

二叉树

先序遍历：算法A

05 - E7

真君曰：“昔吕洞宾居庐山而成仙，鬼谷子居云梦而得道，今或无此吉地么？” 璞曰：“有，但当遍历耳。”

邓俊辉

deng@tsinghua.edu.cn

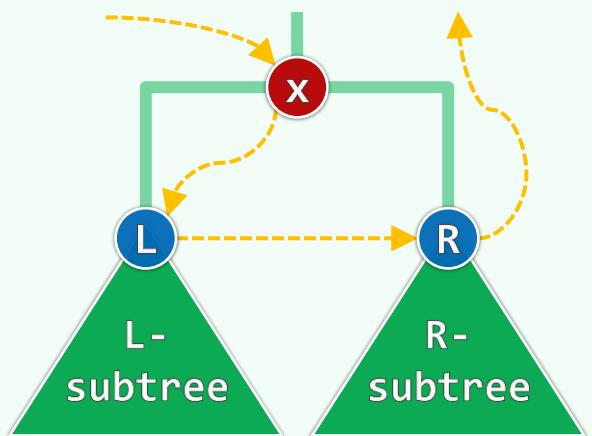
遍历

按照某种次序访问树中各节点，每个节点被访问恰好一次： $T = L \cup X \cup R$

$x | [L] | [R]$

先序

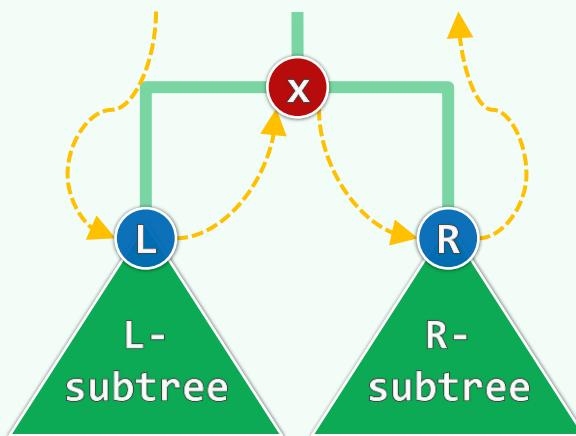
preorder



$[L] | x | [R]$

中序

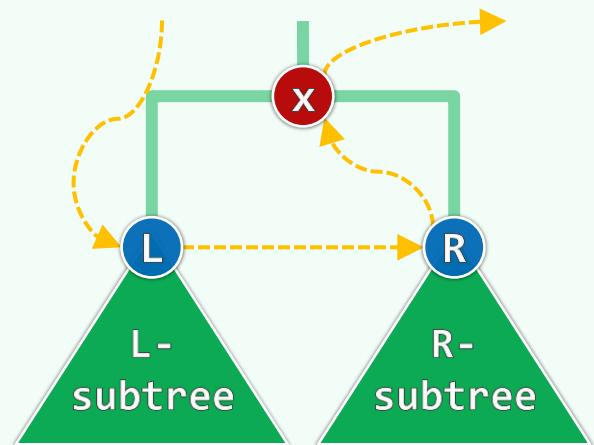
inorder



$[L] | [R] | x$

后序

postorder



