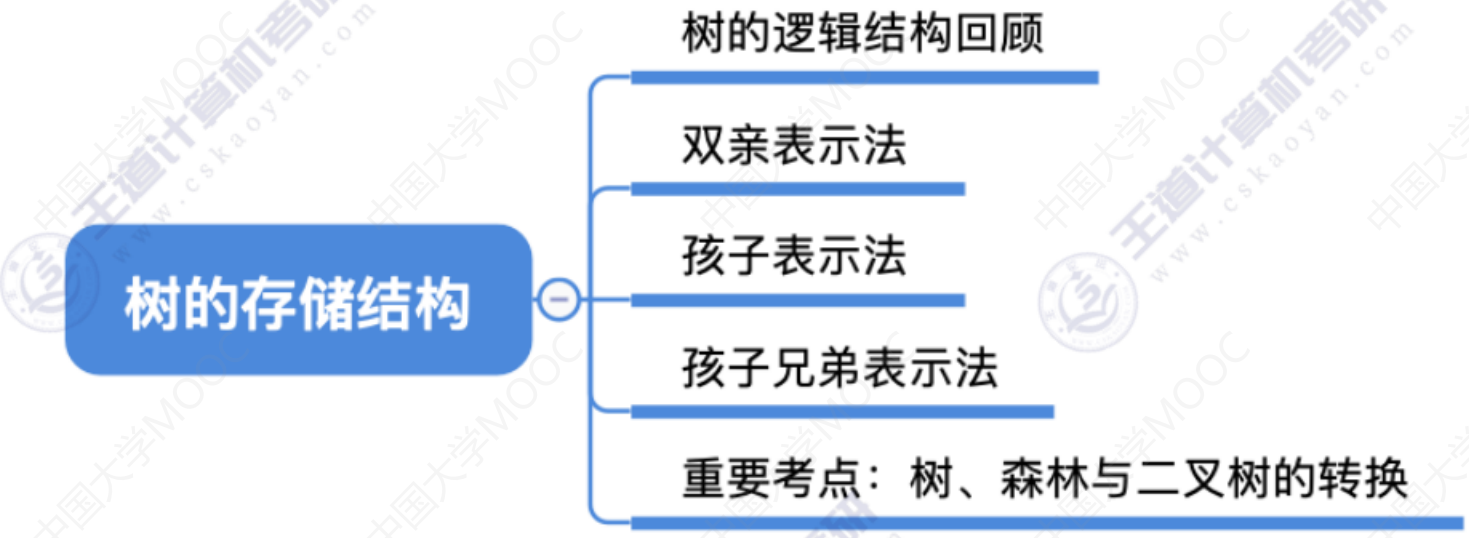
本节内容

树

存储结构

1

知识总览



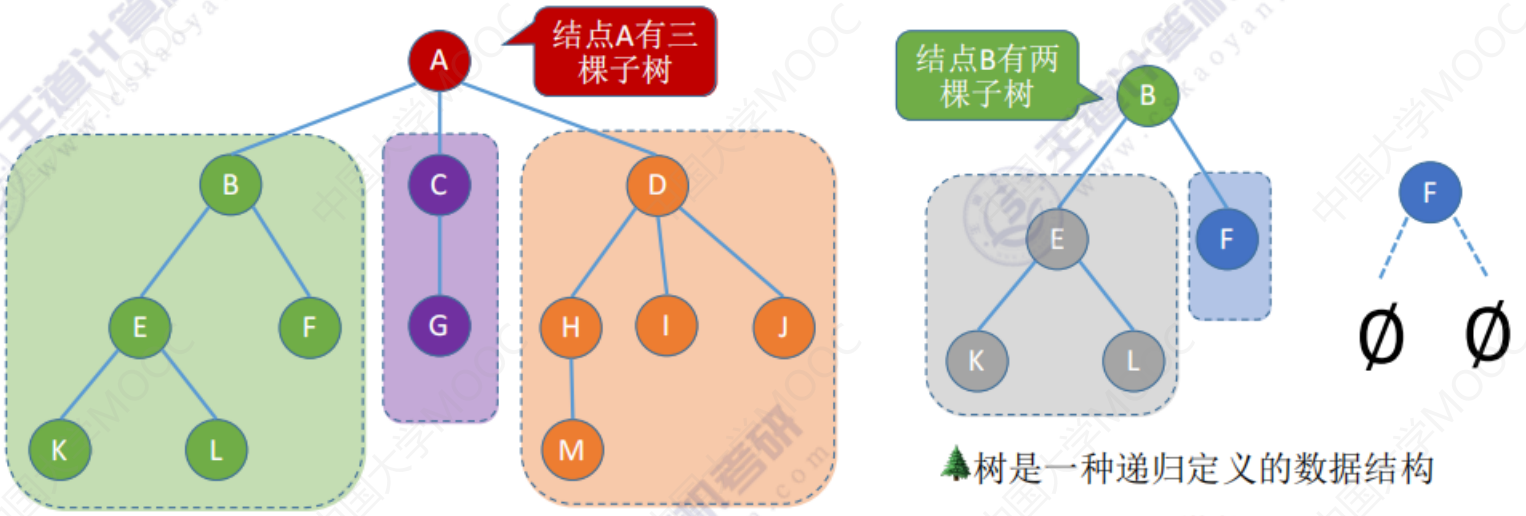
2

树的逻辑结构

树是n（$n \geq 0$）个结点的有限集合，$n = 0$时，称为空树，这是一种特殊情况。在任意一棵非空树中应满足：

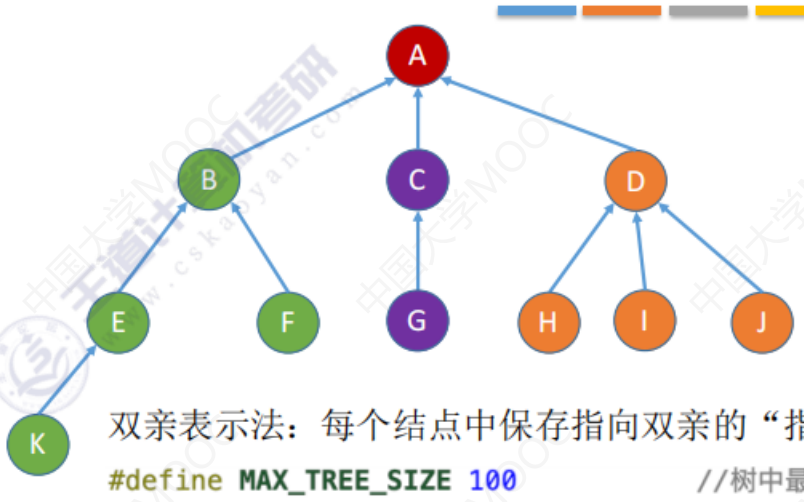
1）有且仅有一个特定的称为根的结点。

2）当$n > 1$时，其余结点可分为m（$m > 0$）个互不相交的有限集合$T_1, T_2, \ldots, T_m$，其中每个集合本身又是一棵树，并且称为根结点的子树。



🌲树是一种递归定义的数据结构

双亲表示法（顺序存储）



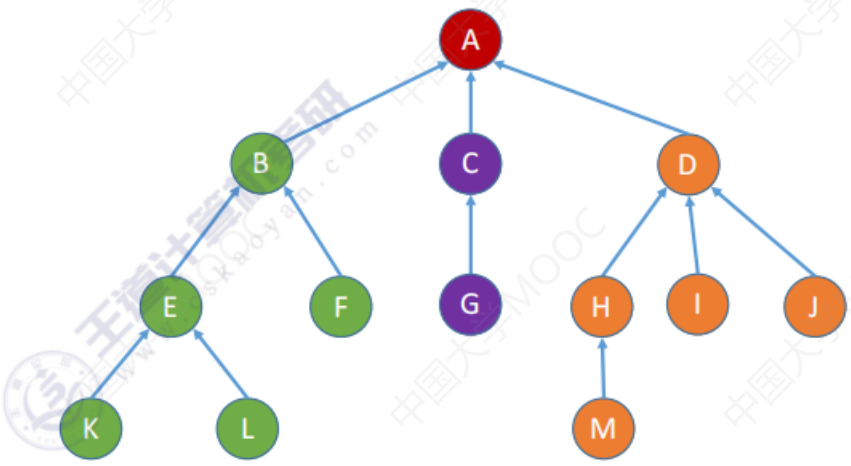
双亲表示法：每个结点中保存指向双亲的“指针”

```
#define MAX_TREE_SIZE 100                //树中最多结点数
typedef struct{                          //树的结点定义
    ElemType data;                       //数据元素
    int parent;                          //双亲位置域
}PTNode;
typedef struct{                          //树的类型定义
    PTNode nodes[MAX_TREE_SIZE];         //双亲表示
    int n;                               //结点数
}PTree;
```

	data	parent
0	A	-1
1	B	0
2	C	0
3	D	0
4	E	1
5	F	1
6	G	2
7	H	3
8	I	3
9	J	3
10	K	4
11		
12		
13		

根节点固定存储在0，-1表示没有双亲

双亲表示法（顺序存储）



双亲表示法：每个结点中保存指向双亲的“指针”

