树与森林

学习目标

掌握树的存储结构

熟知树、森林与二叉树的转化

熟知树和森林的遍历

树的存储结构

树的存储

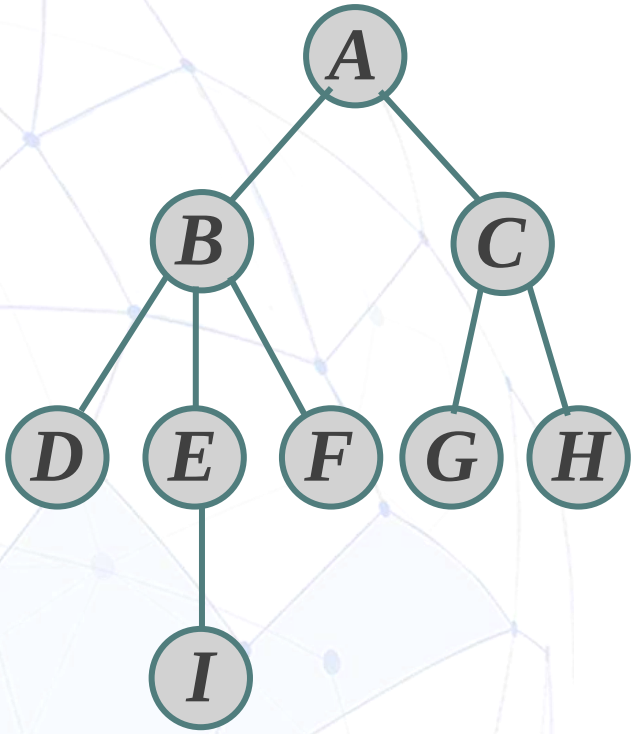
- 实现树的存储结构，关键是什么？

  如何表示树中结点之间的逻辑关系

- 什么是存储结构？

  数据元素及其逻辑关系在存储器中的表示

- 树中结点之间的逻辑关系是什么？

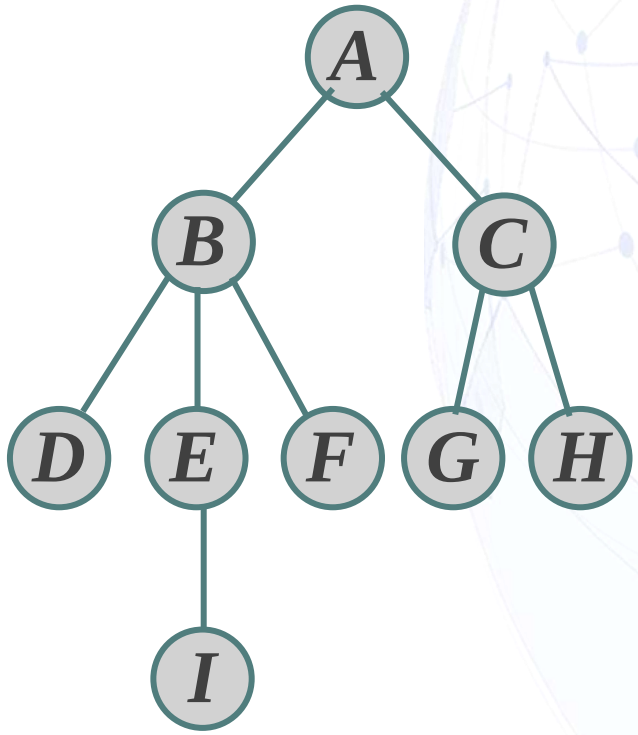


思考问题的出发点：如何表示结点的双亲和孩子

- 树的双亲表示法
- 树的孩子表示法
- 树的孩子兄弟表示法

双亲表示法

树的双亲表示法：用一维数组存储树中各个结点（一般按层序存储）的数据信息以及该结点的双亲在数组中的下标




```
template <typename DataType>
struct PNode
{
    DataType data;
    int parent;
};
```

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

双亲表示法

如何定义树的双亲表示法?

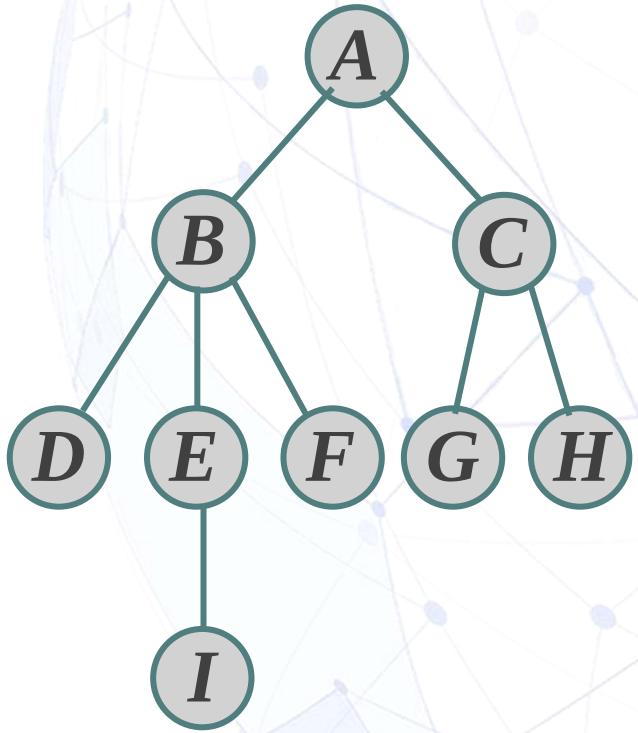
InitTree：初始化一棵树
DestroyTree：销毁一棵树
PreOrder：前序遍历树
PostOrder：后序遍历树
LeverOrder：层序遍历树

```
template <typename DataType>
struct PNode
{
    DataType data;
    int parent;
};
```

```
const int MaxSize = 100;
template <typename DataType>
class PTree
{
public:
    PTree( );
   ~PTree( );
    PreOrder( );
    PostOrder( );
    LeverOrder( );
private:
    PNode tree[MaxSize];
    int treeNum;
};
```

双亲表示法

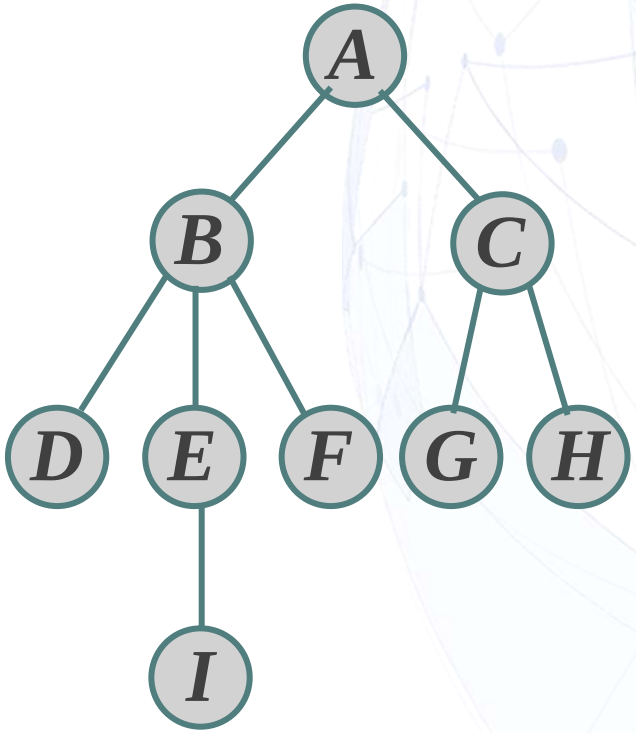
- 如何查找双亲结点？时间性能？ $O(1)$
- 如何查找孩子结点？时间性能？ $O(n)$
- 如何快速查找孩子结点？ 增设firstChild域



	data	parent	firstChild
0	A	−1	1
1	B	0	3
2	C	0	6
3	D	1	−1
4	E	1	8
5	F	1	−1
6	G	2	−1
7	H	2	−1
8	I	4	−1