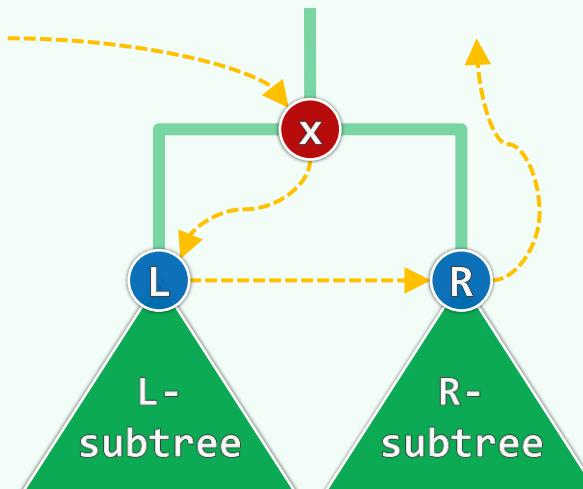
遍历结果 ~ 遍历过程 ~ 遍历次序 ~ 遍历策略

递归实现

❖ 应用：先序输出文件树结构：`c:\> tree.com c:\windows`

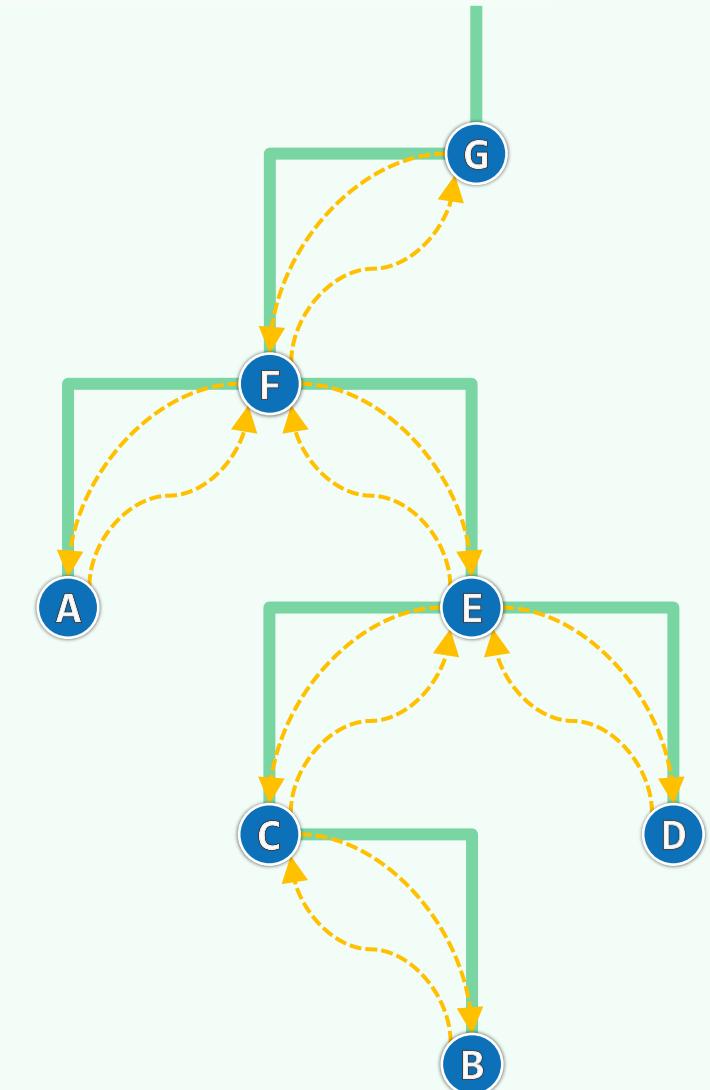
❖ template <typename T, typename VST>

```
void traverse( BinNodePosi(T) x, VST & visit ) {  
    if ( ! x ) return;  
  
    visit( x->data );  
  
    traverse( x->lc, visit );  
  
    traverse( x->rc, visit );  
}
```



