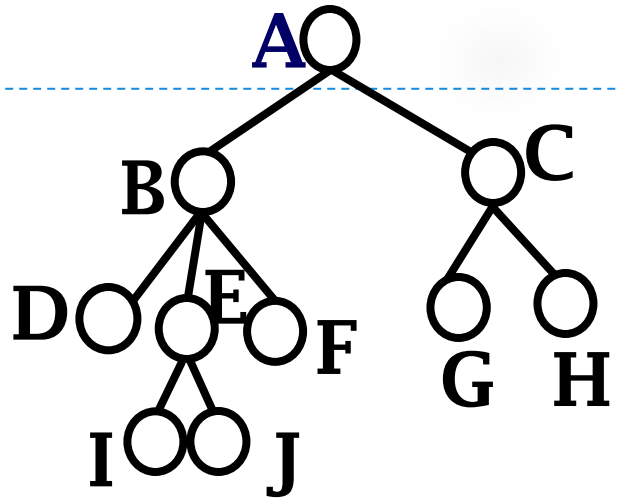
数据结构与算法（六）

张铭 主讲

采用教材：张铭，王腾蛟，赵海燕 编写
高等教育出版社，2008. 6（“十一五”国家级规划教材）

http://www.jpk.pku.edu.cn/pkujpk/course/sjjg

第6章　树



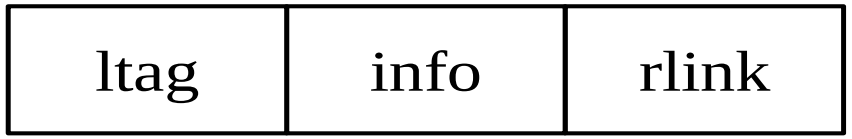
- 树的定义和基本术语
- 树的链式存储结构
- **树的顺序存储结构**
- K叉树

树的顺序存储结构

- 带右链的先根次序表示
- 带双标记的先根次序表示
- 带双标记的层次次序表示
- 带度数的后根次序表示

带右链的先根次序表示

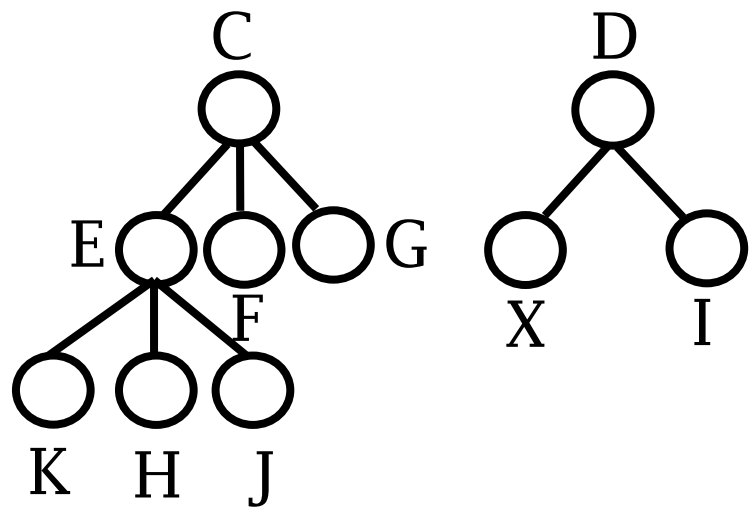
- 结点按先根次序顺序连续存储

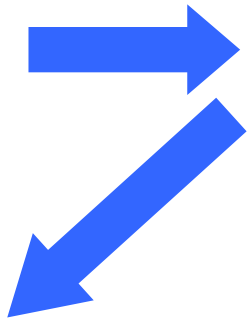


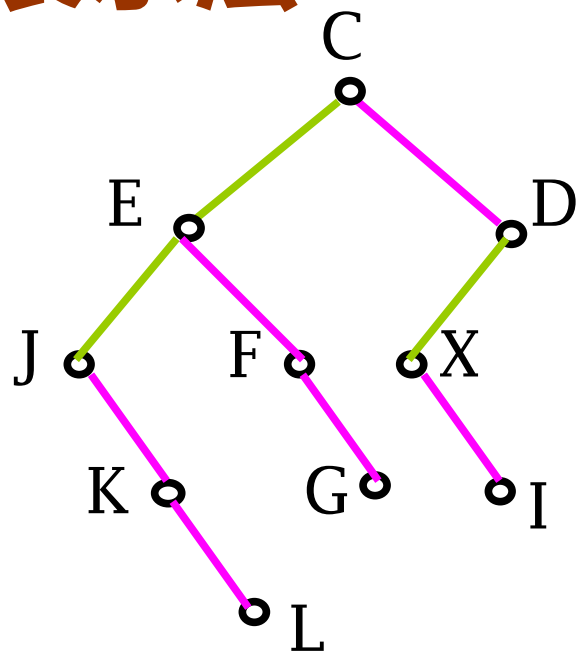
  - info：结点的数据
  - rlink：右指针
    - 指向结点的下一个兄弟、即对应的二叉树中结点的右子结点
  - ltag：标记
    - 树结点没有子结点，即二叉树结点没有左子结点，ltag为 1
    - 否则为0

