

大宗百世不迁；小宗五世则迁

5. 二叉树

(e3) 后序遍历

邓俊辉

deng@tsinghua.edu.cn

递归

❖ 应用 : `BinNode::size()`

`BinTree::updateHeight()`

❖ `template <typename T, typename VST>`

`void traverse(BinNodePosi(T) x, VST & visit) {`

`if (!x) return;`

`traverse(x->lChild, visit);`

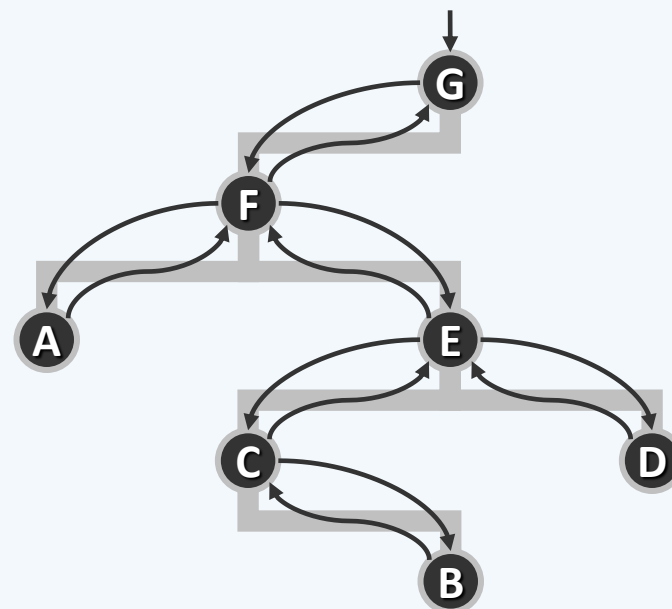
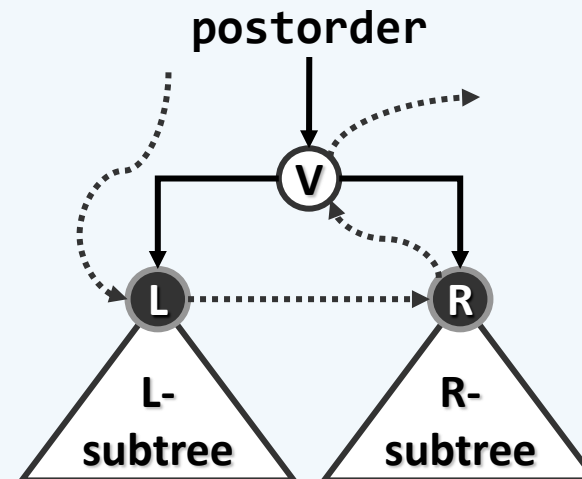
`traverse(x->rChild, visit);`

`visit(x->data);`

`} //T(n) = T(a) + T(n - a - 1) + O(1) = O(n)`

❖ 挑战 : 不依赖递归机制 , 能否实现后序遍历 ?

如何实现 ? 效率如何 ?



