

05-G2

二叉树

后序遍历：迭代算法

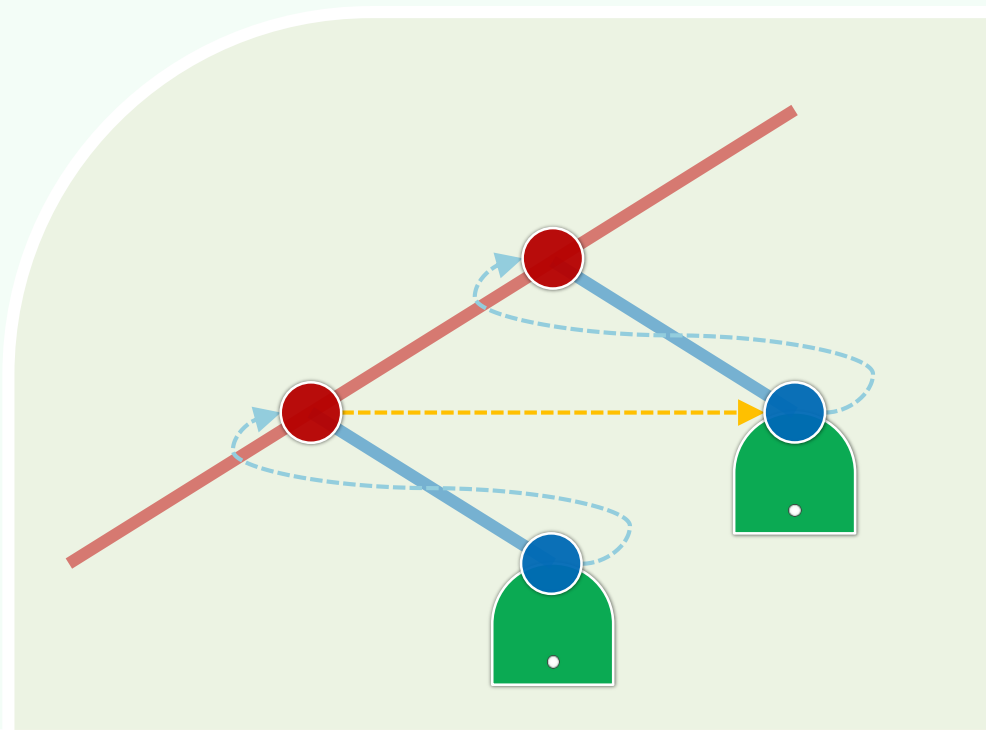
邓俊辉

deng@tsinghua.edu.cn

大宗百世不迁，小宗五世则迁

# 序曲

```
template <typename T> static void gotoLeftmostLeaf( 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->rc ); //优先入栈  
            S.push( x->lc ); //然后转向左孩子  
        } else //实不得已  
            S.push( x->rc ); //才转向右孩子  
    S.pop(); //返回之前，弹出栈顶的空节点  
}
```



全曲

```
template <typename T, typename V> void travPost_I( BinNodePosi<T> x, V & visit ) {  
  
    Stack < BinNodePosi<T> > S; //辅助栈  
  
    if ( x ) S.push( x ); //根节点首先入栈  
  
    while ( ! S.empty() ) { //x始终为当前节点  
        if ( S.top() != x->parent ) //若栈顶非x之父（而为右兄），则  
            gotoLeftmostLeaf( S ); //在其右兄子树中找到最靠左的叶子  
        x = S.pop(); //弹出栈顶（即前一节点之后继）以更新x  
        visit( x->data ); //并随即访问之  
    }  
}
```

The diagram shows a binary tree with three internal nodes (red circles) and two leaf nodes (green shapes). The root node has a left child and a right child. The left child has a left child and a right child. The right child of the root has a right child. A red line traces the path from the root to the leftmost leaf. A blue dashed line shows the return path from the leaf back up to the root. A yellow arrow points from the leftmost leaf to its right sibling.