6.3 树的顺序存储结构

带右链的先根次序表示法

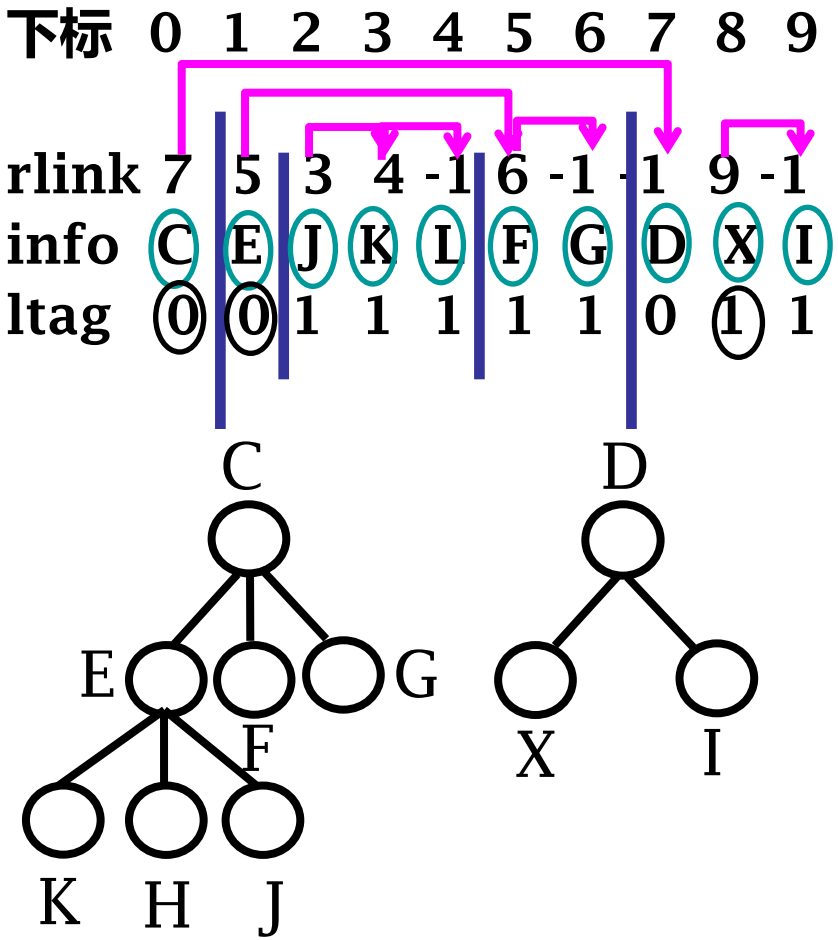


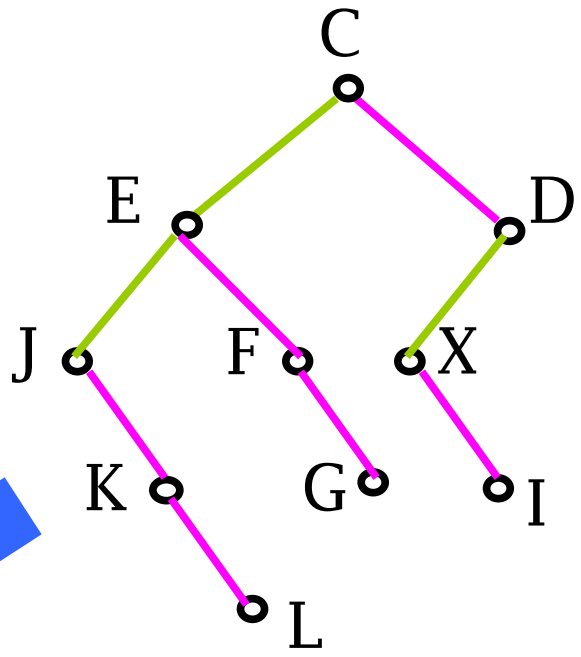




下标	0	1	2	3	4	5	6	7	8	9
rlink	7	5	3	4	-1	6	-1	-1	9	-1
info	C	E	J	K	L	F	G	D	X	I
ltag	0	0	1	1	1	1	1	0	1	1

