



数据结构与算法（六）

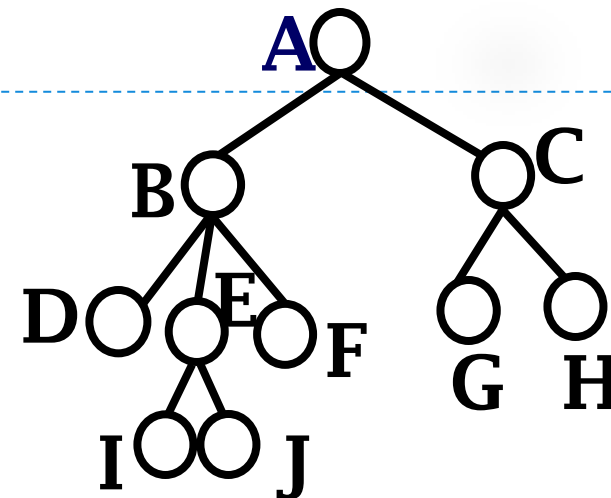
张铭 主讲

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.jpku.pku.edu.cn/pkujpku/course/sjjg>

第6章 树

- 树的定义和基本术语
 - 树和森林
 - 森林与二叉树的等价转换
 - 树的抽象数据类型
 - 树的遍历
- 树的链式存储结构
- 树的顺序存储结构
- K叉树



6.1 树的定义和基本术语

树的抽象数据类型

```
template<class T>
class TreeNode {
public:
    TreeNode(const T& value);
    virtual ~TreeNode() {};
    bool isLeaf();
    T Value();
    TreeNode<T> *LeftMostChild();
    TreeNode<T> *RightSibling();
    void setValue(const T& value);
    void setChild(TreeNode<T> *pointer);
    void setSibling(TreeNode<T> *pointer);
    void InsertFirst(TreeNode<T> *node);
    void InsertNext(TreeNode<T> *node);
};
```

