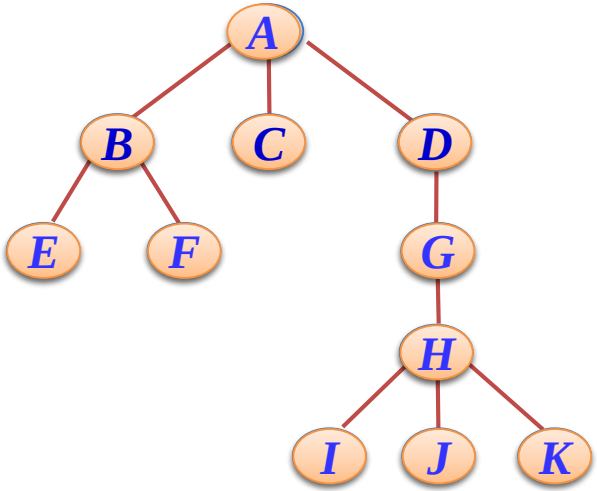
  若树不空，则先依次后根遍历各棵子树，然后访问根节点。

- 层次遍历：

  若树不空，则自上而下、自左至右访问树中每个节点。

注意：先根和后根遍历算法都是递归的。

树的先根遍历示例的演示

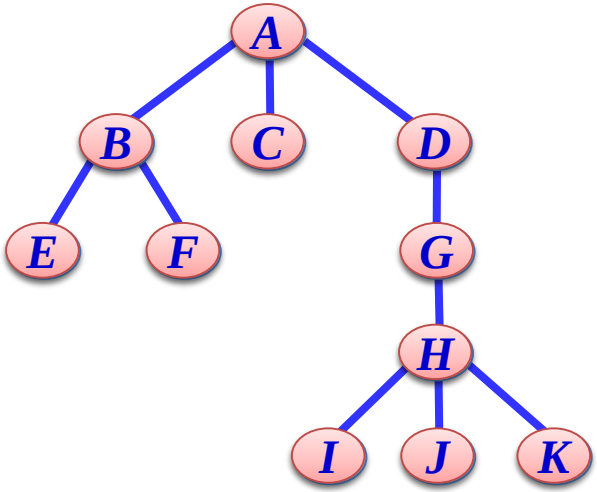


先根遍历的顶点访问次序：

A B E F C D G H I J K

遍历完毕

树的后根遍历示例的演示

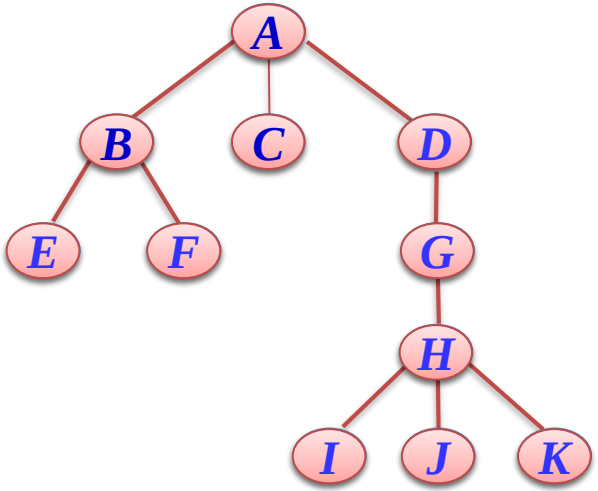


后根遍历的顶点访问次序：

E F B C I J K H G D A

遍历完毕

树的层次遍历示例的演示



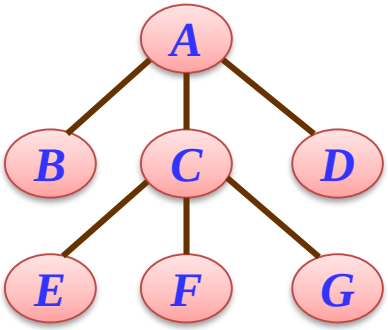
层次遍历的顶点访问次序：

A B C D E F G H I J K

遍历完毕

7.1.6 树的存储结构

1、双亲存储结构



伪指针指示其双亲节点的位置

0	A	-1
1	B	0
2	C	0
3	D	0
4	E	2
5	F	2
6	G	2