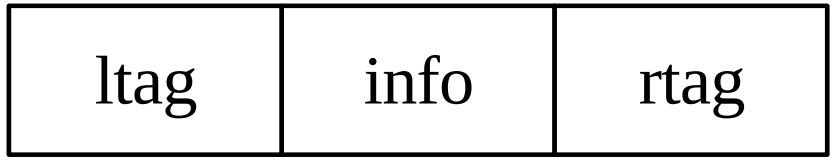
从先根rlink-ltag到树





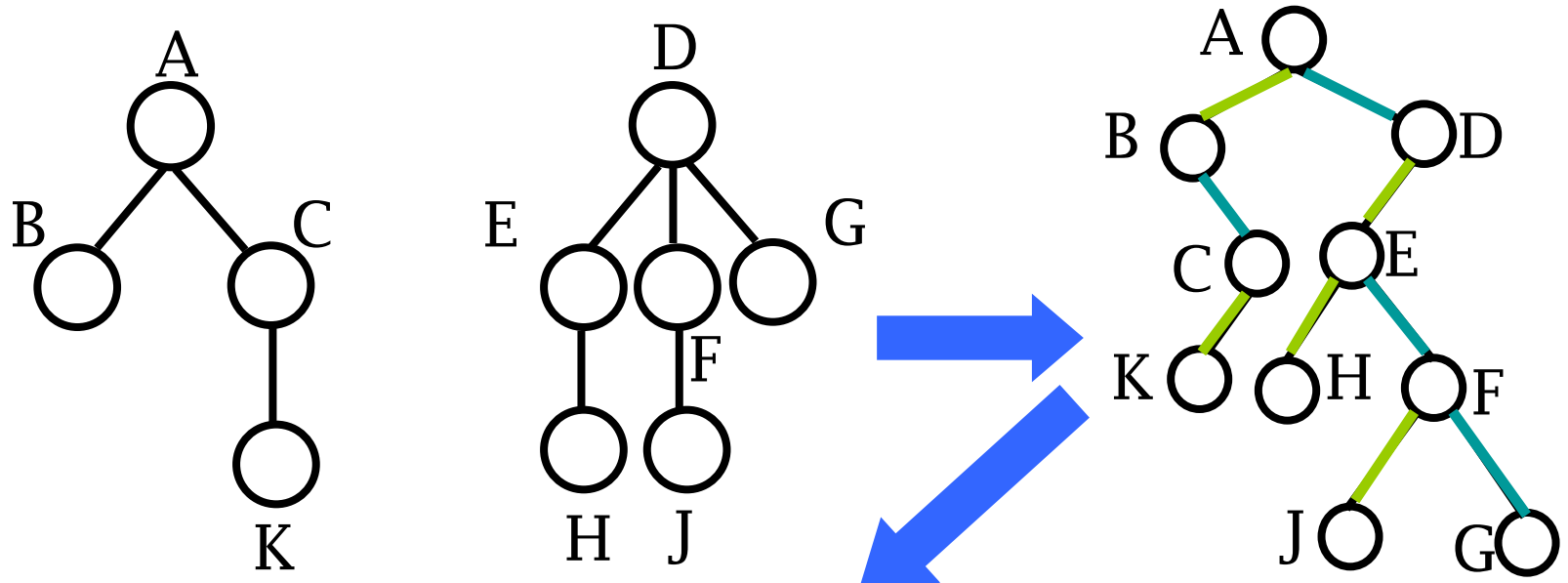
带双标记的先根次序表示

- 带右链的先根次序表示”中rlink也有冗余，可以把rlink指针替换为一个标志位rtag，成为“带双标记的先根次序表示”。其中，每个结点包括结点本身数据，以及两个标志位ltag和rtag，其结点的形式为：



由结点的先根次序以及ltag、rtag两个标志位，就可以确定树“左孩子/右兄弟”链表中结点的llink和rlink值。其中llink的确定与带右链的先根次序表示法相同。