// 树结点的ADT

// 拷贝构造函数

// 析构函数

// 判断当前结点是否为叶结点

// 返回结点的值

// 返回第一个左孩子

// 返回右兄弟

// 设置当前结点的值

// 设置左孩子

// 设置右兄弟

// 以第一个左孩子身份插入结点

// 以右兄弟的身份插入结点

6.1 树的定义和基本术语

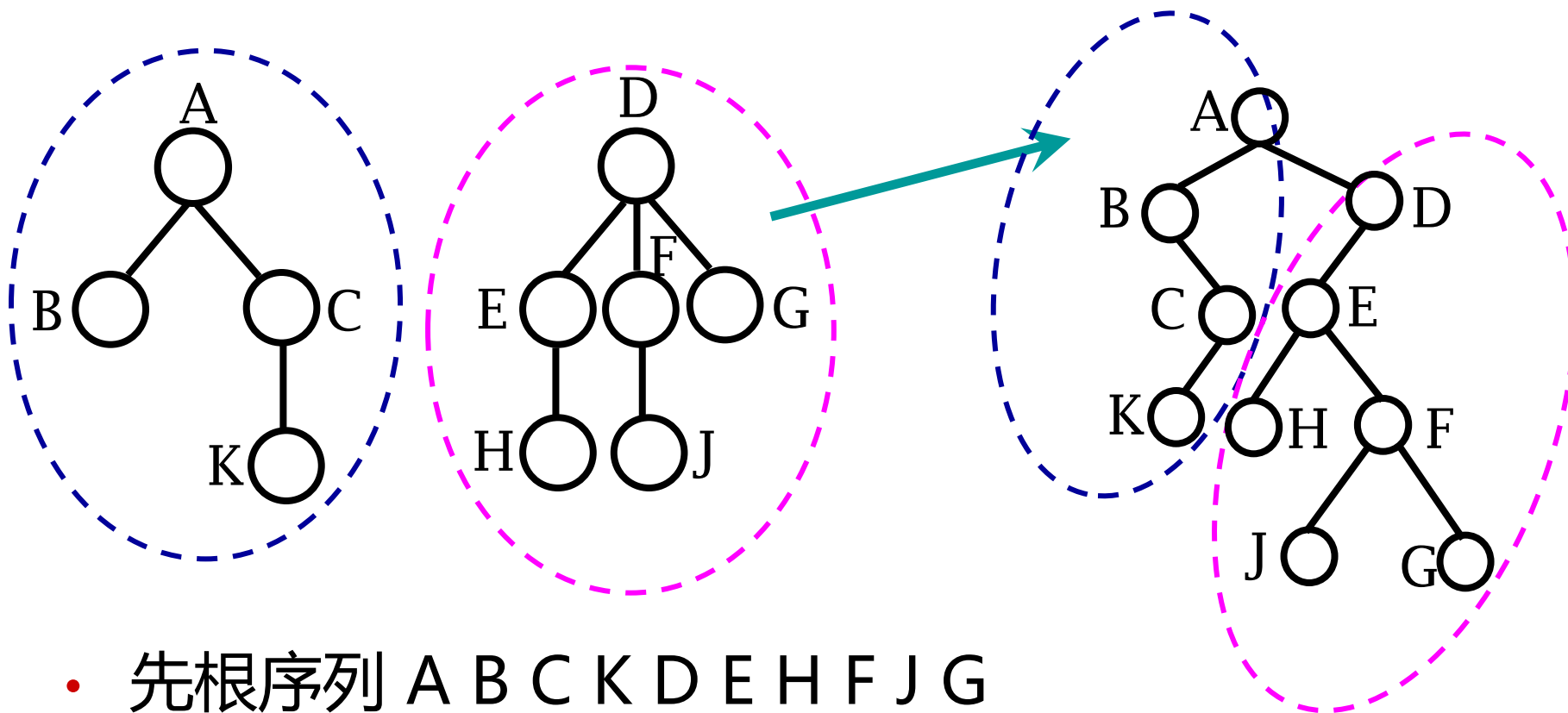
树的抽象数据类型

```
template<class T>
class Tree {
public:
    Tree();
    virtual ~Tree();
    TreeNode<T>* getRoot();
    void CreateRoot(const T& rootValue);
    bool isEmpty();
    TreeNode<T>* Parent(TreeNode<T> *current);
    TreeNode<T>* PrevSibling(TreeNode<T> *current);
    void DeleteSubTree(TreeNode<T> *subroot);
    void RootFirstTraverse(TreeNode<T> *root);
    void RootLastTraverse(TreeNode<T> *root);
    void WidthTraverse(TreeNode<T> *root);
};
```

// 构造函数
// 析构函数
// 返回树中的根结点
// 创建值为rootValue的根结点
// 判断是否为空树
// 返回父结点
// 返回前一个兄弟
// 删除以subroot子树
// 先根深度优先遍历树
// 后根深度优先遍历树
// 广度优先遍历树