孩子表示法

如何表示结点的孩子呢？ 方案一：指针域的个数等于树的度



data	child1	child2	……	childd

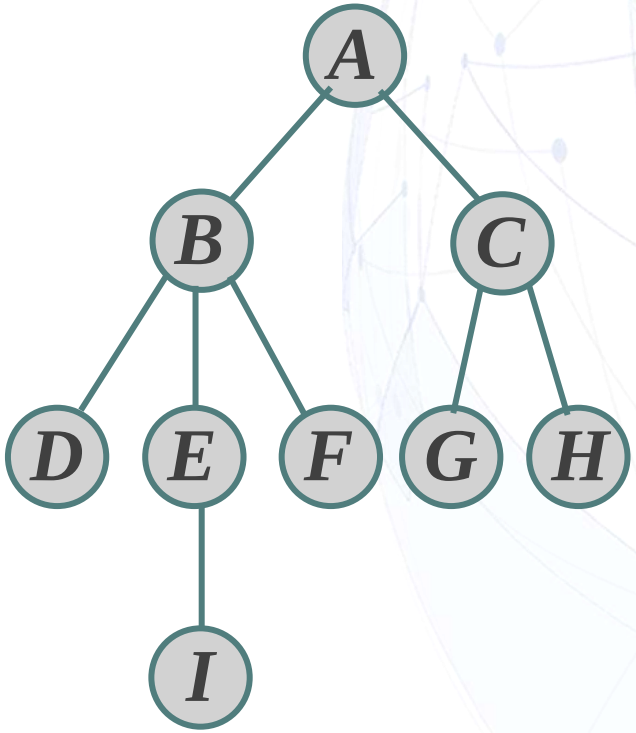
其中：data：数据域，存放该结点的数据信息

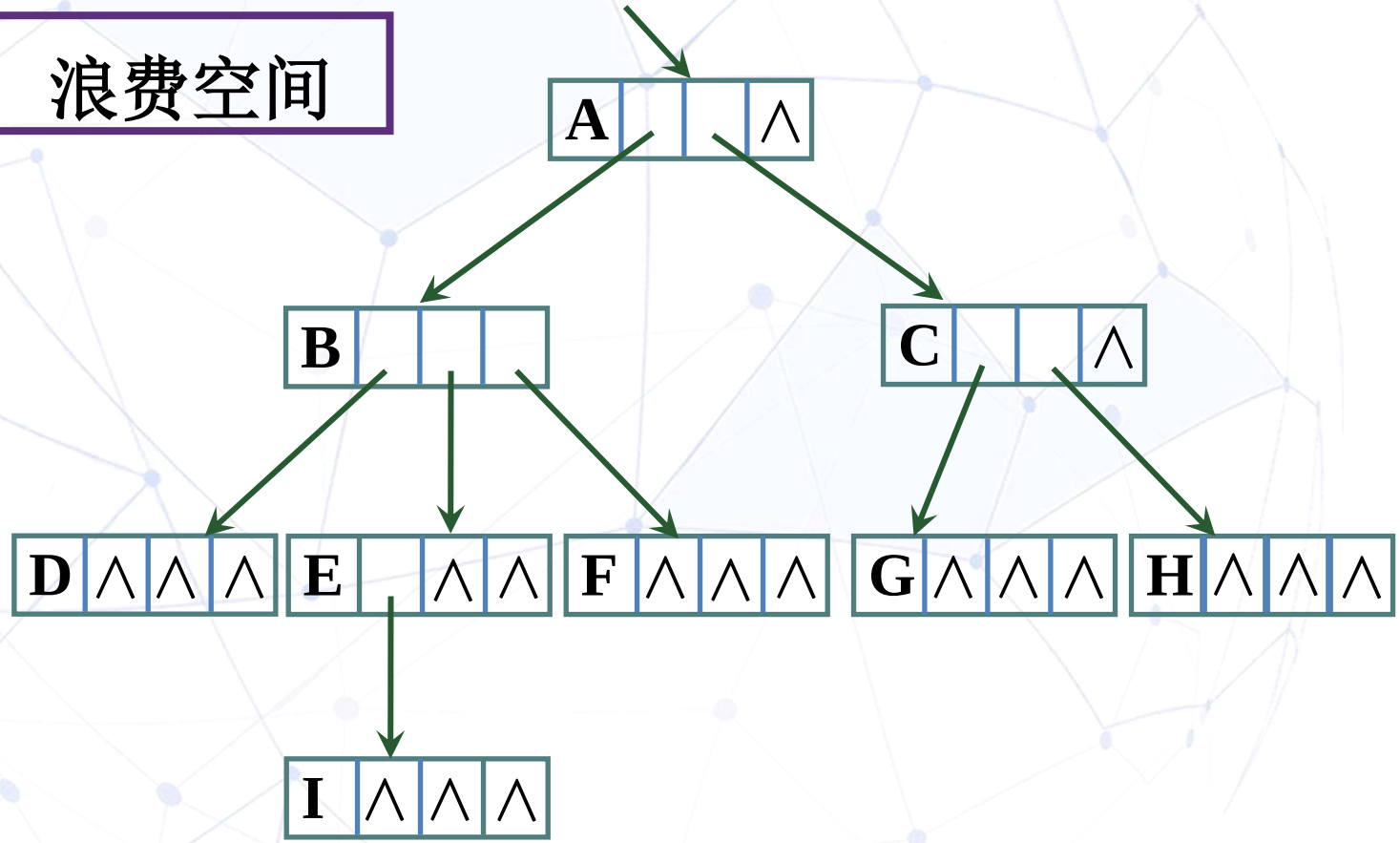
child1~childd：指针域，指向该结点的孩子

孩子表示法

如何表示结点的孩子呢？ 方案一：指针域的个数等于树的度

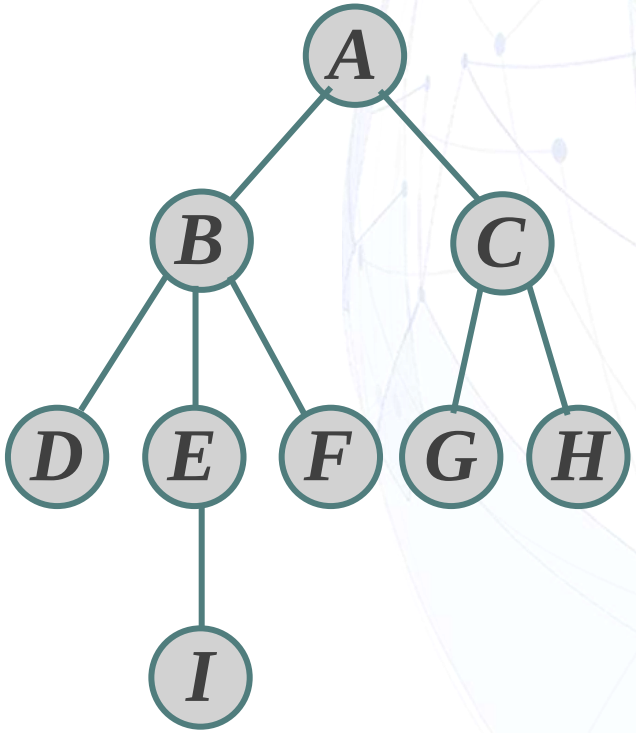
缺点：浪费空间





孩子表示法

如何表示结点的孩子呢？ 方案二：指针域的个数等于该结点的度



data	degree	child1	child2	……	childd

其中：data：数据域，存放该结点的数据信息

degree：数据域，存放该结点的度

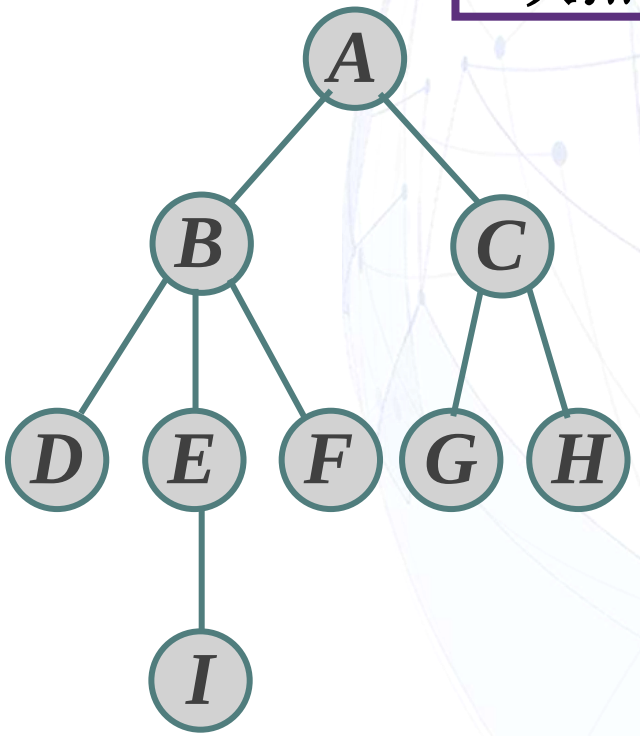
child1~childd：指针域，指向该结点的孩子

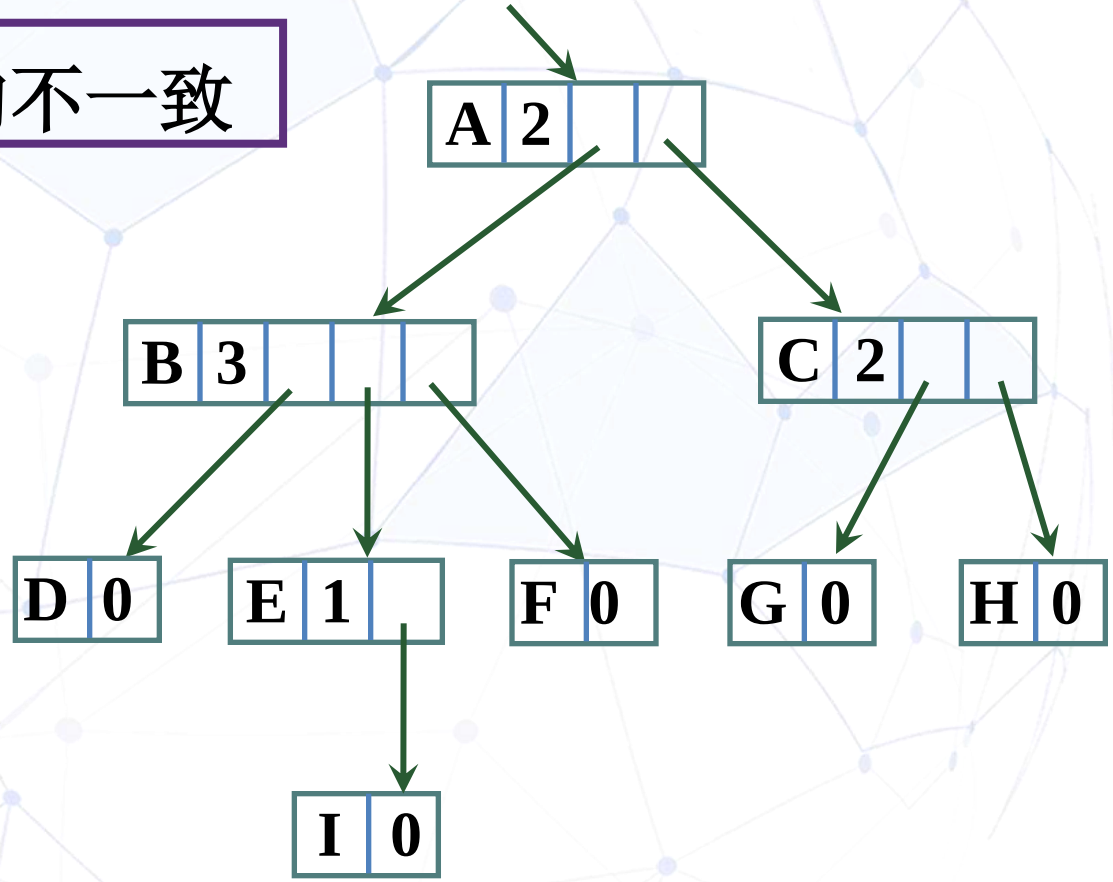
代表：二叉树的二叉链表

孩子表示法

如何表示结点的孩子呢？ 方案二：指针域的个数等于该结点的度

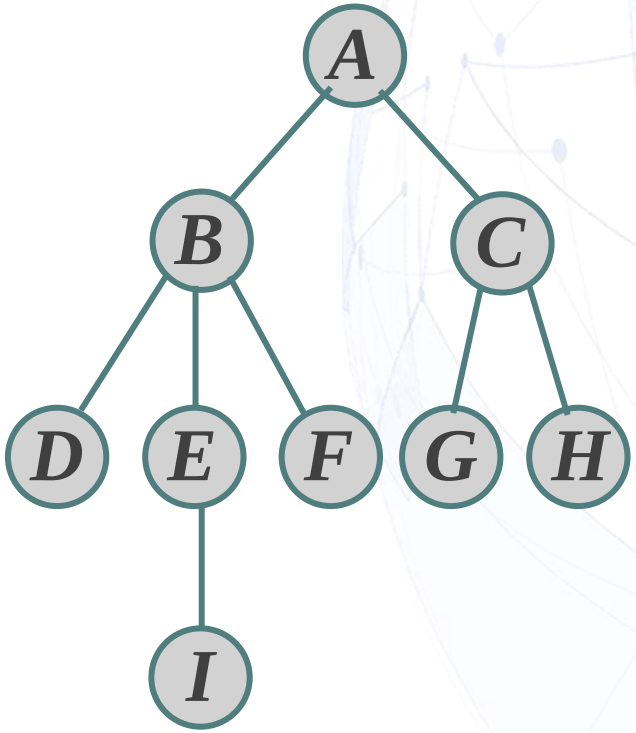
缺点：结点结构不一致