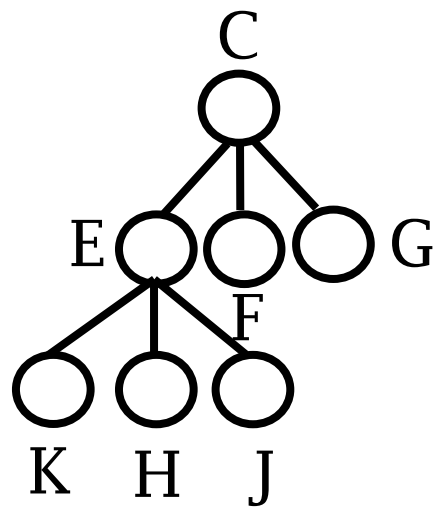
带双标记位的先根次序表示法

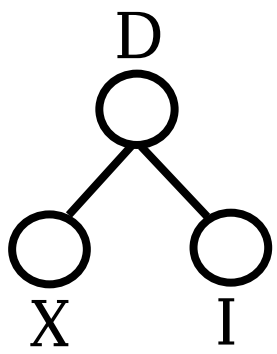


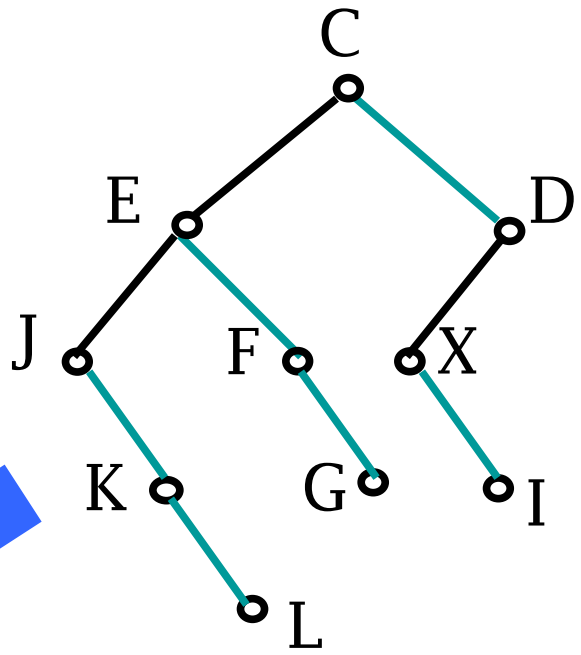
rtag	0	0	1	1	1	0	1	0	1	1
info	A	B	C	K	D	E	H	F	J	G
ltag	0	1	0	1	0	0	1	0	1	1

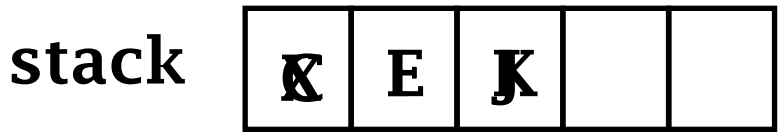
从rtag-ltag先根序列到树

下标	0	1	2	3	4	5	6	7	8	9
rtag	0	0	0	0	1	0	1	1	0	1
info	C	E	J	K	L	F	G	D	X	I
ltag	0	0	1	1	1	1	1	0	1	1









从双标记的先根次序恢复树

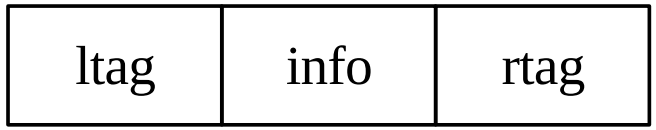
```
template<class T>
class DualTagTreeNode  {                          // 双标记位先根次序树结点类
public:
    T info;                                       // 结点数据信息
    int ltag, rtag;                               // 左、右标记
    DualTagTreeNode();                            // 构造函数
    virtual ~DualTagTreeNode(); };

template <class T>
Tree<T>::Tree(DualTagTreeNode<T> *nodeArray, int count)  {
     // 利用带双标记位的先根次序表示构造左孩子右兄弟表示的树
    using std::stack;                                          // 使用STL中的栈
    stack<TreeNode<T>* > aStack;
    TreeNode<T> *pointer = new TreeNode<T>;   // 准备建立根结点
    root = pointer;
```