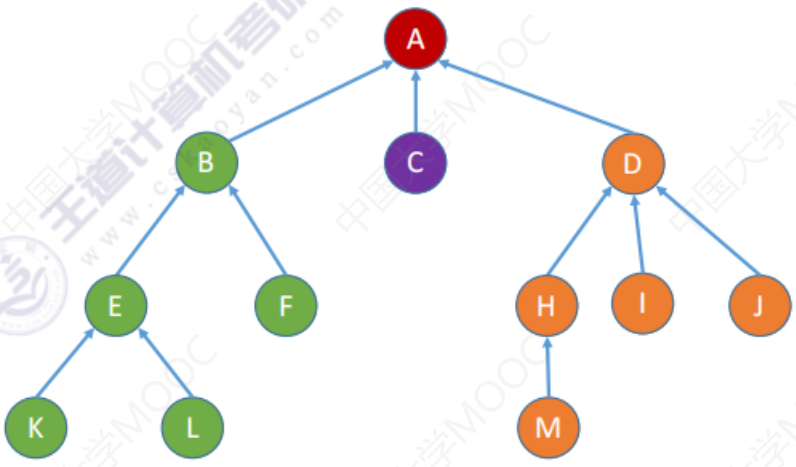
	data	parent
0	A	-1
1	B	0
2	C	0
3	D	0
4	E	1
5	F	1
6	G	2
7	H	3
8	I	3
9	J	3
10	K	4
11	M	7
12	L	4
13		

新增数据元素，无需按逻辑上的次序存储

王道考研/CSKAOYAN.COM

5

双亲表示法（顺序存储）



双亲表示法：每个结点中保存指向双亲的“指针”

	data	parent
0	A	-1
1	B	0
2	C	0
3	D	0
4	E	1
5	F	1
6		-1
7	H	3
8	I	3
9	J	3
10	K	4
11	M	7
12	L	4
13		

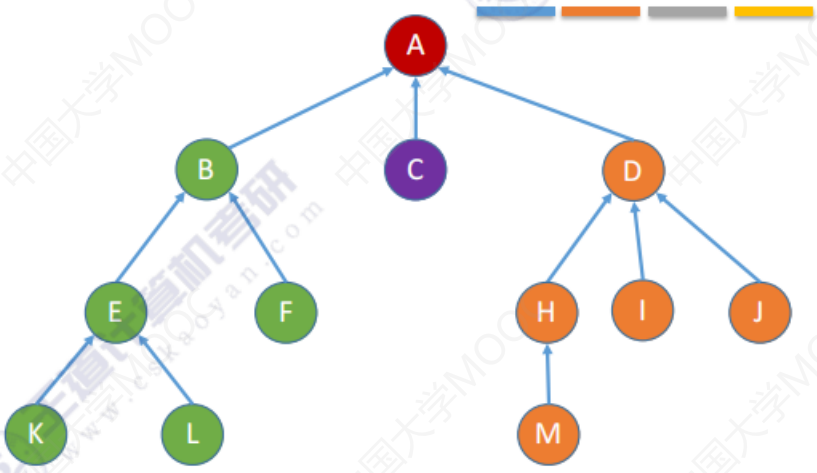
删除数据元素（方案一）

新增数据元素，无需按逻辑上的次序存储

王道考研/CSKAOYAN.COM

6

双亲表示法（顺序存储）



双亲表示法：每个结点中保存指向双亲的“指针”

```
typedef struct{
    PTNode nodes[MAX_TREE_SIZE];
    int n;
}PTree;
```

如果删除的不是叶子结点呢？

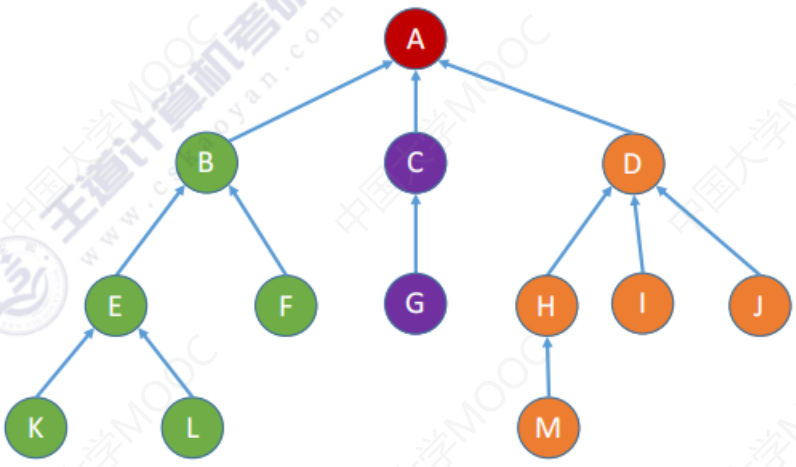
	data	parent
0	A	-1
1	B	0
2	C	0
3	D	0
4	E	1
5	F	1
6	L	4
7	H	3
8	I	3
9	J	3
10	K	4
11	M	7
12		
13		