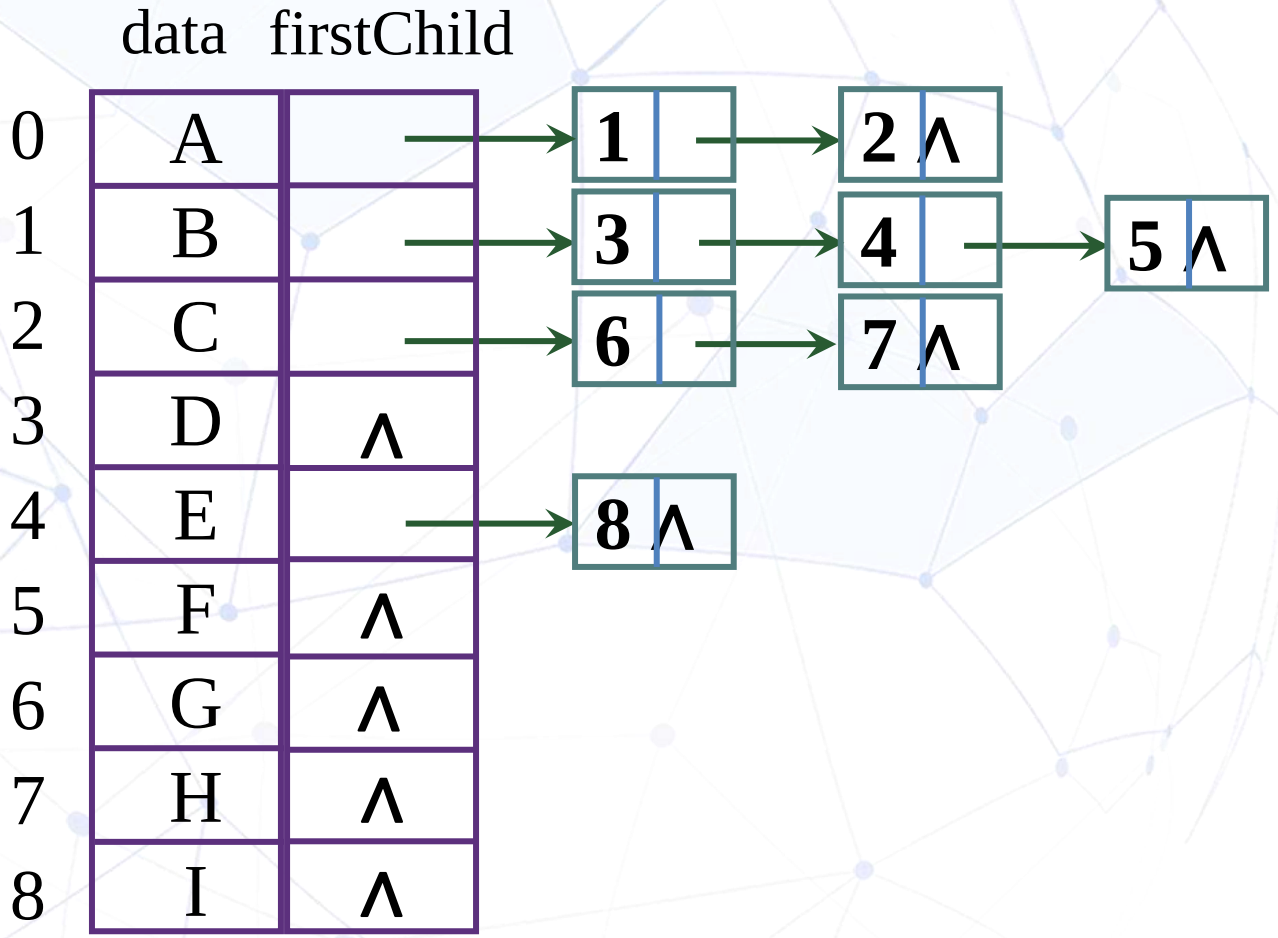




孩子表示法

如何表示结点的孩子呢？ 将结点的所有孩子构成一个单链表





孩子表示法

如何定义树的孩子表示法呢?

child	next

```
struct ChildNode
{
    int child;
    ChildNode *next;
};
```

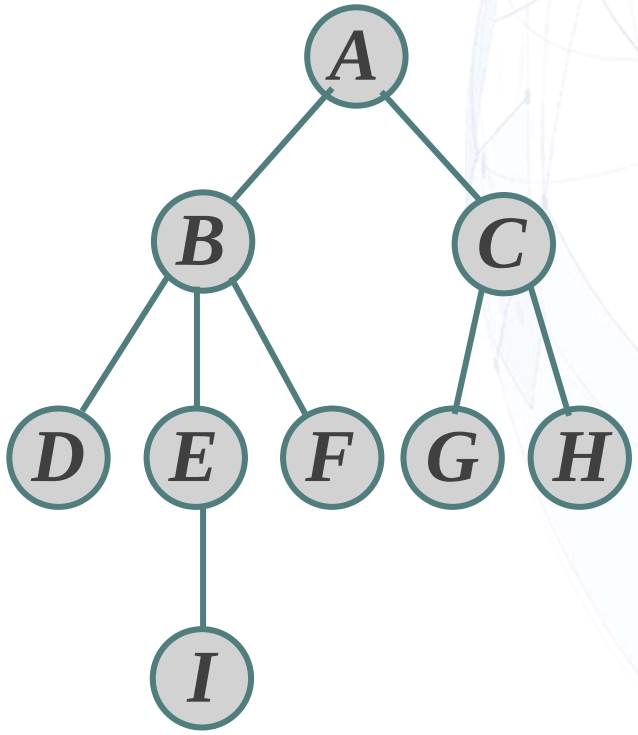
data	firstChild

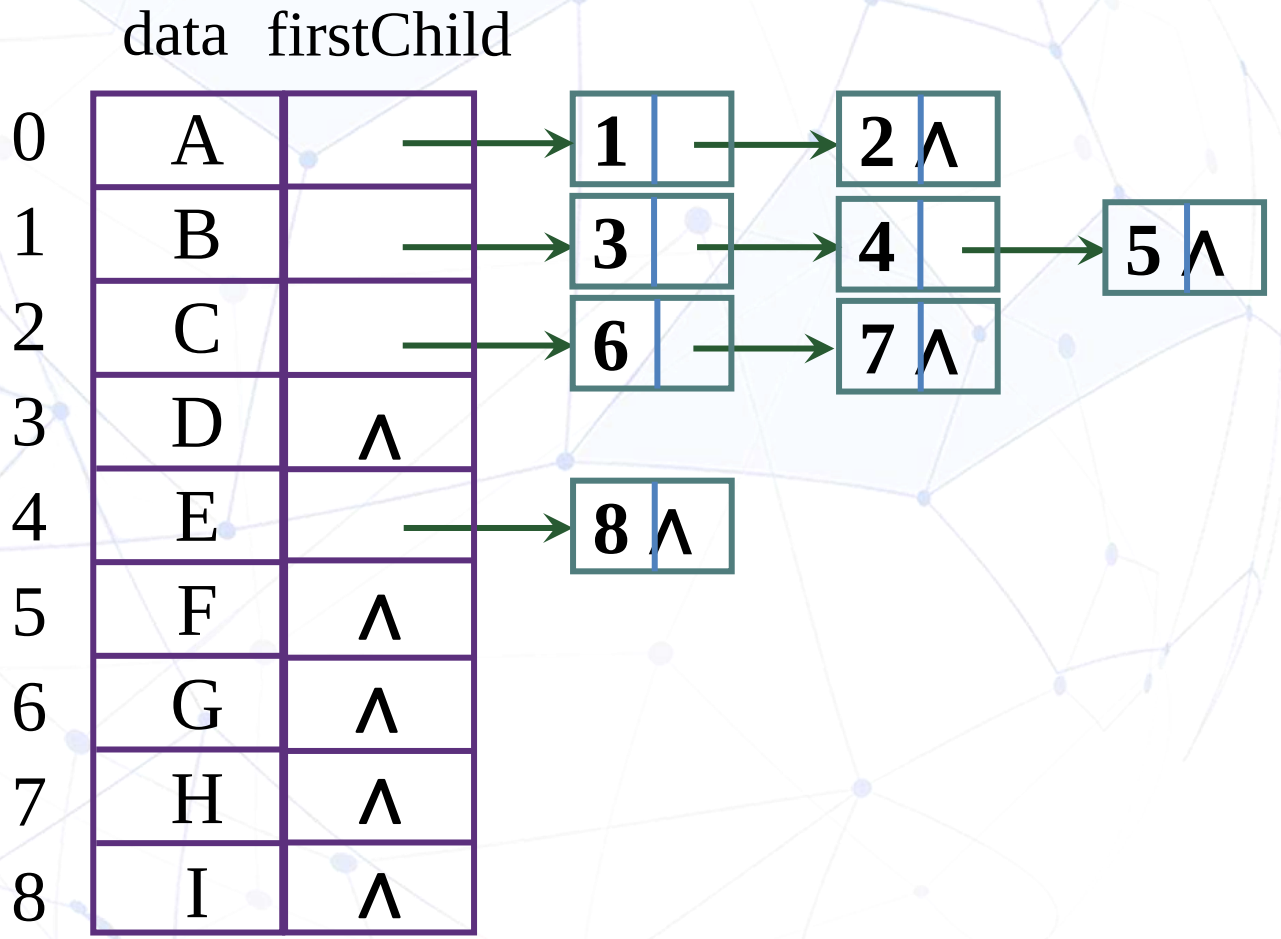
```
template <typename DataType>
struct TreeNode
{
    DataType  data;
    ChildNode *firstChild;
};
```

```
const int MaxSize = 100;
template <typename DataType>
class CTree
{
public:
     PTree( );
    ~PTree( );
     PreOrder( );
     PostOrder( );
     LeverOrder( );
private:
    TreeNode tree[MaxSize];
    int treeNum;
};
```