6.3 树的顺序存储结构

```
for (int i = 0; i < count-1; i++) {                    // 处理一个结点
  pointer->setValue(nodeArray[i].info);                // 结点赋值
  if (nodeArray[i].rtag == 0)                          // 若右标记为0则将结点压栈
     aStack.push(pointer);
  else pointer->setSibling(NULL);                      // 右标记为1，则右兄弟指针为空
  TreeNode<T> *temppointer = new TreeNode<T>;          // 预先准备下一个
  if (nodeArray[i].ltag == 0)                          // 左标记为0，则设置孩子结点
     pointer->setChild(temppointer);
  else {                                         // 若左标记为1
     pointer->setChild(NULL);                    // 孩子指针设为空
     pointer = aStack.top();                             // 取栈顶元素
     aStack.pop();
     pointer->setSibling(temppointer); }                 // 为栈顶设置一个兄弟结点
  pointer = temppointer; }
pointer->setValue(nodeArray[count-1].info); // 处理最后一个结点
pointer->setChild(NULL);  pointer->setSibling(NULL); }
```

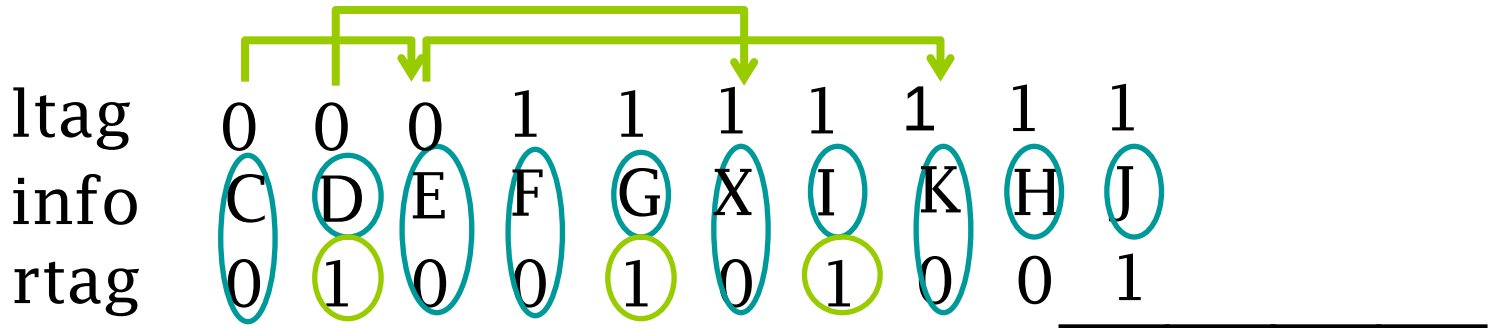
带双标记的层次次序表示法

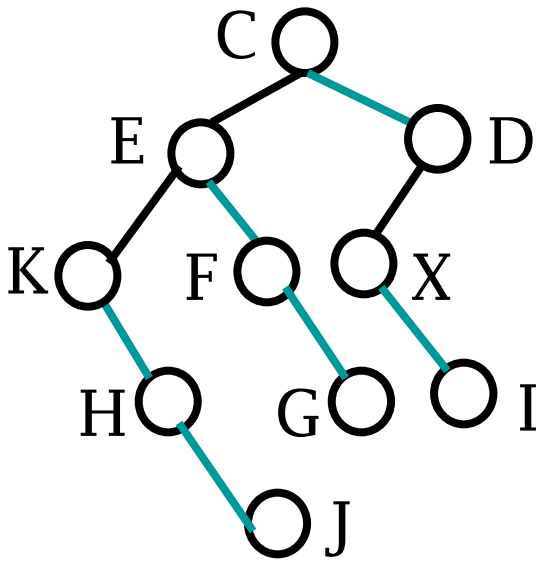
- 结点按 层次次序顺序 存储在连续存储单元



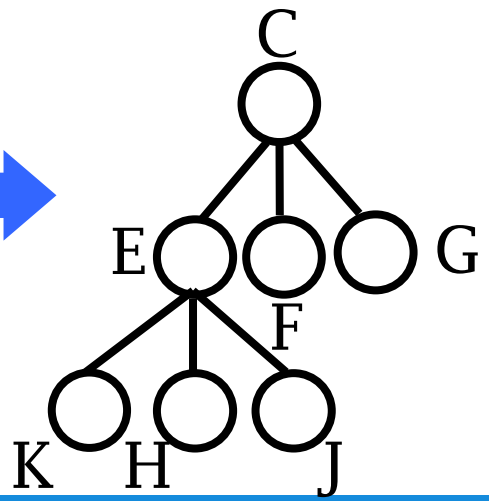
- info是结点的数据
- ltag是一个一位的左标记，当结点没有子节点，即对应的二叉树中结点没有左子结点时， ltag为1，否则为0
- rtag是一个一位的右标记，当结点没有下一个兄弟，即对应的二叉树中结点没有右子结点时， rtag为1，否则为0