迭代：难点

❖ 难度在于

对左、右子树的递归遍历，**都不是**尾递归

❖ 解决方法

找到第一个被访问的节点

将其祖先及其右兄弟（如果存在）用栈保存

❖ 这样，原问题就被分解为

依次对若干棵右子树的遍历问题

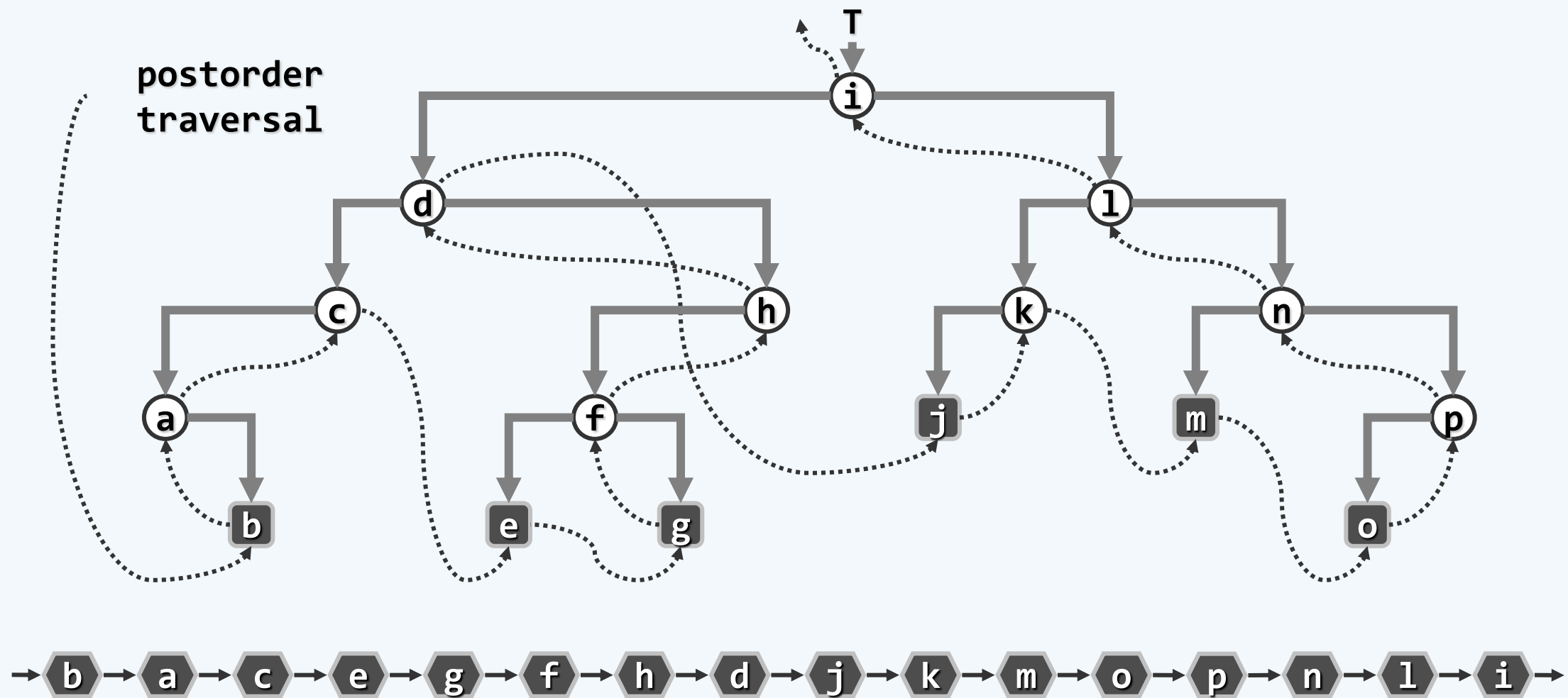
//同样地，这里应依什么“次”？

❖ 于是，首先要解决的问题仍是

后序遍历任一二叉树T时

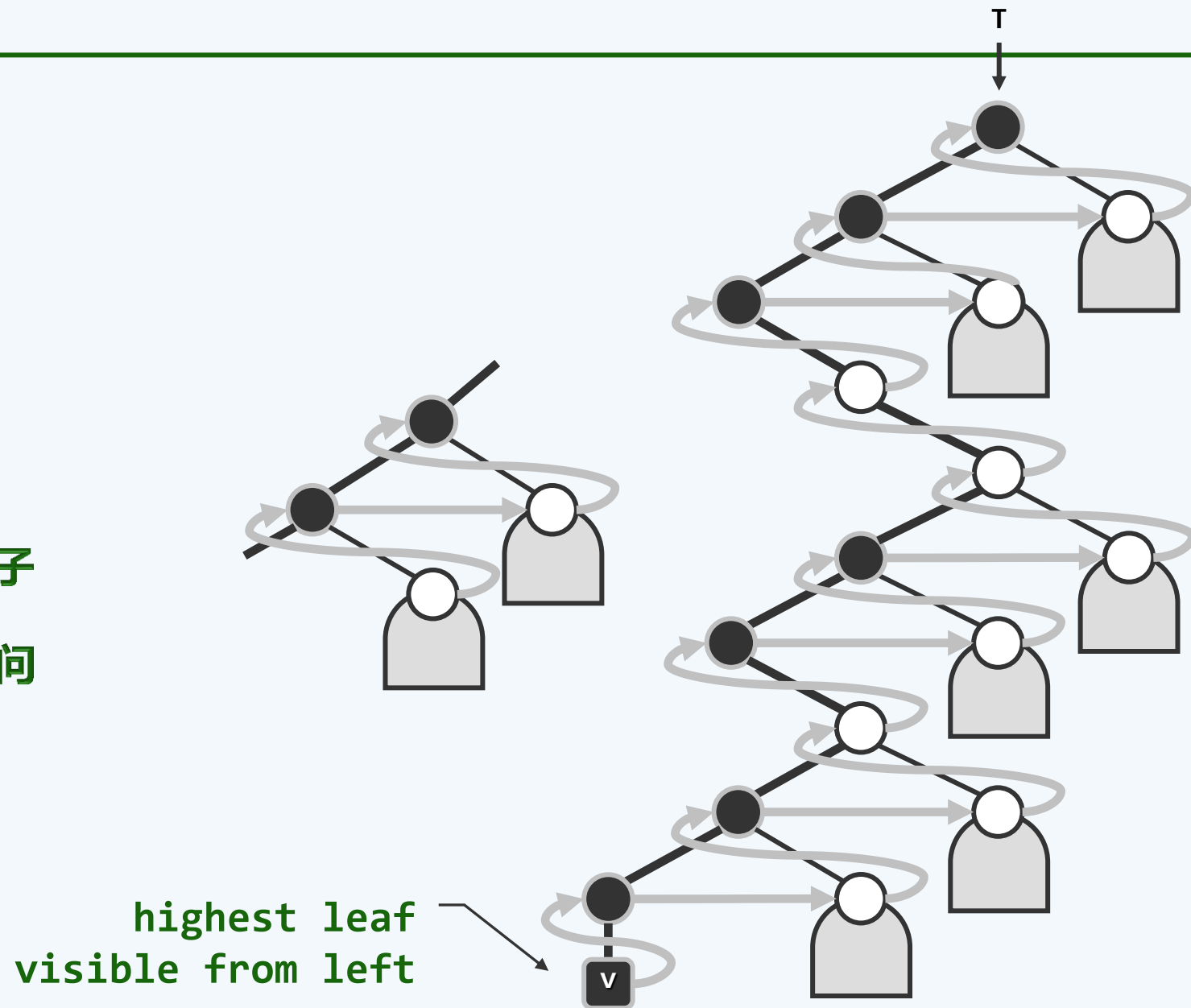
首先被访问的是哪个节点？如何找到它？

迭代：思路



迭代：思路

- ❖ 从根出发下行
尽可能沿左分支
实不得已，才沿右分支
- ❖ 最后一个节点
必是叶子，而且
是从左侧**可见**的**最高**叶子
- ❖ 这匹叶子，将首先接受访问



迭代：实现

```
❖ template <typename T>

static void gotoHLVFL( Stack <BinNodePosi(T)> & S ) {

    while ( BinNodePosi(T) x = S.top() ) //自顶而下反复检查栈顶节点

        if ( HasLChild(*x) ) { //尽可能向左。在此之前

            if ( HasRChild(*x) ) //若有右孩子，则

                S.push( x->rChild ); //优先入栈

            S.push( x->lChild ); //然后转向左孩子

        } else //实不得已

            S.push( x->rChild ); //才转向右孩子

    S.pop(); //返回之前，弹出栈顶的空节点

}
```

迭代：实现

❖ template <typename T, typename VST>

```
void travPost_I( BinNodePosi(T) x, VST & visit ) {
```

```
    Stack <BinNodePosi(T)> S; //辅助栈
```

```
    if (x) S.push(x); //根节点非空则首先入栈
```

```
    while ( !S.empty() ) { //x为当前节点
```

```
        if ( S.top() != x->parent ) //栈顶非x之父（则必为其右兄）
```

```
            gotoHLVFL(S); //在x的右子树中，找到HLVFL
```