孩子表示法

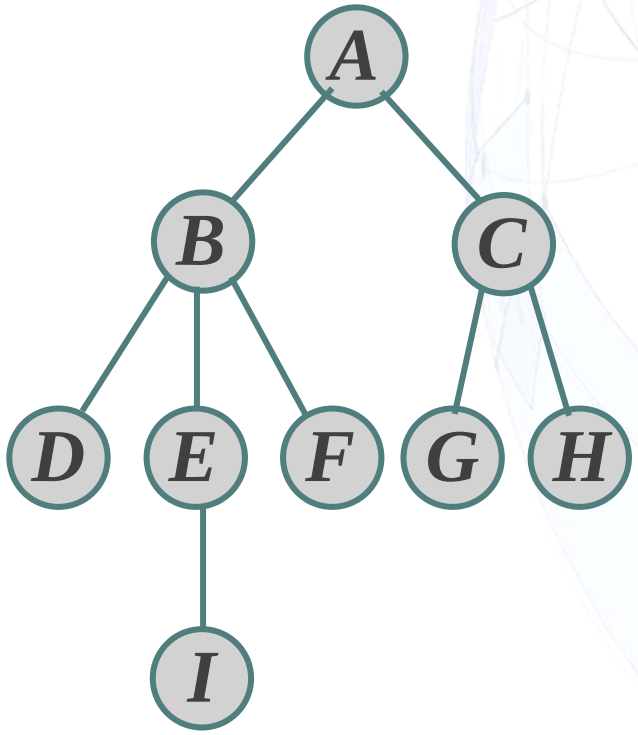
- 如何查找孩子结点？时间性能？ $O(1)$
- 如何查找双亲结点？时间性能？ $O(n)$
- 如何快速查找双亲？



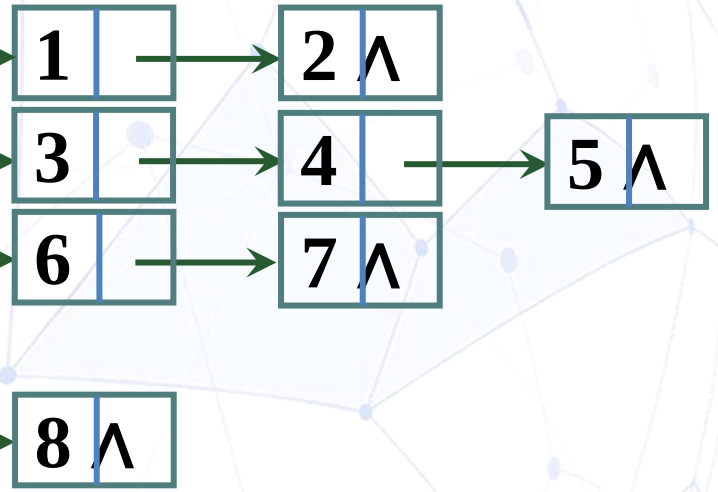


孩子表示法

- 如何查找孩子结点？时间性能？ $O(1)$
- 如何查找双亲结点？时间性能？ $O(n)$
- 如何快速查找双亲？

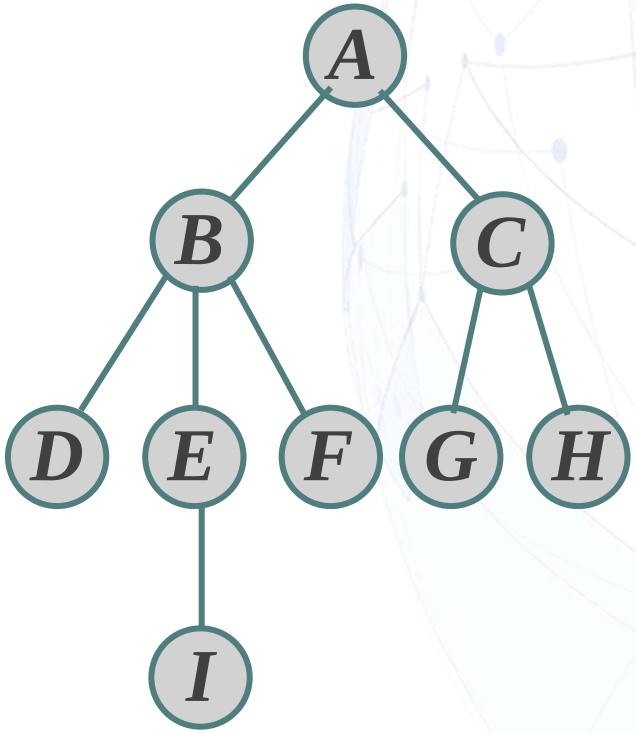


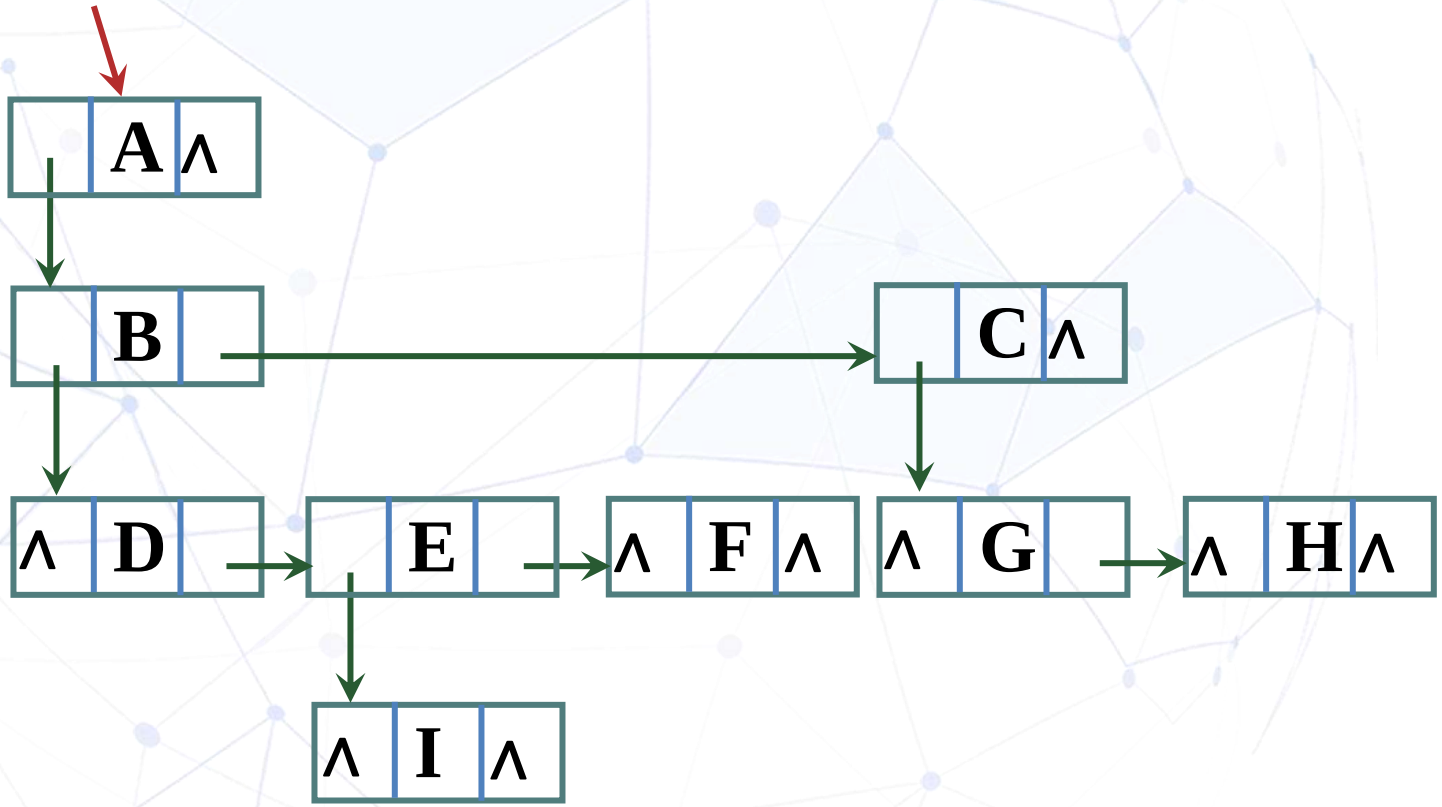
	data	parent	firstChild
0	A	−1	
1	B	0	
2	C	0	
3	D	1	∧
4	E	1	
5	F	1	∧
6	G	2	∧
7	H	2	∧
8	I	4	∧



孩子兄弟表示法

- 树的孩子兄弟表示法（二叉链表）：链表中的每个结点包括数据域和分别指向该结点的第一个孩子和右兄弟的指针





孩子兄弟表示法

如何定义树的孩子兄弟存储结构?