6.1 树的定义和基本术语

森林的遍历



- 先根序列 A B C K D E H F J G
- 后根序列 B K C A H E J F G D



遍历森林vs遍历二叉树

- 先根次序遍历森林
 - 前序法遍历二叉树
- 后根次序遍历森林
 - 按中序法遍历对应的二叉树
- 中根遍历？
 - 无法明确规定根在哪两个子结点之间

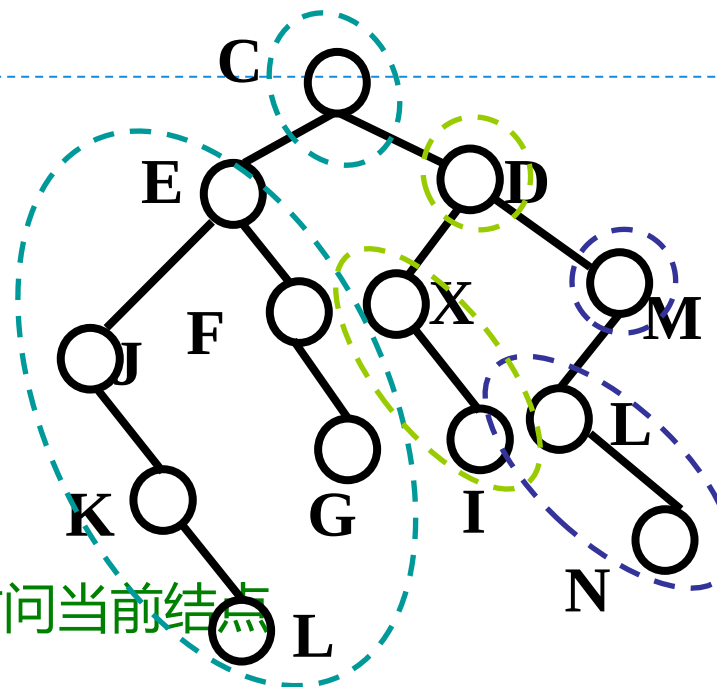
6.1 树的定义和基本术语

先根深度优先遍历森林

```

template<class T>
void Tree<T>::RootFirstTraverse(
    TreeNode<T> * root) {
    while (root != NULL) {
        Visit(root->Value());
        // 遍历第1棵树根的子树森林(树根除外)
        RootFirstTraverse(root->LeftMostChild());
        root = root->RightSibling();
    }
}

```



// 访问当前结点

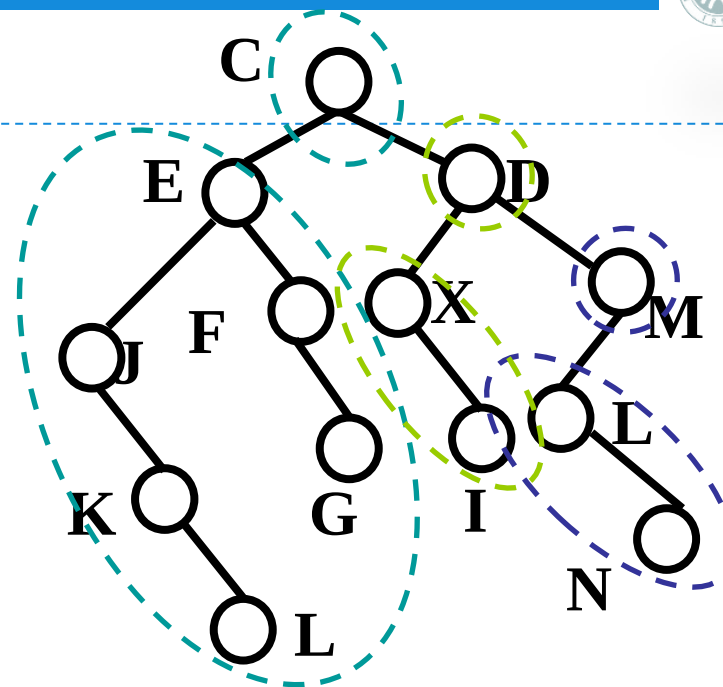
// 遍历其他树

后根深度优先遍历森林

```

template<class T>
void Tree<T>::RootLastTraverse(
    TreeNode<T> * root) {
    while (root != NULL) {
        // 遍历第一棵树根的子树森林
        RootLastTraverse(root->LeftMostChild());
        Visit(root->Value());           // 访问当前结点
        root = root->RightSibling();    // 遍历其他树
    }
}