删除数据元素（方案二）

王道考研/CSKAOYAN.COM

7

双亲表示法（顺序存储）



双亲表示法：每个结点中保存指向双亲的“指针”

	data	parent
0	A	-1
1	B	0
2	C	0
p → 3	D	0
4	E	1
5	F	1
6		-1
7	H	3
8	I	3
9	J	3
10	K	4
11	M	7
12	L	4
13		

优点：查指定结点的双亲很方便

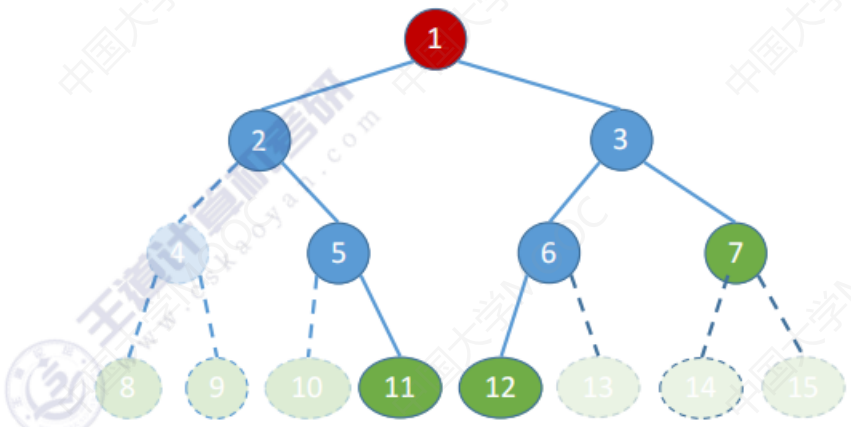
空数据导致遍历更慢

缺点：查指定结点的孩子只能从头遍历

王道考研/CSKAOYAN.COM

8

回顾：二叉树的顺序存储

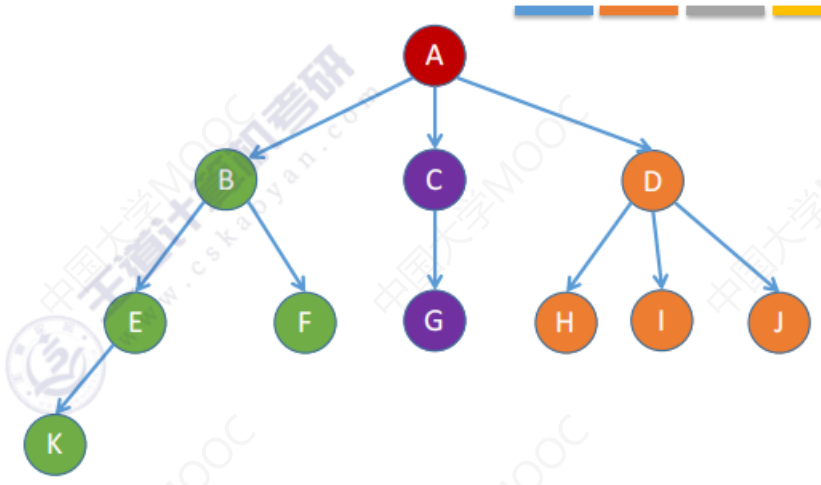


二叉树的顺序存储中，一定要把二叉树的结点编号与完全二叉树对应起来

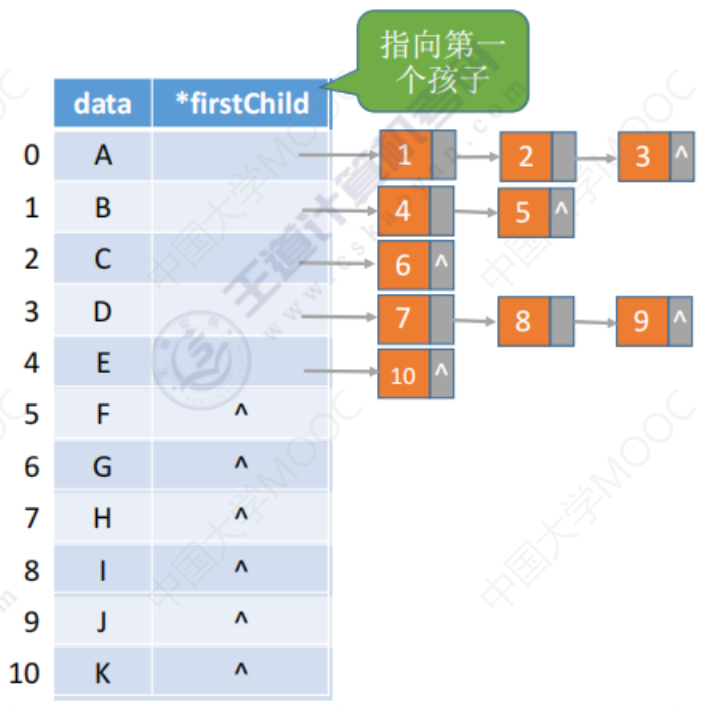
- i 的左孩子 ——2i
- i 的右孩子 ——2i+1
- i 的父节点 ——$\lfloor i/2 \rfloor$