$$\diamond T(n) = \mathcal{O}(1) + T(a) + T(n - a - 1) = \mathcal{O}(n)$$

❖ 挑战：不依赖递归机制，能否实现先序遍历？如何实现？效率如何？



思路

❖ 先序遍历任一二叉树T的过程，无非是

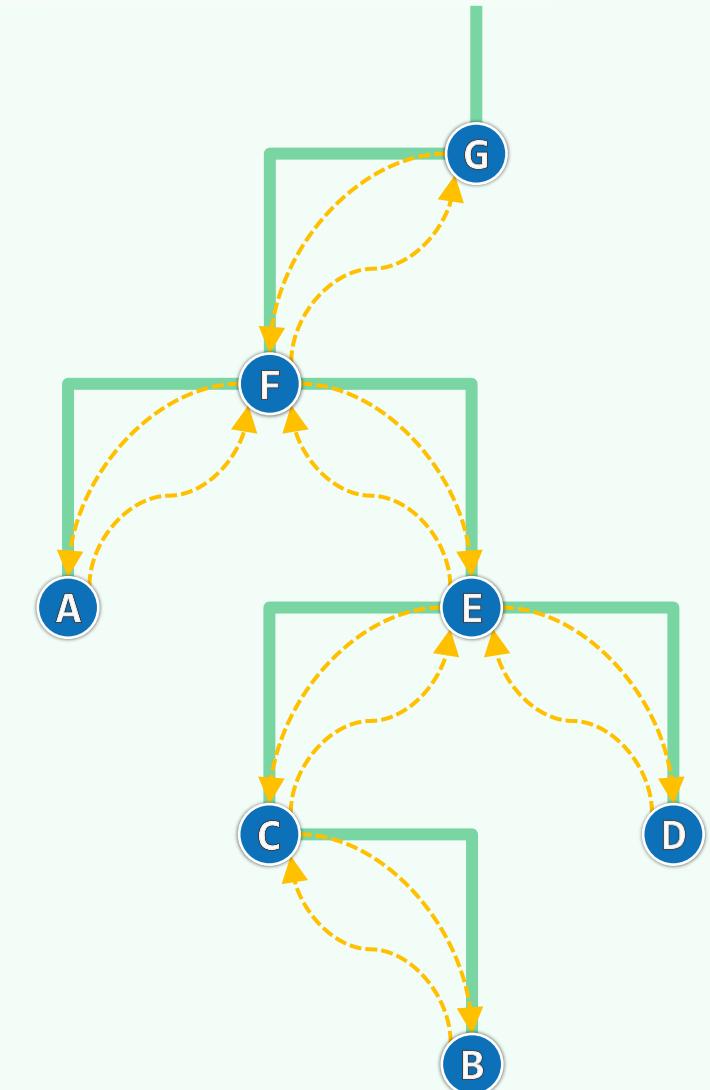
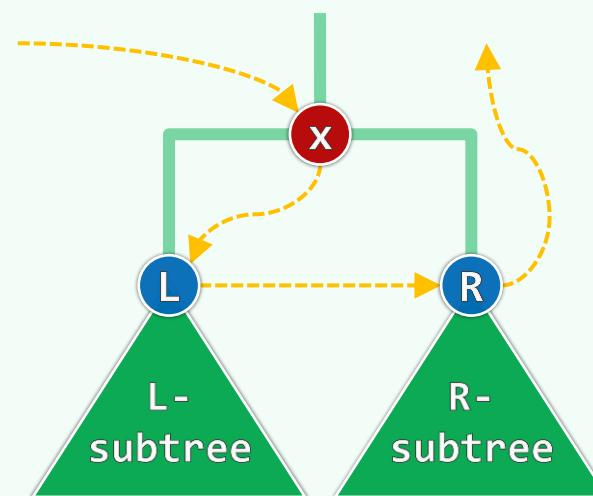
- 先访问根节点 x
- 再先后递归地遍历 T_L 和 T_R