树中任何节点只有唯一的双亲节点

树的双亲存储结构

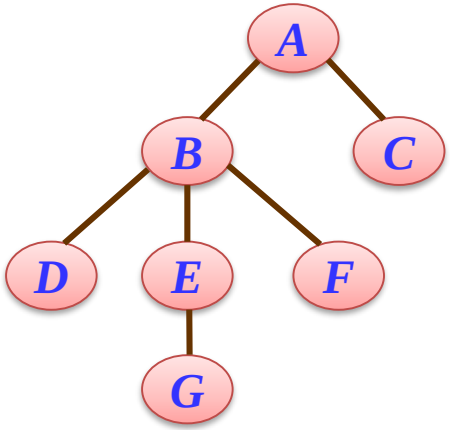
双亲存储结构的类型声明如下：

```
typedef struct
{    ElemType data;          //节点的值
     int parent;             //指向双亲的位置
} PTree[MaxSize];
```

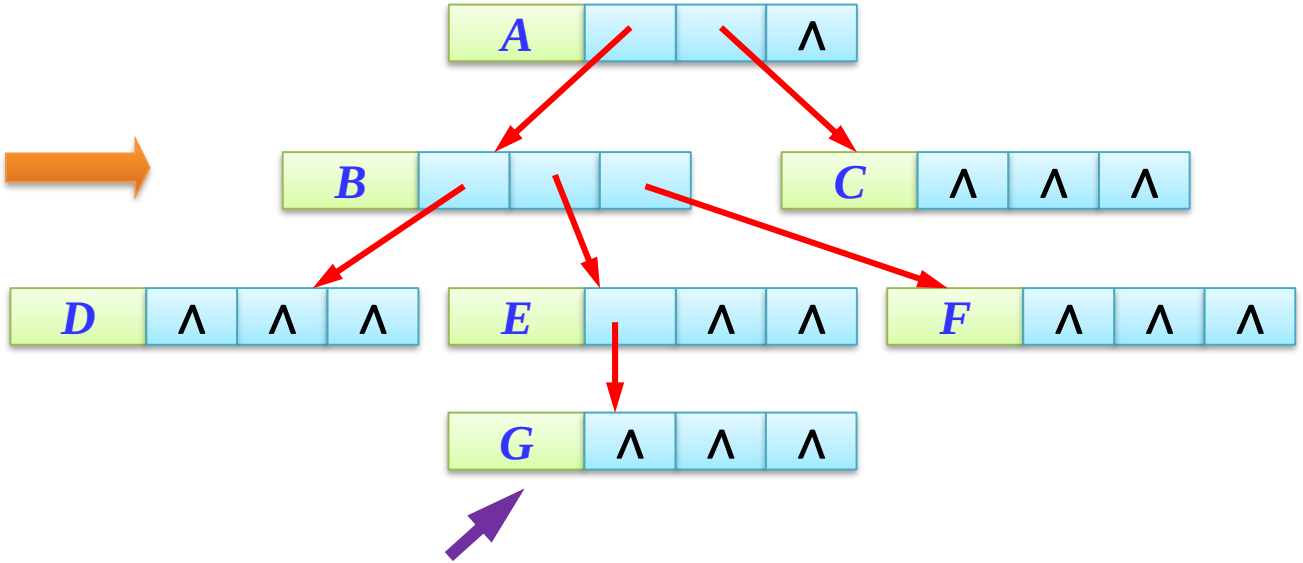
思考题： 该存储结构的优缺点？

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

2、孩子链存储结构



每个指针指向一颗子树



树的孩子链存储结构

孩子链存储结构的节点类型声明如下：

```
typedef struct node
{     ElemType data;                    //节点的值
      struct node *sons[MaxSons];       //指向孩子节点
} TSonNode;
```