```



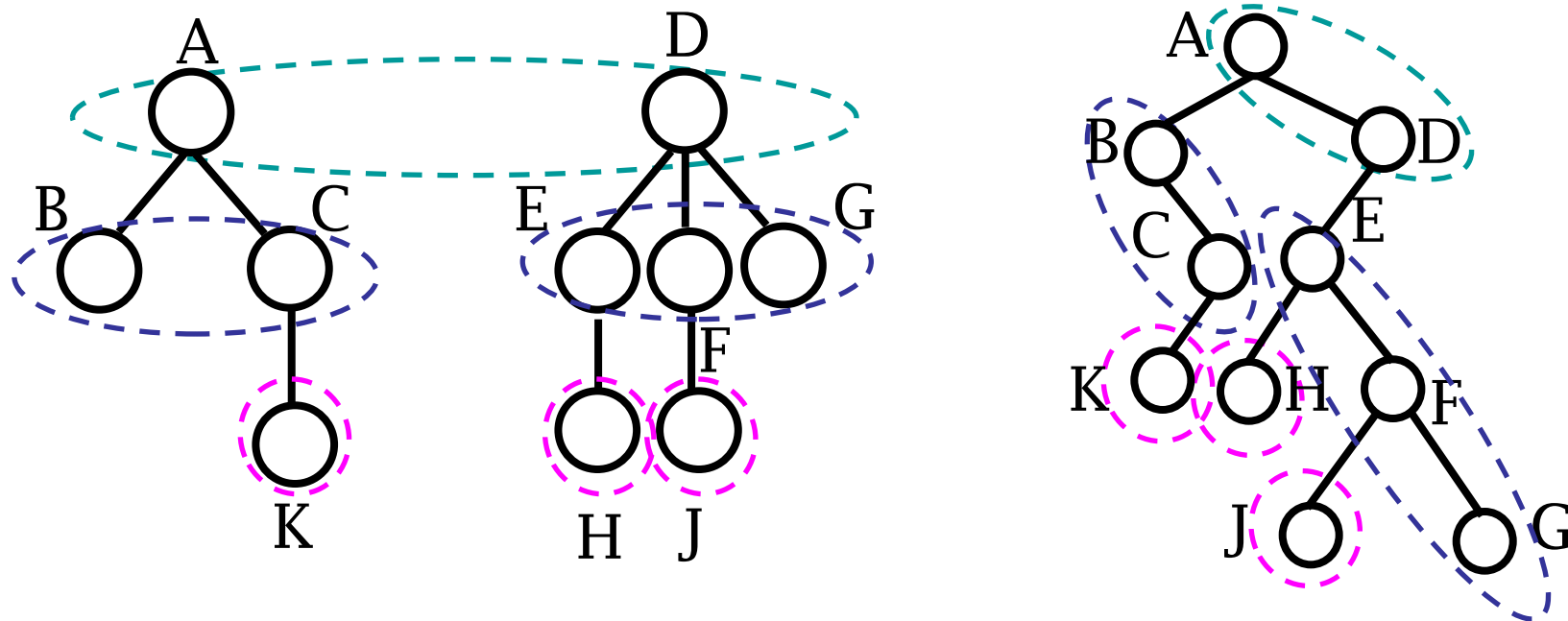


宽度优先遍历森林

- 宽度优先遍历
 - 也称广度优先遍历
 - 或称层次遍历
- a) 首先依次访问层数为0的结点
- b) 然后依次访问层数为1的结点
- c) 直到访问完最下一层的所有结点