❖ 递归实现中

对左、右子树的递归遍历

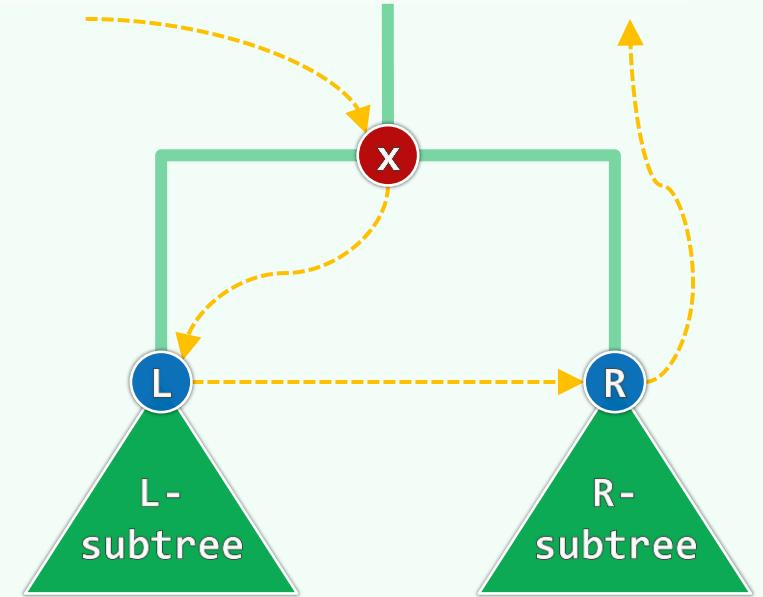
- 都类似于尾递归
- 故不难直接消除

❖ 思路：二分递归 \rightarrow 迭代 + 单递归 \rightarrow 迭代 + 栈

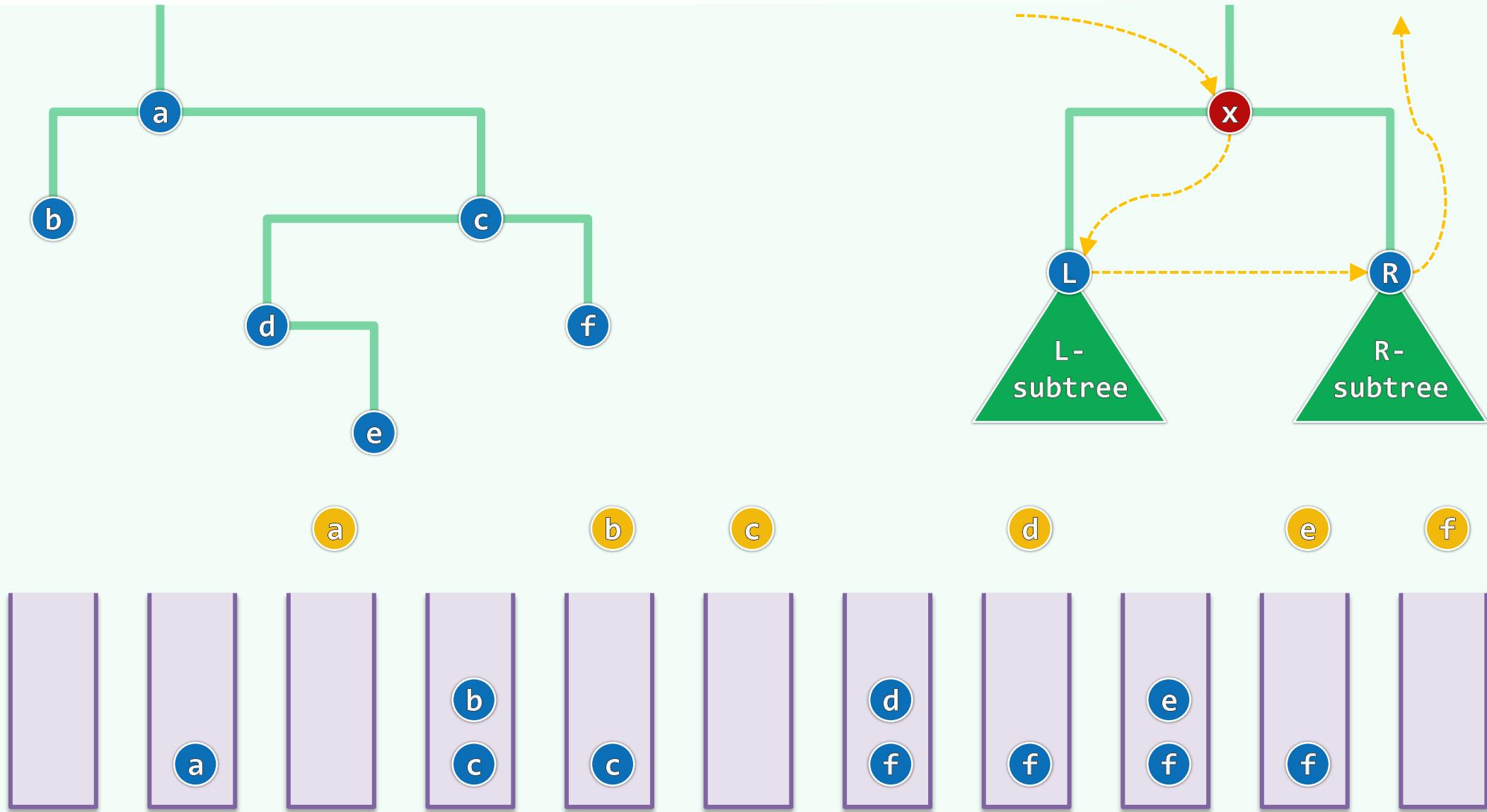


迭代算法A

```
template <typename T, typename VST>
void travPre_I1( BinNodePosi(T) x, VST & visit ) {
    Stack < BinNodePosi(T) > S; //辅助栈
    if (x) S.push( x ); //根节点入栈
    while ( ! S.empty() ) { //在栈变空之前反复循环
        x = S.pop(); visit( x->data ); //弹出并访问当前节点
        if ( HasRChild( *x ) ) S.push( x->rc ); //右孩子先入后出
        if ( HasLChild( *x ) ) S.push( x->lc ); //左孩子后入先出
    } //体会以上两句的次序
}
```



实例



评价

❖ 正确性

- 无遗漏：可按深度归纳证明，每个节点都会被访问到
- 根优先：任一子树中，根被访问后才会访问其它节点
- 左先右后：同一节点的左子树，先于右子树被访问

❖ 复杂度 $\mathcal{O}(n)$

- 每步迭代，都有一个节点出栈并被访问
- 每个节点入/出栈一次且仅一次
- 每步迭代只需 $\mathcal{O}(1)$ 时间

❖ 很遗憾，以上消除尾递归的思路不易推广，需要另寻他法...