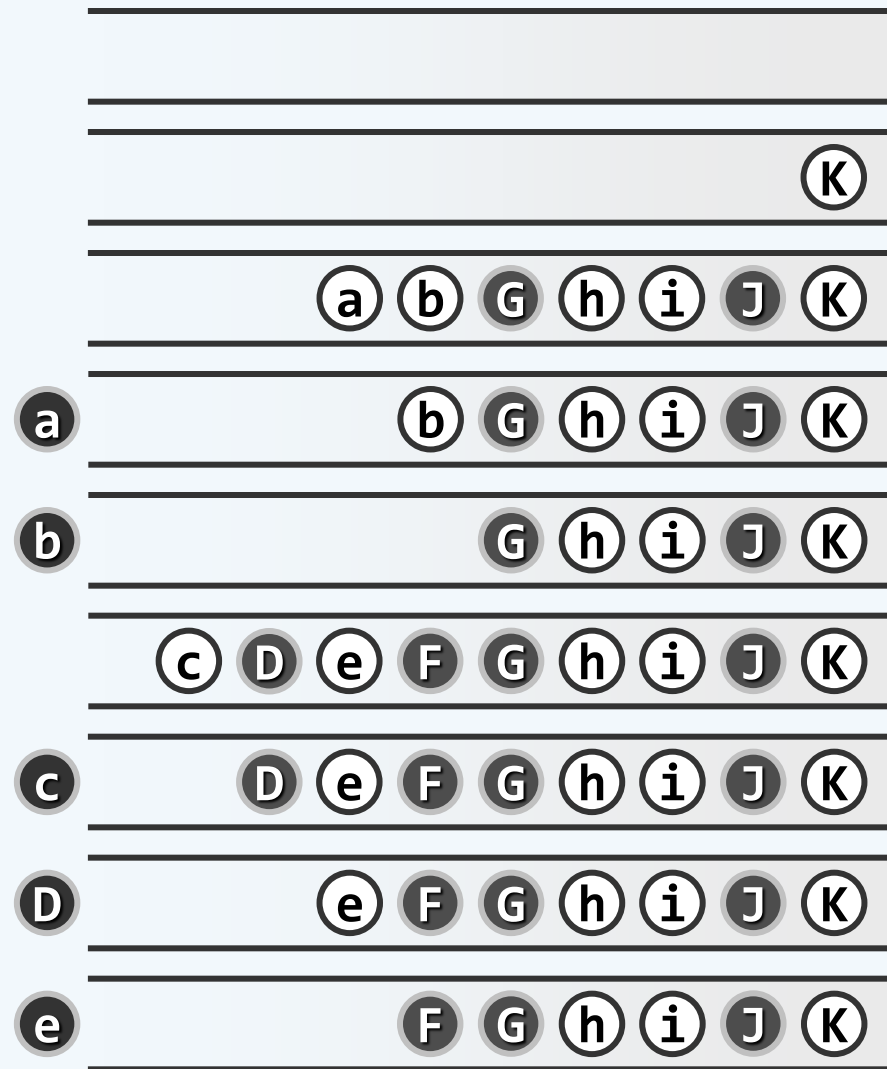
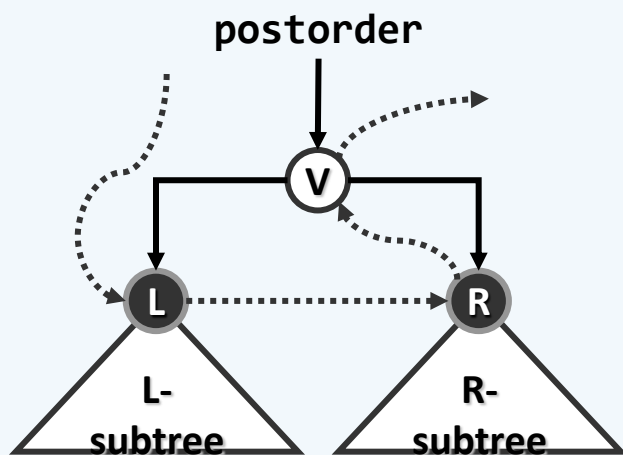
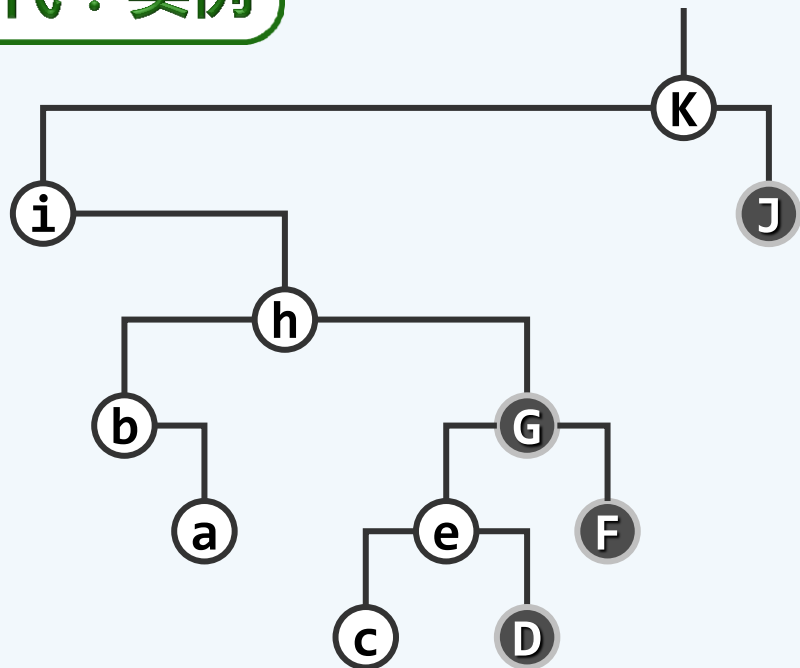
```
        x = S.pop(); //弹出栈顶（即前一节点之后继）以更新x，并随即
```

```
        visit( x->data ); //访问之
```