结点编号不仅反映了存储位置，也隐含了结点之间的逻辑关系

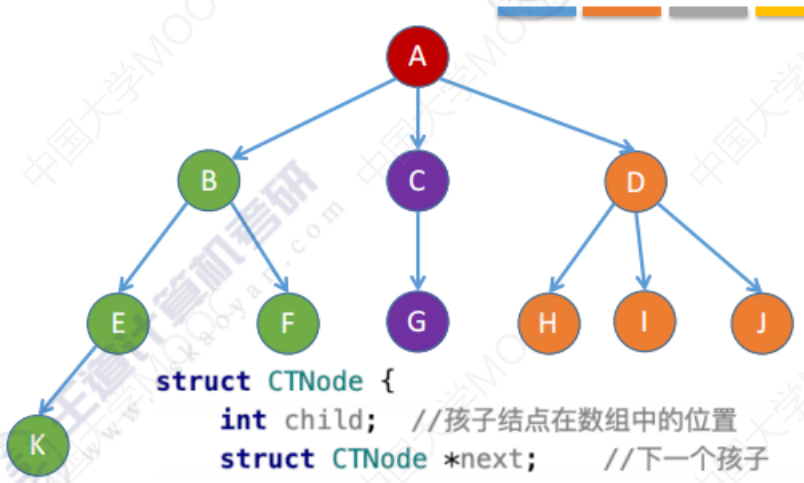
孩子表示法（顺序+链式存储）



孩子表示法：顺序存储各个节点，每个结点中保存孩子链表头指针

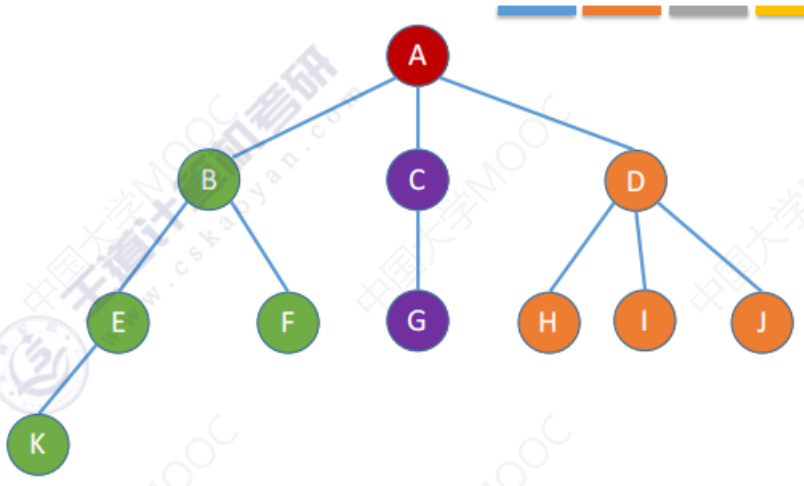


孩子表示法（顺序+链式存储）




```
struct CTNode {
    int child;  //孩子结点在数组中的位置
    struct CTNode *next;     //下一个孩子
};
typedef struct {
    ElemType data;
    struct CTNode *firstChild;  //第一个孩子
} CTBox;
typedef struct {
    CTBox nodes[MAX_TREE_SIZE];
    int n, r;   //结点数和根的位置
} CTree;
```