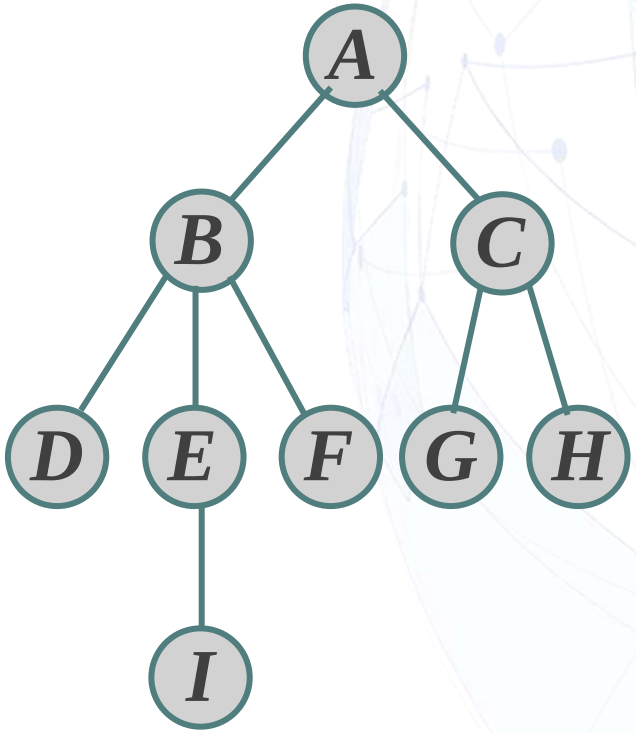
firstchild	data	rightsib

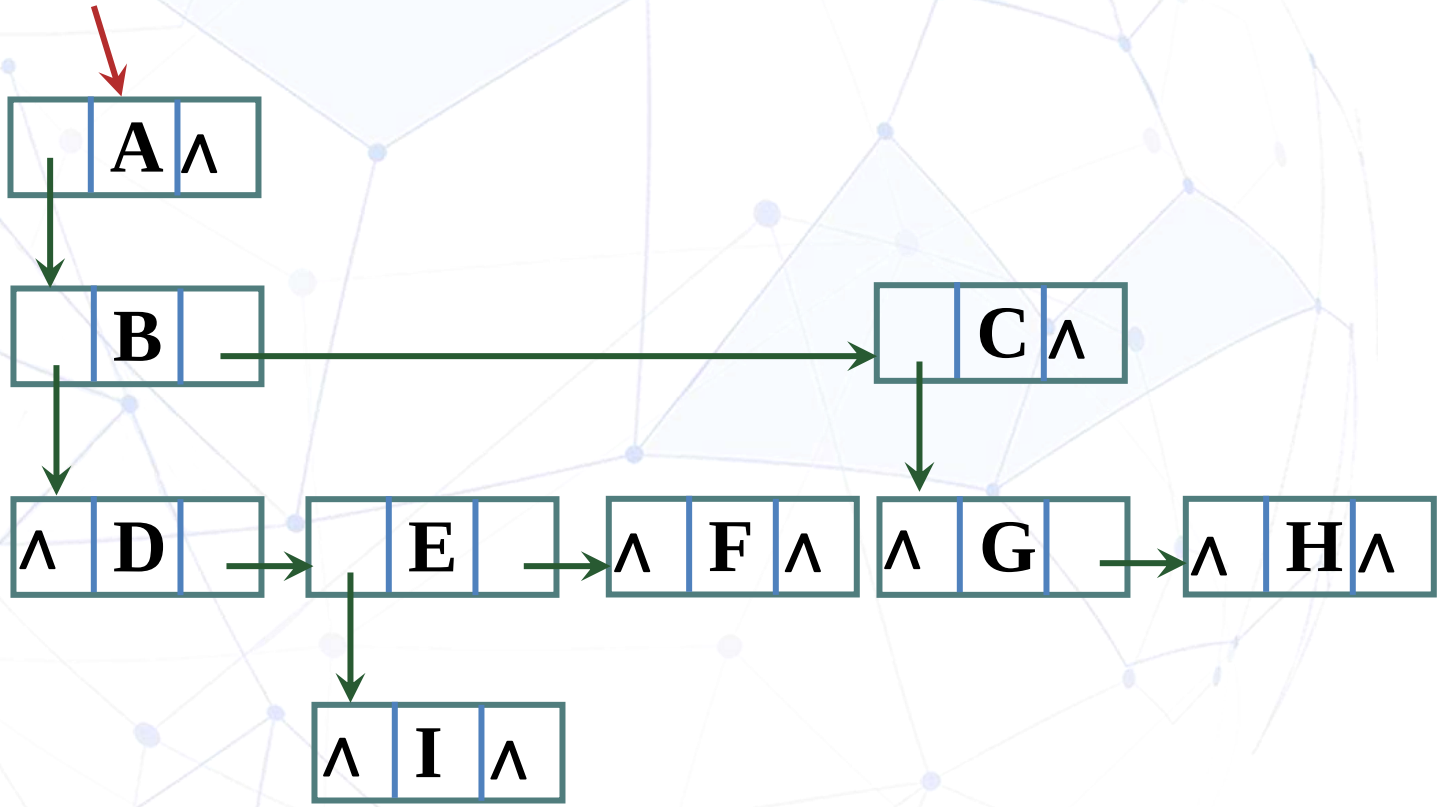
```
template <typename DataType>
struct CSNode
{
    DataType data;
    CSNode *firstchild, *rightsib;
} ;
```

```
template <typename DataType>
class CTree
{
public:
    PTree( );
   ~PTree( );
    PreOrder( );
    PostOrder( );
    LeverOrder( );
private:
    CSNode *root;
};
```

孩子兄弟表示法

- 如何查找兄弟结点？时间性能？ $O(1)$
- 如何查找孩子结点？时间性能？ $O(n)$ ⇨ 已知该结点指针： $O(1)$





孩子兄弟表示法是树与二叉树转换的桥梁

单选题 5分

设置

1. 实现树的存储结构，关键是如何存储结点的双亲或孩子。

A 正确

B 错误

提交

单选题 5分

设置

2. 树的双亲表示法可以按任意次序存储结点的数据信息。

A 正确

B 错误

提交

单选题 5分

设置

5. 树的孩子兄弟表示法是一种链式存储结构。

A 正确

B 错误

提交

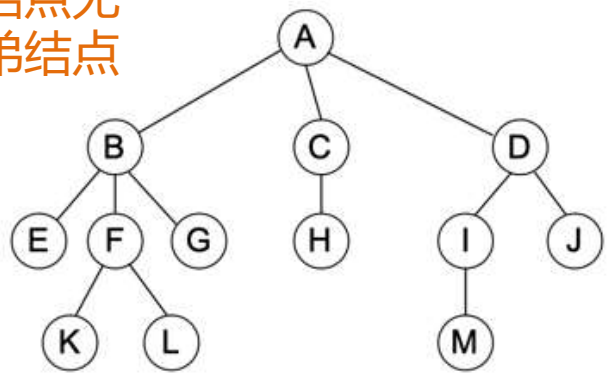
树与二叉树的转换

树与二叉树的转换

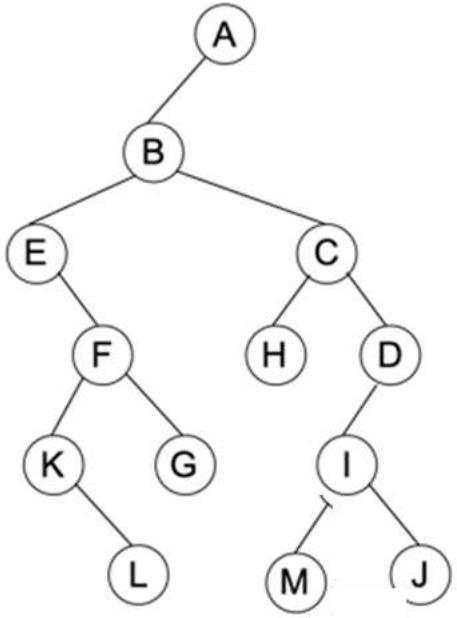
对任何一个树，存在唯一的一个二叉树与它对应，两者具有相同的二叉链表结构

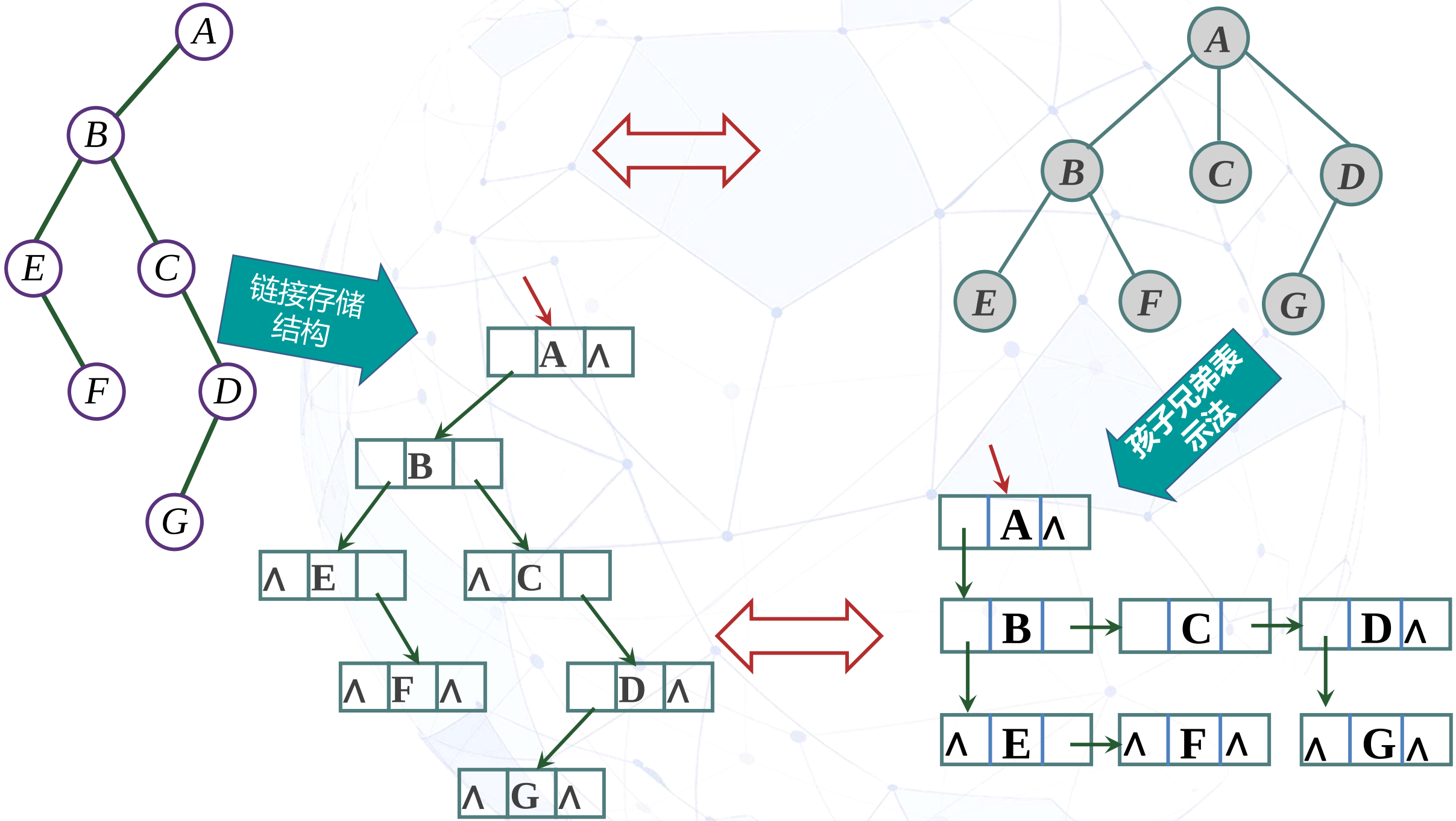
树

根结点无兄弟结点



二叉树




A
B
E
C
F
D
G
链接存储结构
A ∧
B
∧ E
∧ C
∧ F ∧
D ∧
∧ G ∧
A
B
C
D
E
F
G
孩子兄弟表示法
A ∧
B
C
D ∧
∧ E
∧ F ∧
∧ G ∧

树与二叉树的对应关系

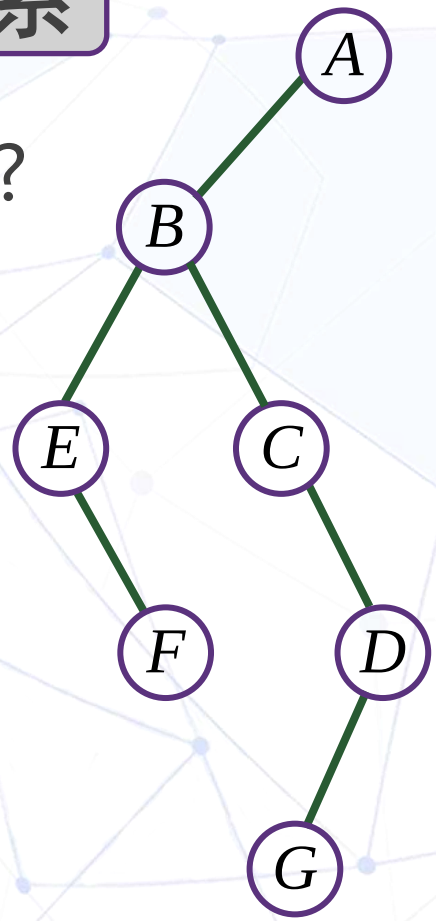
逻辑关系有什么变化?