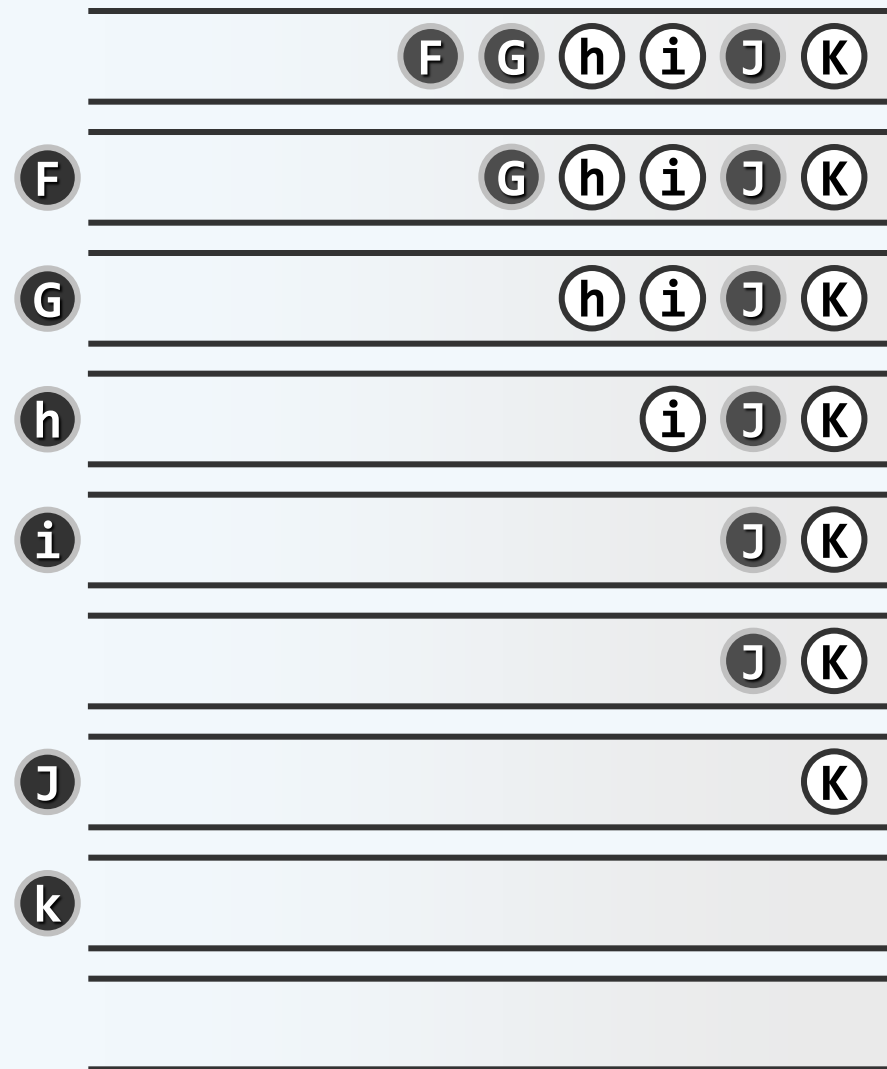
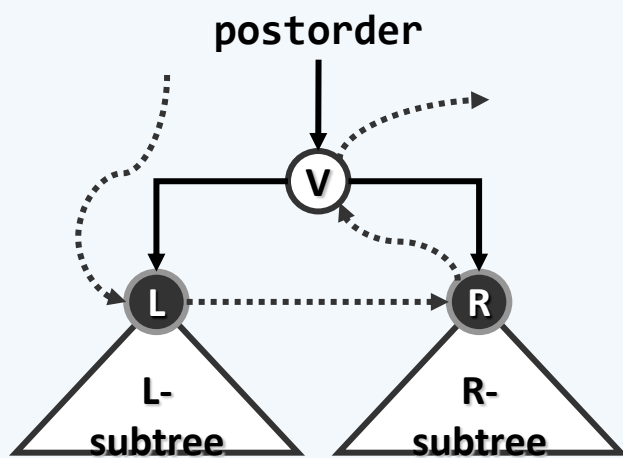
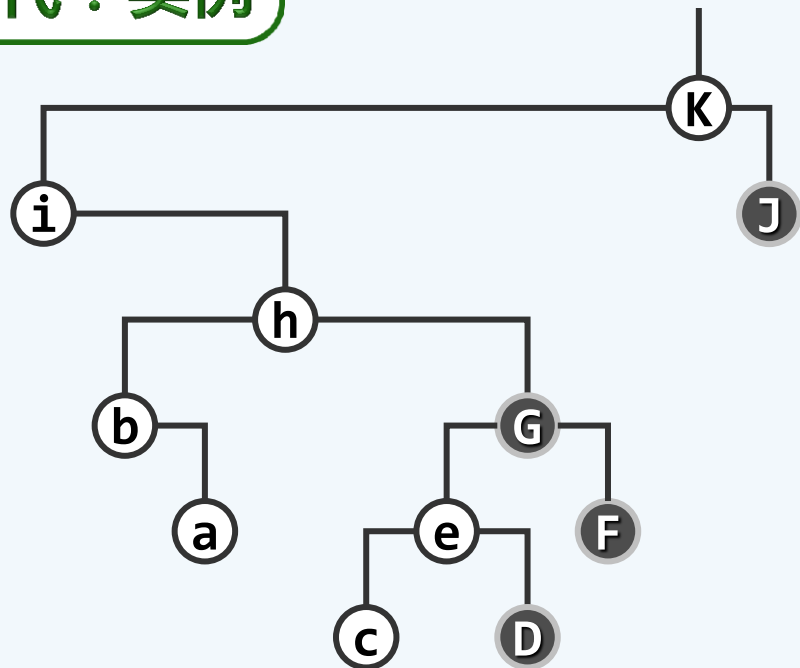
```
    }
```

```
}
```