孩子兄弟表示法（链式存储）

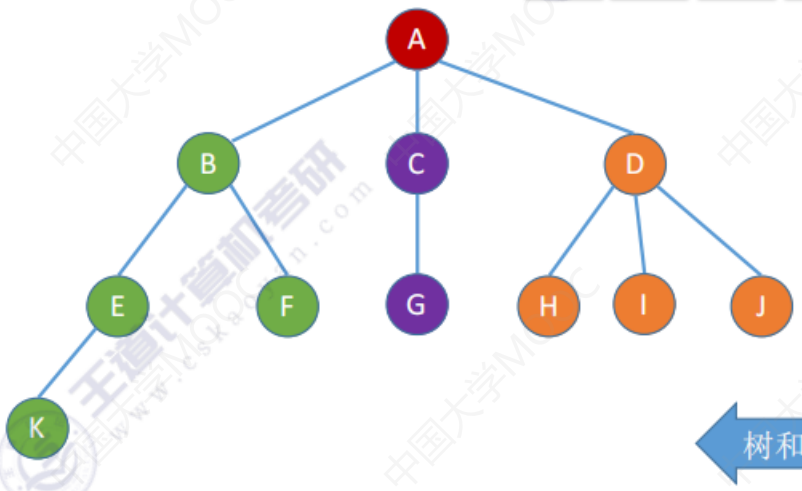



```
//二叉树的结点（链式存储）
typedef struct BiTNode{
    ElemType data;
    struct BiTNode  *lchild,*rchild;
}BiTNode,*BiTree;
```

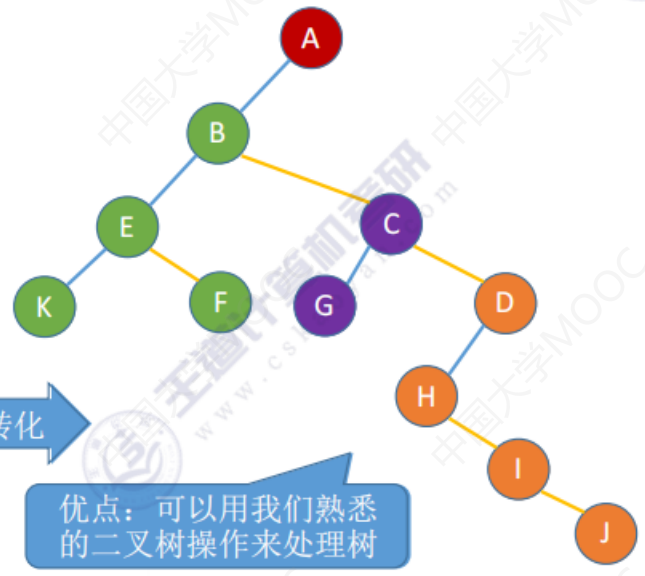
```
//树的存储—孩子兄弟表示法
typedef struct CSNode{
    ElemType data;                                  //数据域
    struct CSNode *firstchild,*nextsibling;         //第一个孩子和右兄弟指针
}CSNode,*CSTree;
```

二叉链表

孩子兄弟表示法（链式存储）



树和二叉树的转化



优点：可以用我们熟悉的二叉树操作来处理树

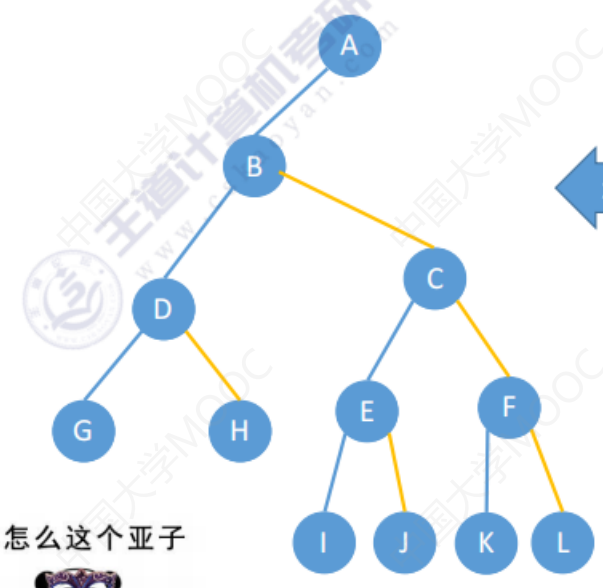
```
//树的存储—孩子兄弟表示法
typedef struct CSNode{
    ElemType data;
    struct CSNode *firstchild,*nextsibling;
}CSNode,*CSTree;
```