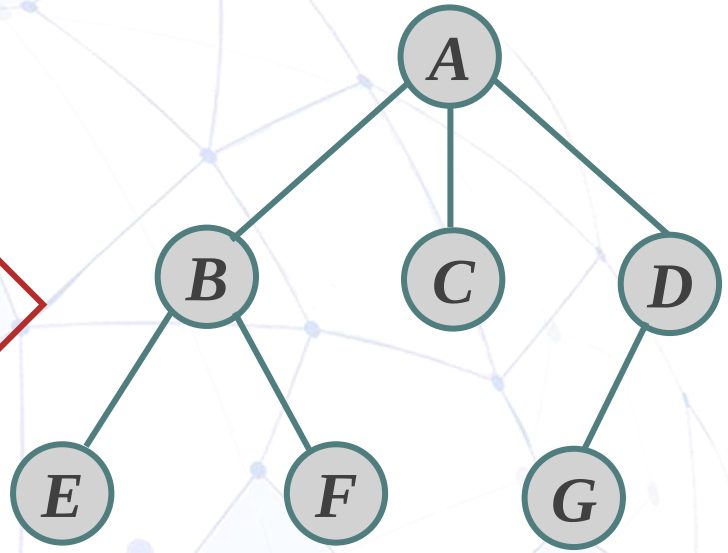
树：兄弟关系

二叉树：双亲和右孩子

树：双亲和长子

二叉树：双亲和左孩子





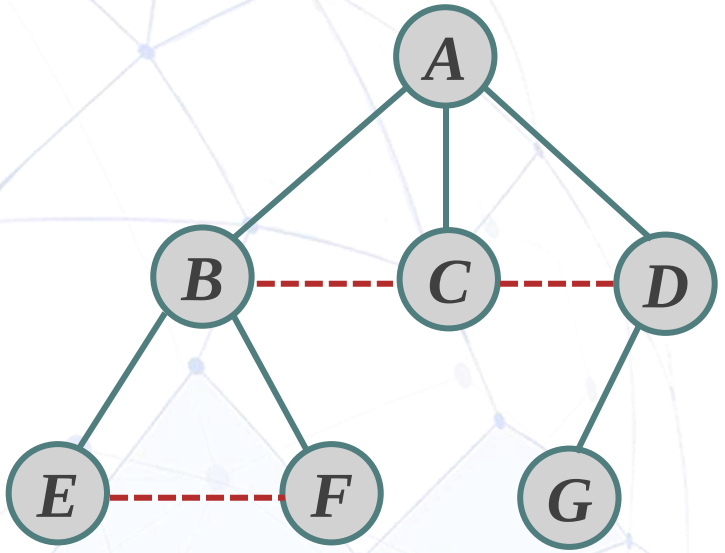
树转换为二叉树

将一棵树转换为二叉树的方法是

（1）加线——树中所有相邻兄弟之间加一条连线

（2）去线——对树中的每个结点，只保留它与第一个孩子结点之间的连线，删去它与其它孩子结点之间的连线。

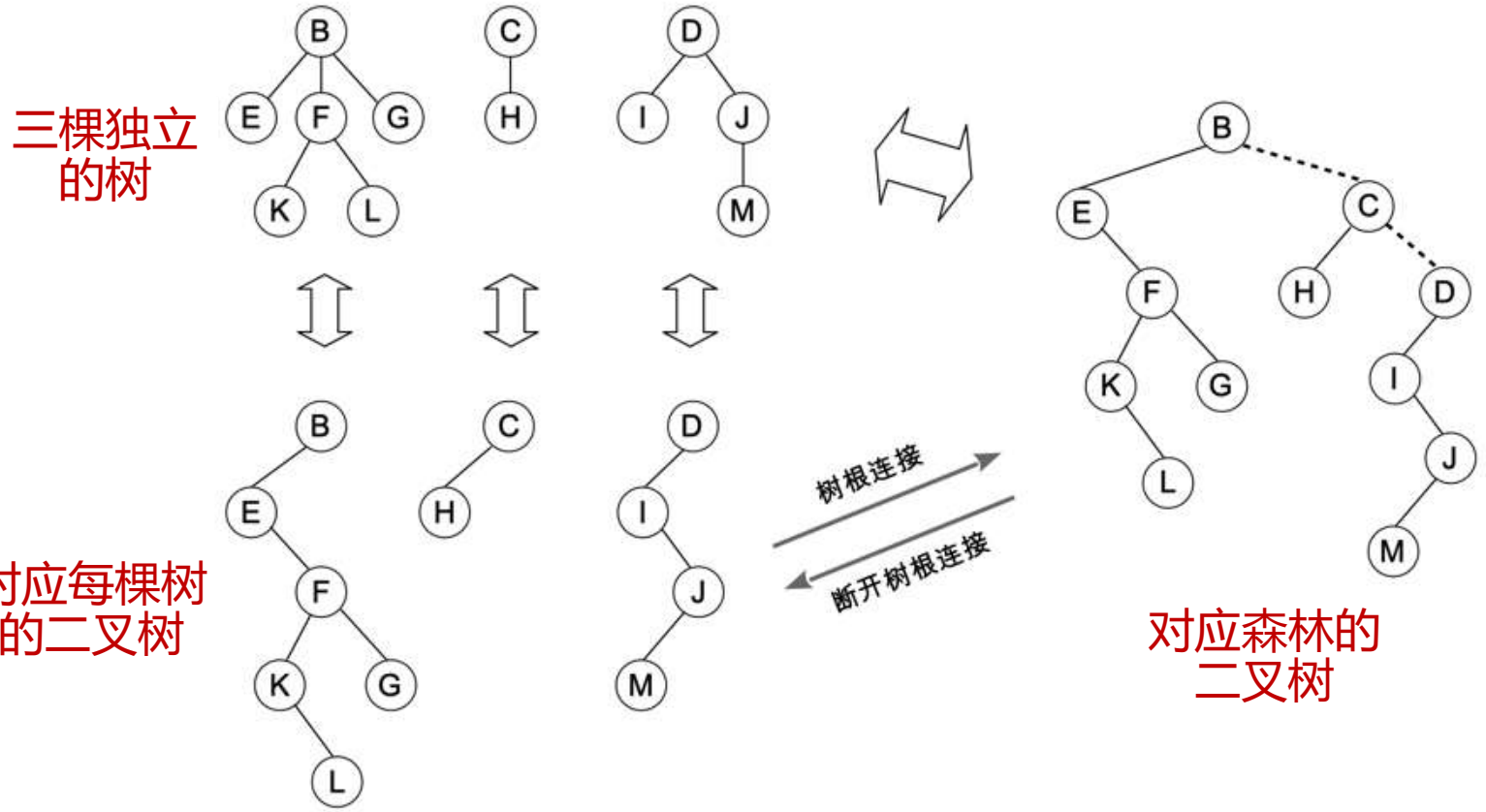
（3）层次调整——以根结点为轴心，将树顺时针转动一定的角度，使之层次分明。



森林与二叉树的转换

对于每一棵独立的树，由于根结点没有兄弟，它对应的二叉树的根没有右子结点，即**右子树为空**

利用右子树的链将树串联起来，建立**森林与二叉树的对应关系**



森林转换成二叉树

(1) 把森林中的每个树转换为二叉树

(2) 把森林中第一个二叉树的根结点作为转换后的二叉树的根，从第二个二叉树开始，把每个二叉树的根作为前一个二叉树的根的右子结点

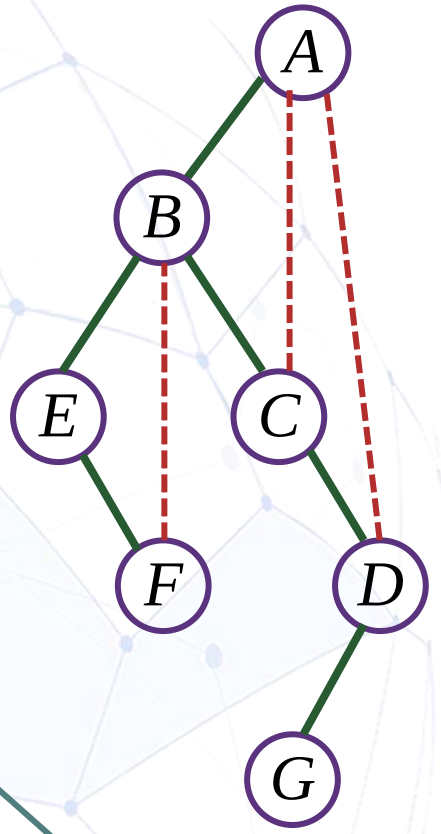
二叉树转换为树（森林）

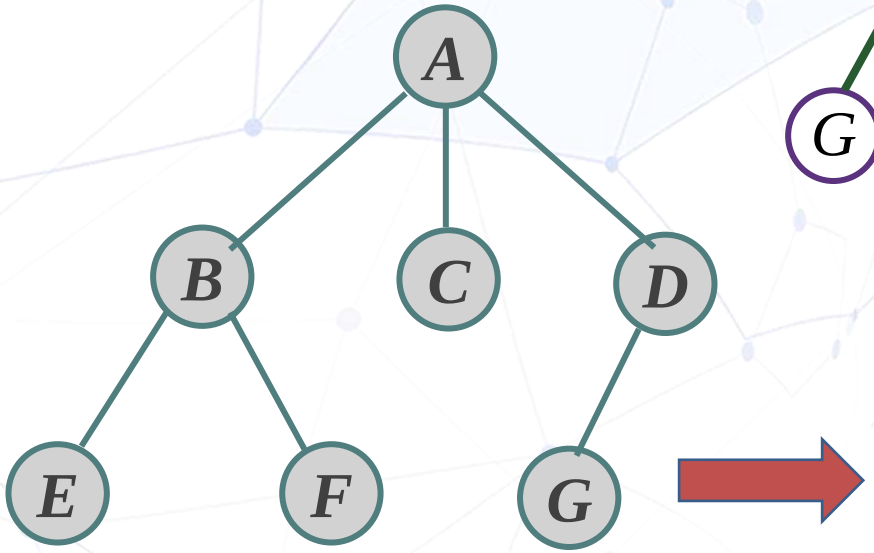
将一棵二叉树还原为树或森林，具体转换方法是

（1）加线——若某结点 x 是其双亲 y 的左孩子，则把结点 x 的右孩子、右孩子的右孩子、……，与结点 y 连线

（2）去线——删去所有双亲结点与右孩子结点的连线

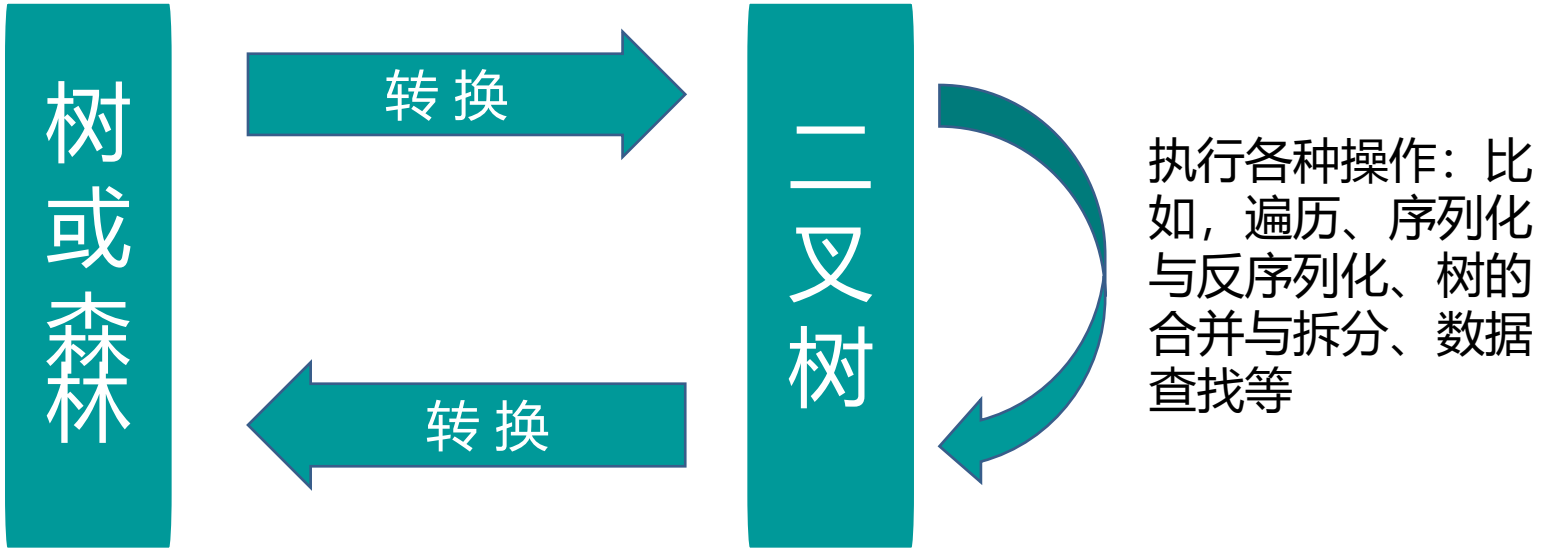
（3）层次调整——整理由（1）、（2）两步所得到的树（森林），使之层次分明。





树、森林与二叉树的转换

树、森林与二叉树的对应关系表明可以把树或森林先转换为二叉树，使用二叉树的各种操作进行处理，处理结束后还可以再转换回原来的树或森林



单选题 5分

设置

1. 由树转换成二叉树，其根结点的右子树总是空的。

A 正确

B 错误

提交

单选题 5分

设置

2. 树转换为二叉树后，树中的双亲和长子关系在二叉树中是（ ）关系。

A 双亲和左孩子

B 双亲和右孩子

C 兄弟

D 祖先

提交

单选题 5分

设置

3. 树转换为二叉树后，树中的兄弟关系在二叉树中是（ ）关系。

A 双亲和左孩子

B 双亲和右孩子

C 兄弟

D 祖先

提交

单选题 5分

设置