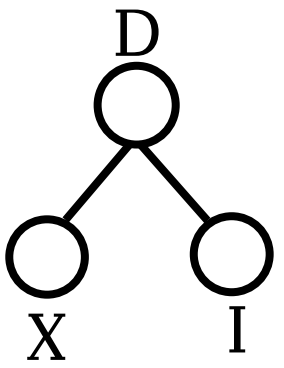
带双标记的层次次序转换为树





queue | C | D | E |





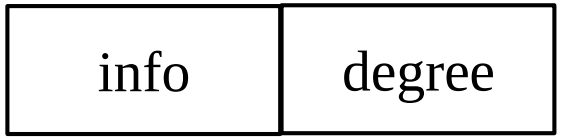
带双标记位的层次次序构造

```
template <class T>
Tree<T>::Tree(DualTagWidthTreeNode<T>* nodeArray, int
count)  {
  using std::queue;                                  // 使用STL队列
  queue<TreeNode<T>*> aQueue;
  TreeNode<T>* pointer=new TreeNode<T>; // 建立根
  root=pointer;
  for(int i=0;i<count-1;i++) {                       // 处理每个结点
    pointer->setValue(nodeArray[i].info);
    if(nodeArray[i].ltag==0) aQueue.push(pointer);   // 入队
       else pointer->setChild(NULL);                 // 左孩子设为空
    TreeNode<T>* temppointer=new TreeNode<T>;
```

```
    if(nodeArray[i].rtag == 0)
      pointer->setSibling(temppointer);
    else {
        pointer->setSibling(NULL);     // 右兄弟设为空
        pointer=aQueue.front();        // 取队列首结点指针
        aQueue.pop();                  // 队首元素出队列
        pointer->setChild(temppointer);
    }
    pointer=temppointer;
  }
  pointer->setValue(nodeArray[count-1].info); // 最后一个结点
  pointer->setChild(NULL);  pointer->setSibling(NULL);
}
```

带度数的后根次序表示

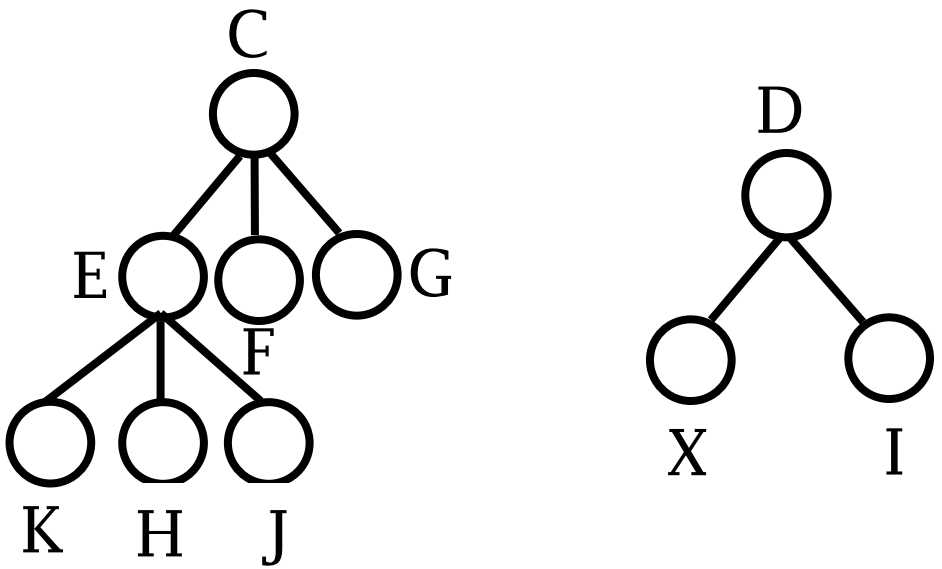
- 在带度数的后根次序表示中，结点按后根次序顺序存储在一片连续的存储单元中，结点的形式为