看作左指针

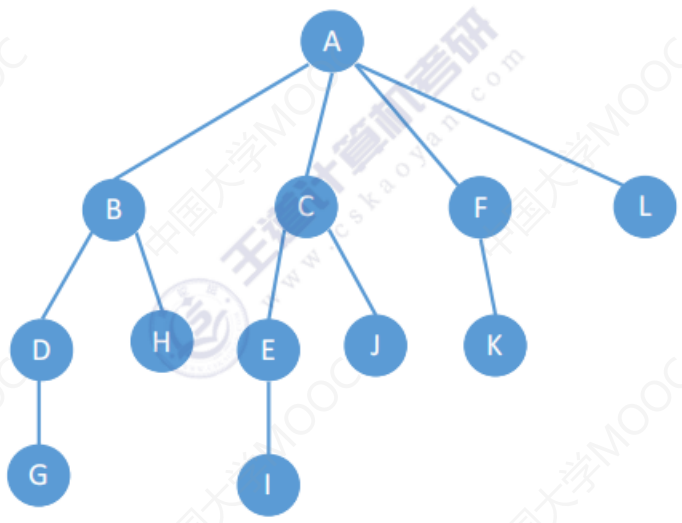
看作右指针

```
//数据域
//第一个孩子和右兄弟指针
```


孩子兄弟表示法（链式存储）



树和二叉树的转化



你怎么这个亚子

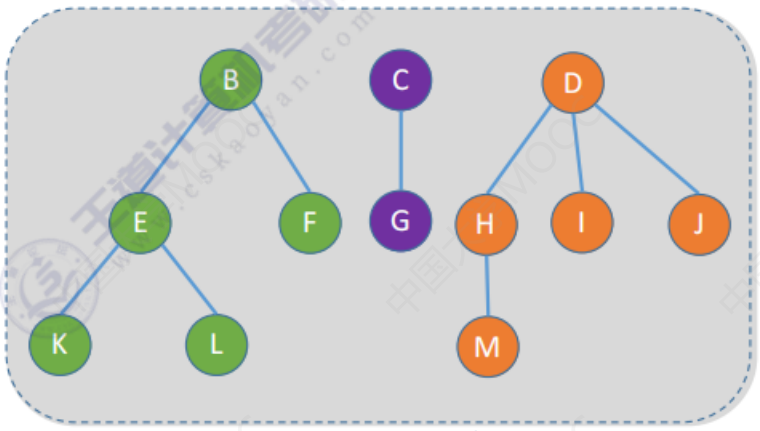
用孩子兄弟表示法存储的树在物理上呈现出“二叉树”的亚子

从逻辑视角看，这棵树原本面貌

森林和二叉树的转换

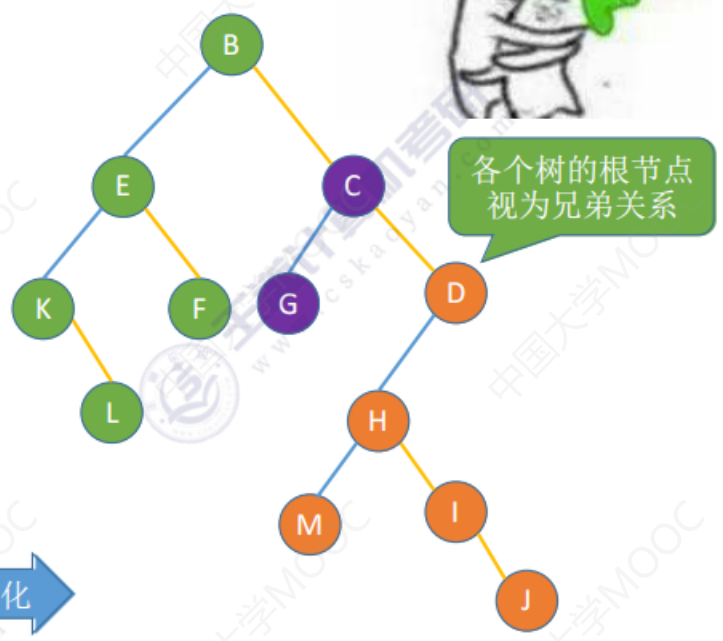
本质：用二叉链表存储森林