5. 二叉树转换为树或森林，需要在二叉树中去掉所有双亲和右孩子的连线。

A 正确

B 错误

提交

单选题 5分

设置

6. 二叉树转换为森林，森林中树的个数取决于二叉树中最右分支上的结点个数。

A 正确

B 错误

提交

树与森林的遍历

树的遍历

树的遍历：从根结点出发，按照某种次序访问树中所有结点，并且每个结点仅被访问一次

前序（根）、后序（根）和层序（次）等

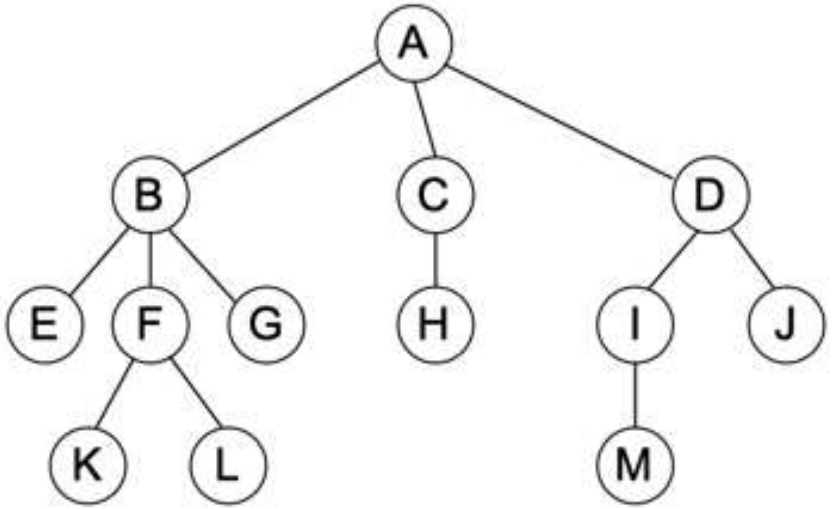
抽象操作，可以是对结点进行各种处理，这里简化为输出结点的数据

树的遍历

树的遍历方案：前序遍历、后序遍历（无中序遍历！）

前序遍历：先访问树根，然后对根的各子树从左向右依次进行前序遍历

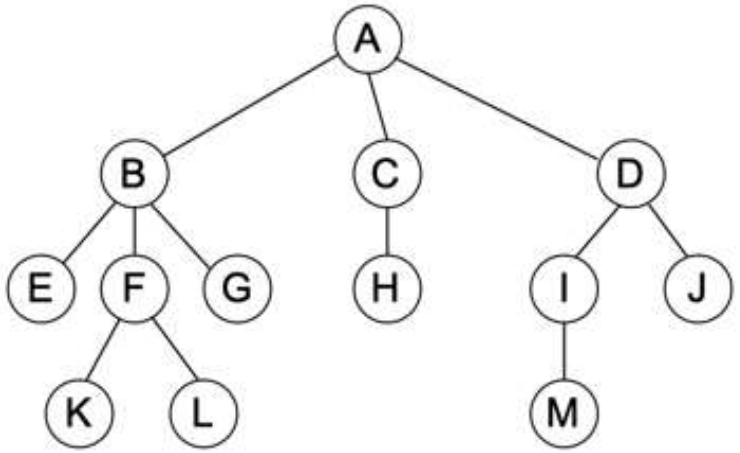
后序遍历：遍历根的各子树，最后访问根结点



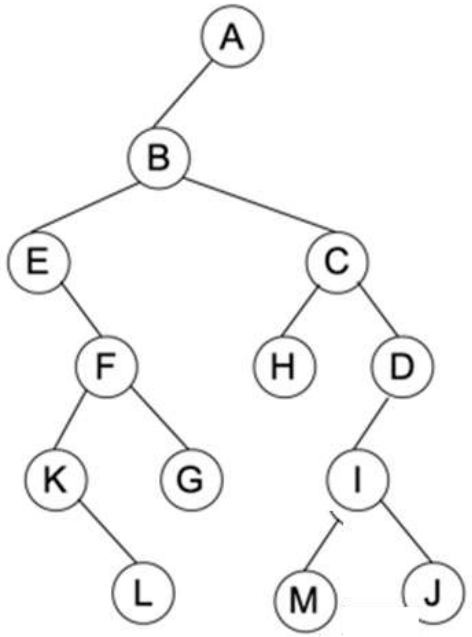
前序遍历：<A, B, E, F, K, L, G, C, H, D, I , M, J>

后序遍历：<E, K, L, F, G, B, H, C, M, I, J, D, A>

树与对应的二叉树的遍历



对应



前序 <A, B, E, F, K, L, G, C, H, D, I, M , J > = <A, B, E, F, K, L, G, C, H, D, I , M, J> 前序

树的前序遍历与对应的二叉树的前序遍历结果相同

后序 <E, K, L, F, G, B, H, C, M, I, J, D, A> = <E, K, L, F, G, B, H, C, M, I, J, D, A> 中序

树的后序遍历与对应的二叉树的中序遍历结果相同

基于二叉树遍历方案的树遍历算法

算法5-19. 前序遍历树 PreOrder(*tree*)

输入： 基于孩子兄弟表示法存储的树*tree*（二叉链表结构）
输出： 按前序遍历的顺序依次访问各结点

二叉树的前序遍历算法

```
if tree ≠ NIL then  //空树不做处理，直接返回
| Visit(tree)                    //先访问根结点
| PreOrder(tree.first_child)    //接下来访问tree所有子孙结点
| PreOrder(tree.next_sibling)  //最后访问tree后序的兄弟结点及其子孙
end
```

算法5-20. 后序遍历树 PostOrder(*tree*)