- 其中info是结点的数据，degree是结点的度数

带度数的后根次序表示法

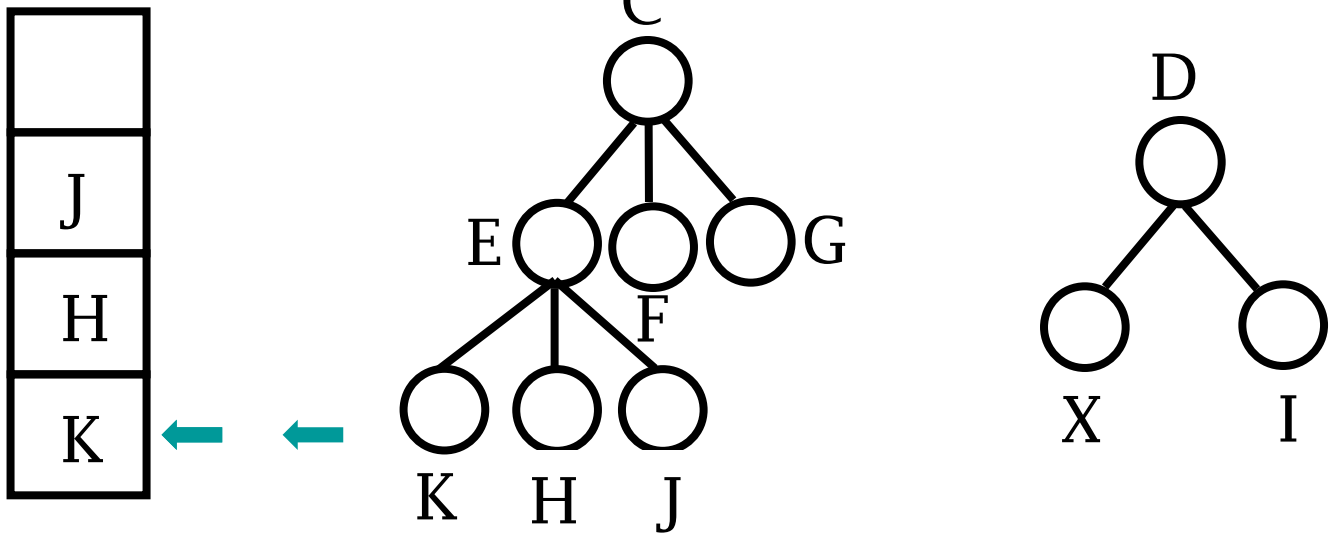
degree	0	0	0	3	0	0	3	0	0	2
info	K	H	J	E	F	G	C	X	I	D



6.3 树的顺序存储结构

带度数的后根次序变成树

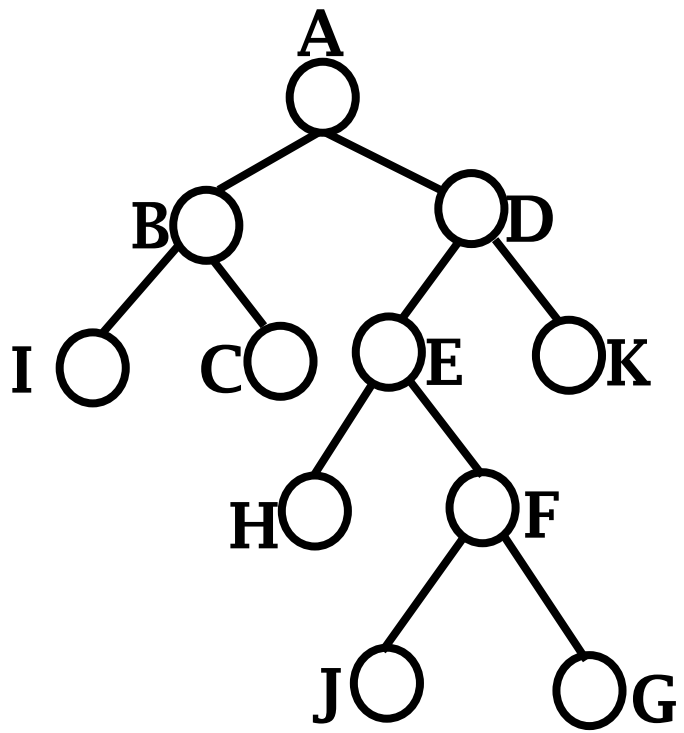
degree	0	0	0	3	0	0	3	0	0	2
info	K	H	J	E	F	G	C	X	I	D