森林。森林是m（$m\geq 0$）棵互不相交的树的集合



森林

森林和二叉树的转化



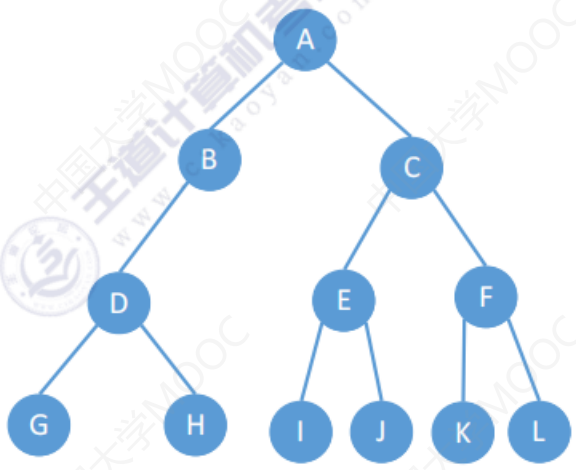
各个树的根节点视为兄弟关系

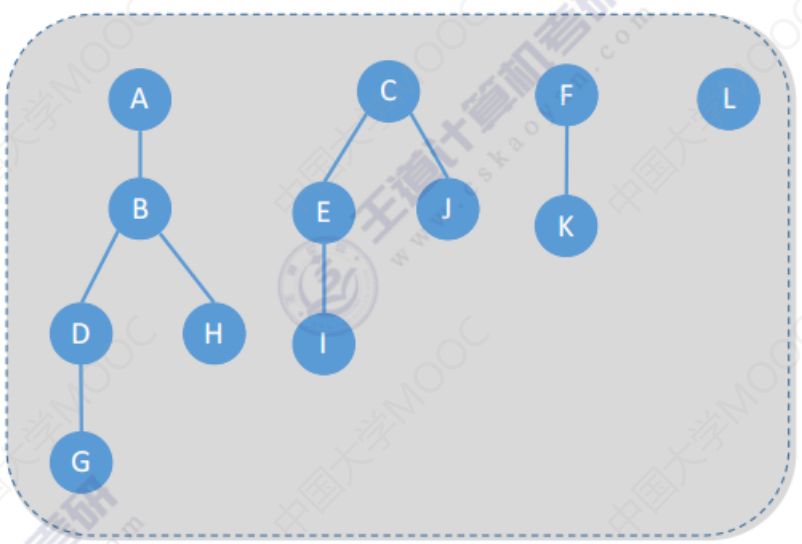
王道考研/CSKAOYAN.COM

15

森林和二叉树的转换

本质：用二叉链表存储森林





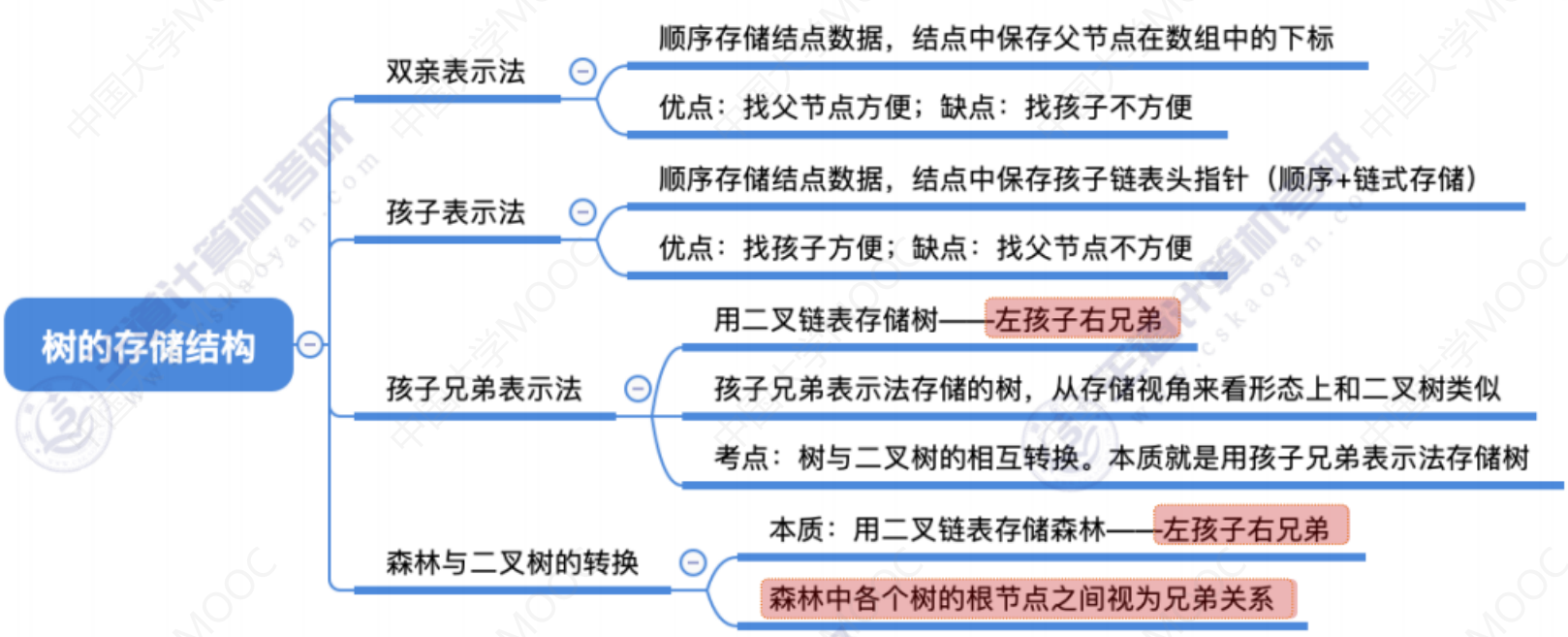
森林和二叉树的转化

森林

王道考研/CSKAOYAN.COM

16

知识回顾与重要考点



王道考研/CSKAOYAN.COM