输入： 基于孩子兄弟表示法存储的树*tree*（二叉链表结构）
输出： 按后序遍历的顺序依次访问各结点

二叉树的中序遍历算法

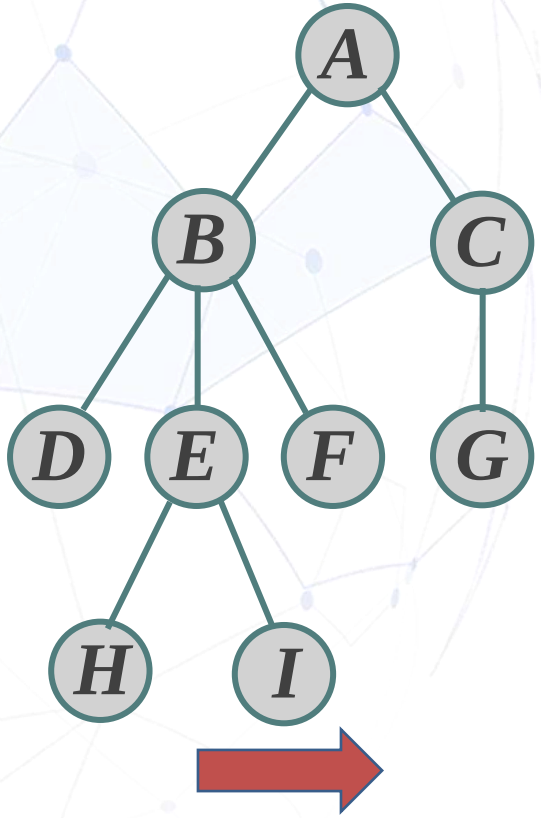
```
if tree ≠ NIL then  //空树不做处理，直接返回
| PostOrder(tree.first_child)    //先访问tree所有子孙结点
| Visit(tree)                     //接下来访问根结点
| PostOrder(tree.next_sibling)  //最后访问tree后序的兄弟结点及其子孙
end
```

树的层序遍历

树的层序遍历操作定义：

从树的根结点开始，自上而下逐层遍历，在同一层中，按从左到右的顺序对结点逐个访问

层序遍历序列：*A B C D E F G H I*



单选题 5分

设置

5. 在树的前序遍历序列中，任意一个结点均处在其子女的前面。

A 正确

B 错误

提交

单选题 5分

设置

6. 对于右图所示树，前序遍历序列是（ ）

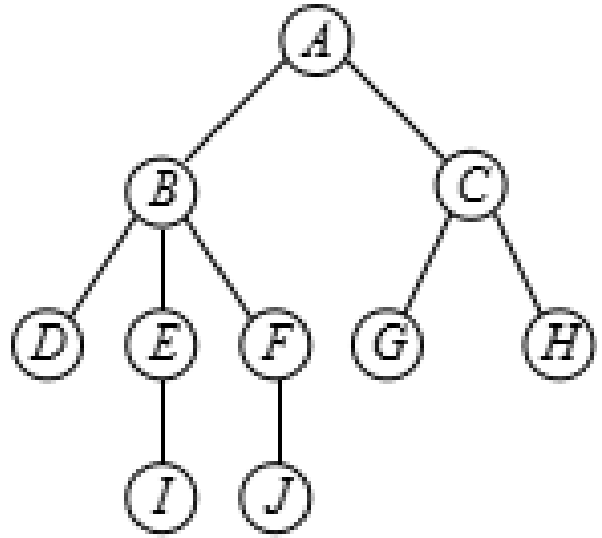


图 5-2 一棵树

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

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

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

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

提交

单选题 5分

设置

7. 对于右图所示树，后序遍历序列是（ ）

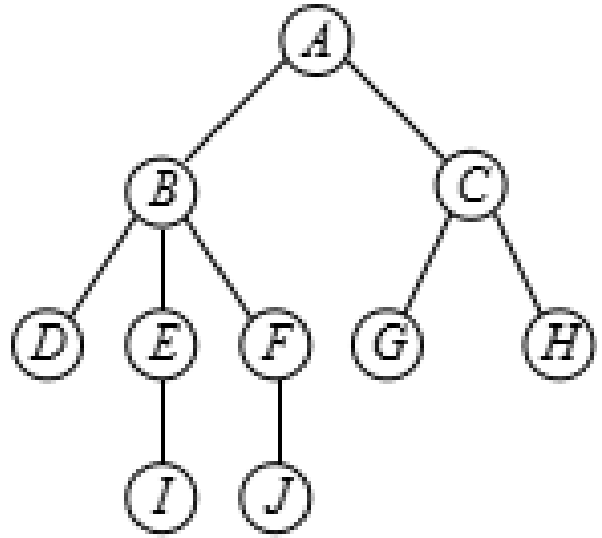


图 5-2 一棵树

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

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

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

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

提交

单选题 5分

设置

8. 对于右图所示树，层序遍历序列是（ ）

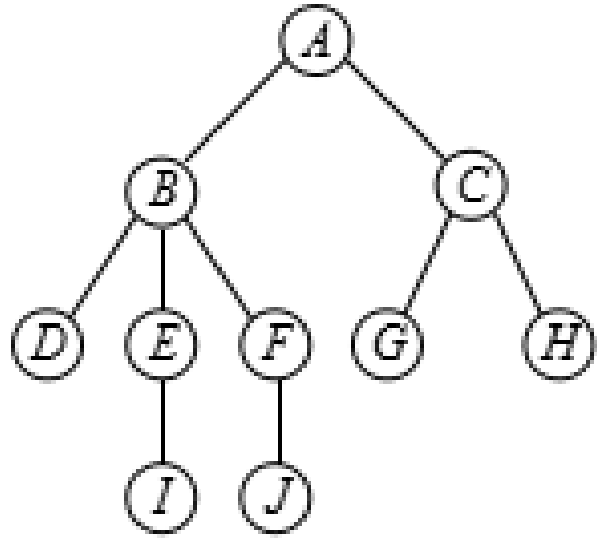


图 5-2 一棵树

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

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

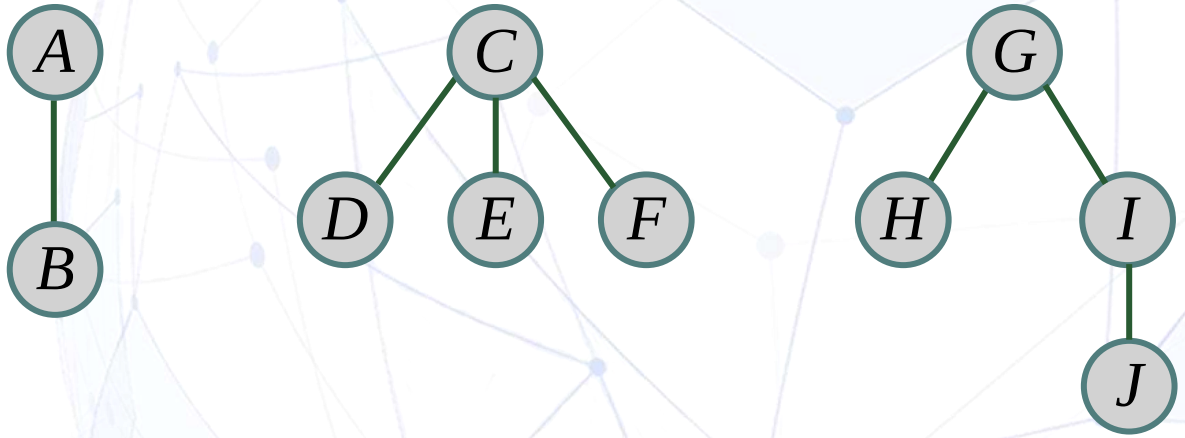
C (A, B, D, E, F, I, J, C, G, H)

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

提交

森林的定义

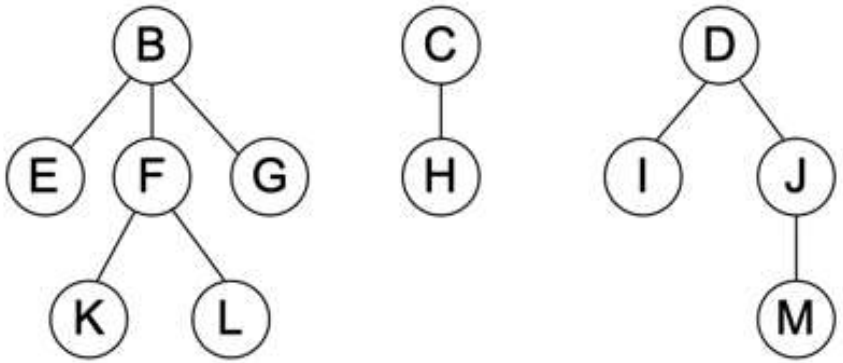
森林：m（$m \geq 0$）棵互不相交的树的集合



森林的遍历

前序遍历：从其中的第一个树开始，按序对每个树进行前序遍历

后序遍历：从其中的第一个树开始，按序对每个树进行后序遍历

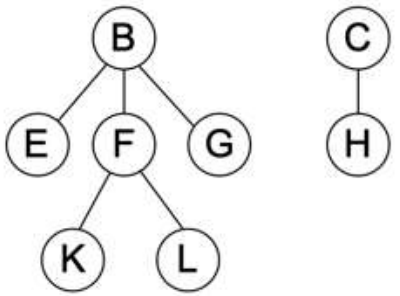


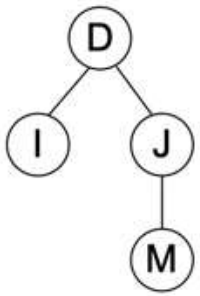
前序遍历：< B, E, F, K, L, G, C, H, D, I, J, M >

后序遍历：< E, K, L, F, G, B, H, C, I, M, J, D >

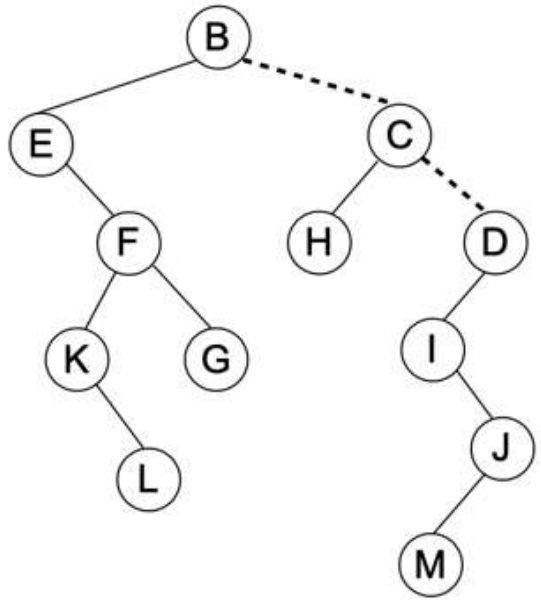
从左向右依次遍历每棵树！

森林与对应的二叉树的遍历





对应



前序 <B, E, F, K, L, G, C, H, D, I, J, M> = <B, E, F, K, L, G, C, H, D, I, J, M> 前序

森林的前序遍历与对应的二叉树的前序遍历结果相同 算法5.19可用

后序 <E, K, L, F, G, B, H, C, I, M, J, D> = <E, K, L, F, G, B, H, C, I, M, J, D> 中序

森林的后序遍历与对应的二叉树的中序遍历结果相同 算法5.20可用

单选题 5分

设置

树的前序遍历序列等价于二叉树的前序遍历序列，树的后序遍历序列等价于二叉树的后序遍历序列。

A 正确

B 错误

提交