迭代：实例



迭代：实例



迭代：正确性

❖ 可归纳证明：

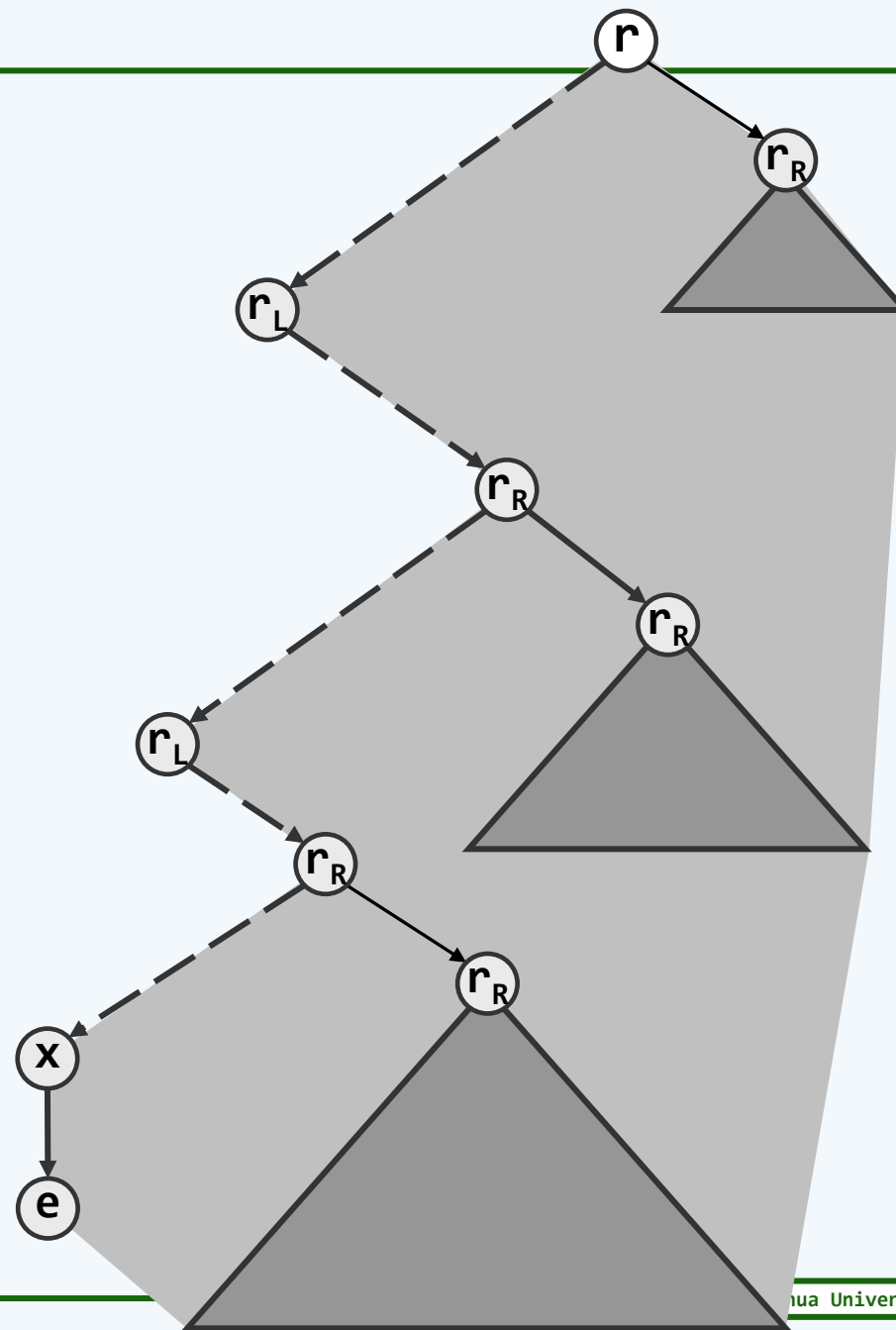
每个节点出栈后

以之为根的子树已经**完全遍历**，而且

其**右兄弟** r （如果存在）就在**栈顶**

❖ 于是，为“递归”遍历 r 对应的子树，只需
从 r 出发...

highest leaf
visible from left



迭代：效率

❖ 是否 $O(n)$ ，取决于以下条件

1) 每次迭代，都有一个节点出栈并被访问

//满足

2) 每个节点入栈一次且仅一次

//满足

3) 每次迭代只需 $O(1)$ 时间

//不再满足，因为...

❖ 单次调用`gotoHLVFL()`

就可能需要 $\Omega(n)$ 时间

❖ 既然如此，难道总体将需要... $O(n^2)$ 时间？

❖ 同样地，事实上这个界远远不紧...

❖ 空间？