6.3 树的顺序存储结构

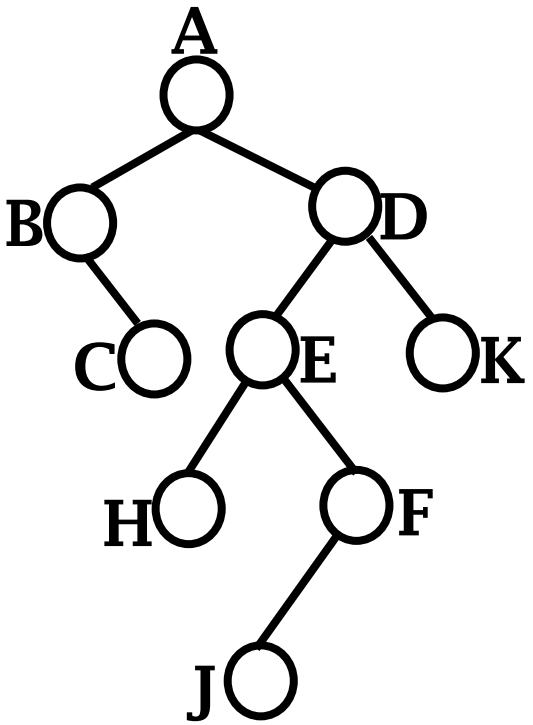
- 带标记的满二叉树前序序列

A’ B’ I C D’ E’ H F’ J G K



- 带标记的伪满二叉树前序序列

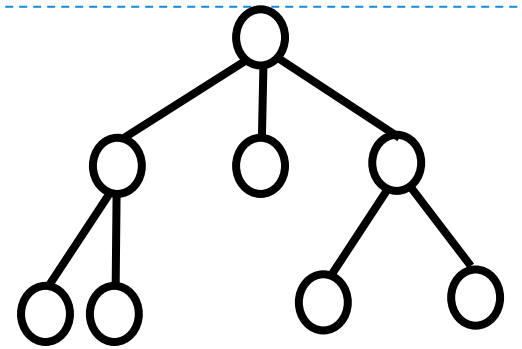
A’ B’ / C D’ E’ H F’ J / K



思考：森林的顺序存储

- 信息冗余问题
- 树的其他顺序存储
  - 带度数的先根次序？
  - 带度数的层次次序？
- 二叉树的顺序存储？
  - 二叉树与森林对应，但语义不同
    - 带右链的二叉树前序
    - 带左链的二叉树层次次序

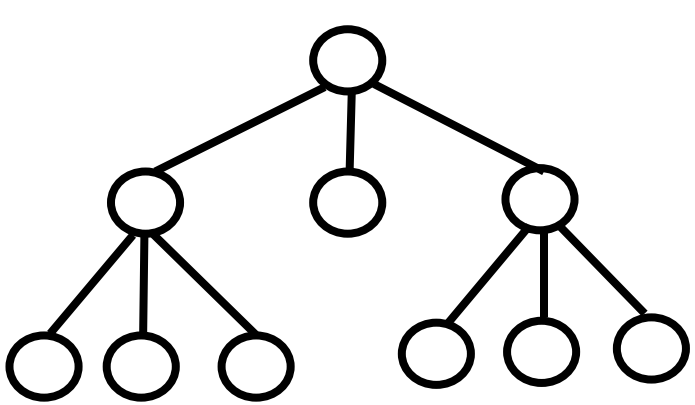
K 叉树定义



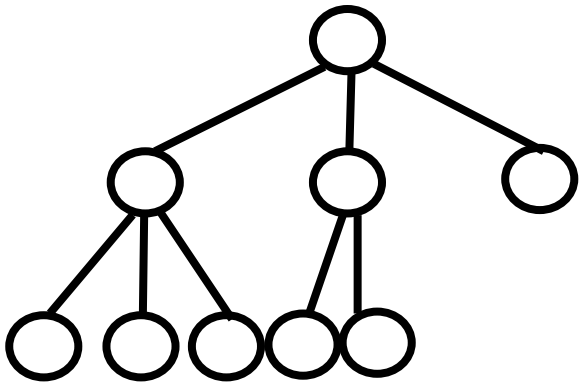
- K 叉树 T 是具有下列性质的有限结点集：
  - (a) 集合可以为空；
  - (b) 非空集合是由一个根结点 root 及 K 棵互不相交的 K 叉树构成。
- 其余结点被划分成 T_0，T_1，…，T_{K-1}（K ≥ 1）个子集，每个子集都是 K 叉树，使得 $T = \{R, T_0, T_1, \cdots, T_{K-1}\}$。
- K 叉树的各分支结点都有 K 个子结点

满 K 叉树和完全 K 叉树

- K 叉树 (K-ary Tree) 的结点有 K 个子结点
- 二叉树的许多性质可以推广到 K 叉树
  - 满K叉树和完全K叉树与满二叉树和完全二叉树是类似的
  - 也可以把完全K叉树存储在一个数组中



满3叉树



完全3叉树

数据结构与算法

谢谢聆听

国家精品课“数据结构与算法”
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

张铭，王腾蛟，赵海燕
高等教育出版社，2008. 6。“十一五”国家级规划教材