6.1 树的定义和基本术语

广度优先遍历森林



- 森林广度优先：A D B C E F G K H J
- 看二叉链存储结构的右斜线**

6.1 树的定义和基本术语

广度优先遍历森林

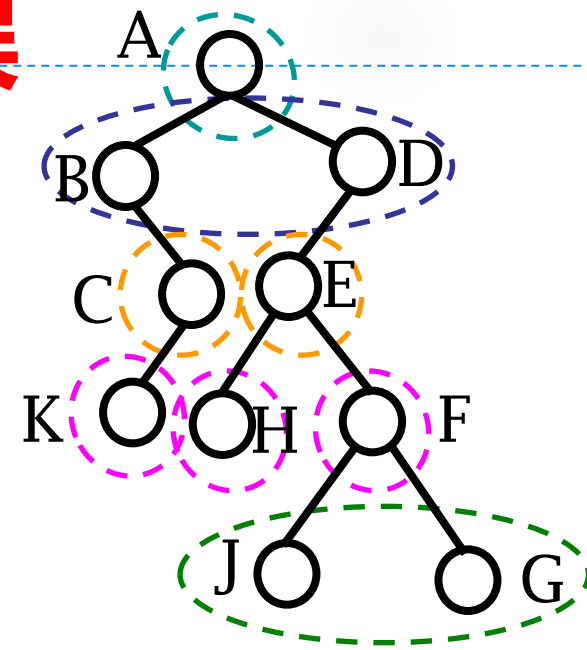
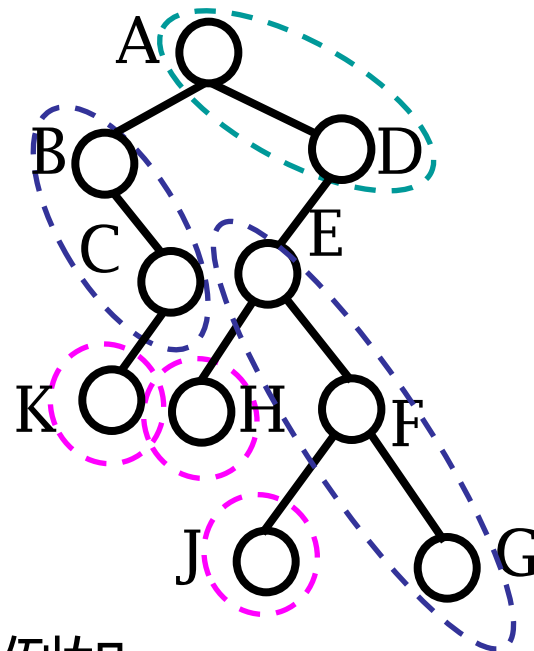
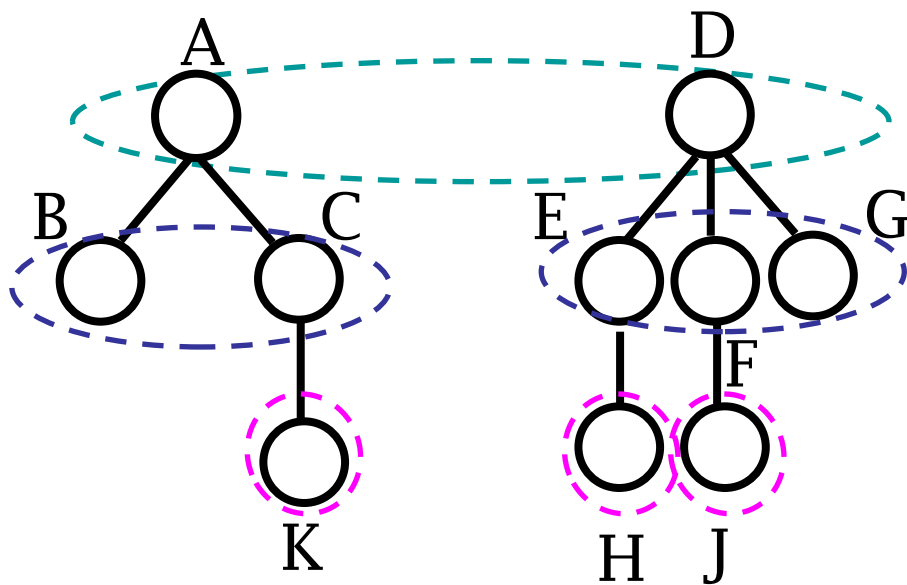
```
template<class T>
void Tree<T>::WidthTraverse(TreeNode<T> * root) {
    using std::queue;                // 使用STL队列
    queue<TreeNode<T>*> aQueue;
    TreeNode<T> * pointer = root;
    while (pointer != NULL) {
        aQueue.push(pointer);        // 当前结点进入队列
        pointer = pointer->RightSibling(); // pointer指向右兄弟
    }
    while (!aQueue.empty()) {
        pointer = aQueue.front();     // 获得队首元素
        aQueue.pop();                 // 当前结点出队列
        Visit(pointer->Value());      // 访问当前结点
        pointer = pointer->LeftMostChild(); // pointer指向最左孩子
        while (pointer != NULL) {    // 当前结点的子结点进队列
            aQueue.push(pointer);
            pointer = pointer->RightSibling();
        }
    }
}
```

思考

- 1. 能否直接用二叉树前序遍历框架来编写森林的先根遍历？
- 2. 能否直接用二叉树中序遍历框架来编写森林的后根遍历？
- 3. 森林的非递归深搜框架？

错误

思考：宽搜的各种观点



- 不能用二叉树的广度遍历模板。例如，
 - 上左图，森林广度优先：A D B C E F G K H J
 - 看二叉树的右斜线
 - 上右图，二叉树广度：A B D C E K H F J G
 - 看平行横线



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