其中，MaxSons为最多的孩子节点个数。

思考题：

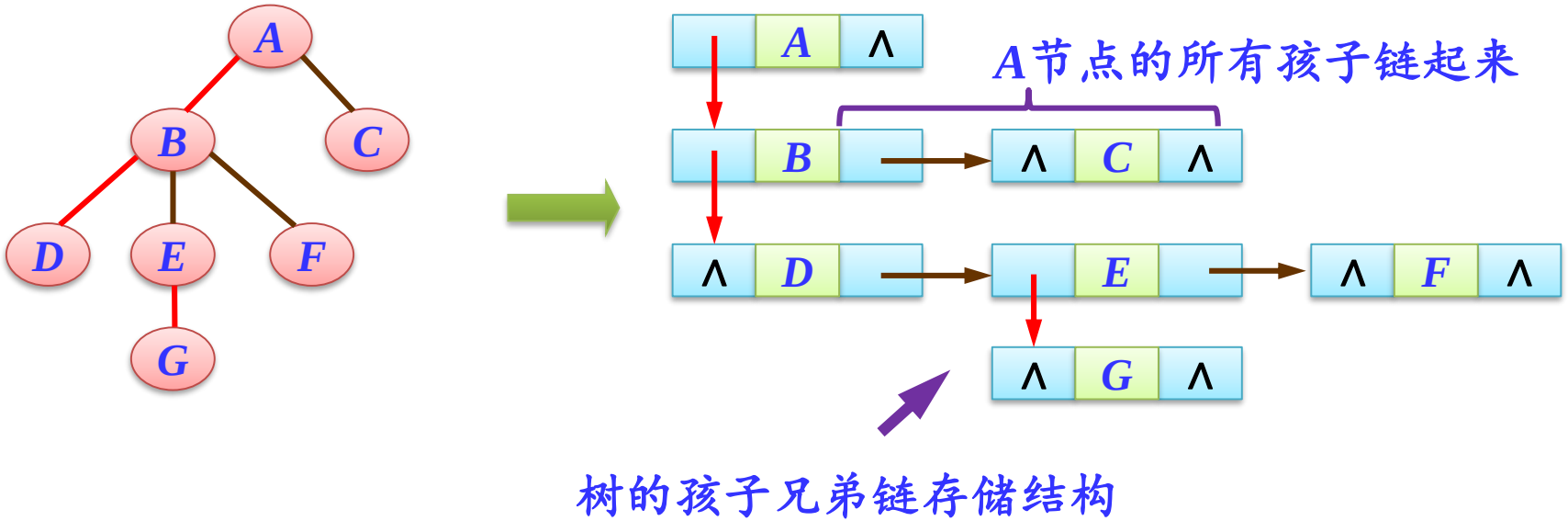
（1）*n*个节点的*m*次树有多少个空指针域？

（2）孩子链存储结构的优缺点？

3、孩子兄弟链存储结构

孩子兄弟链存储结构是为每个节点设计3个域：

- 一个数据元素域
- 第一个孩子节点指针域
- 一个兄弟节点指针域



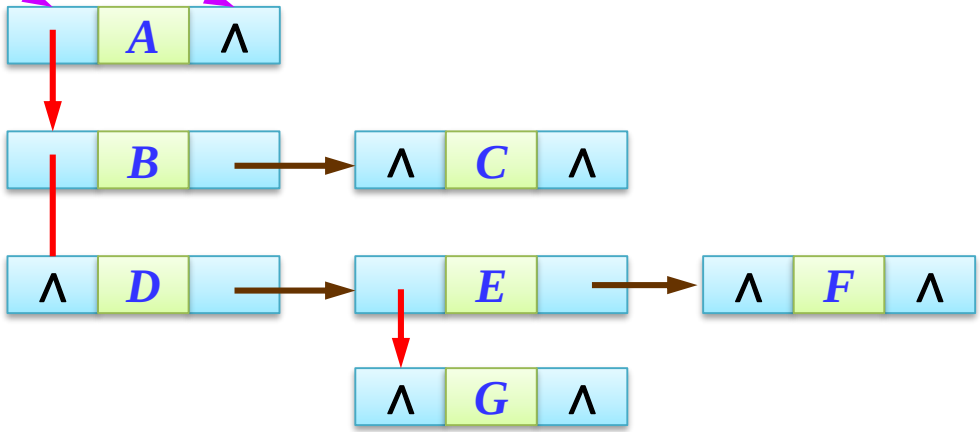
树的孩子兄弟链存储结构

兄弟链存储结构中节点的类型声明如下：

```
typedef struct tnode
{     ElemType data;          //节点的值
      struct tnode *hp;       //指向兄弟
      struct tnode *vp;       //指向孩子节点
} TSBNode;
```

每个节点固定只有两个指针域！！！

思考题： 该存储结构的优缺点？



——本讲完——