## 二叉树

中序遍历:分析

邓俊辉 deng@tsinghua.edu.cn

他从哪条路来,必从哪条路回去,必不得来到这城

## 正确性: 数学归纳

❖ 假设算法可正确地遍历所有规模不超过n的二叉树

❖ 考查任一规模为n+1的二叉树,其根记作x 将藤之终点记作t, t之右孩子记作y

## ❖ 整个算法分为五个阶段:

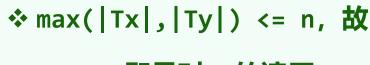
X1: goAlongVine( x )

T: visit( t = S.pop() )

Y1: goAlongVine( y )

Y2: traverse( Ty )

X2: traverse( Tx )



X1+X2 即是对Tx的遍历

Y1+Y2 即是对Ty的遍历

❖ 尽管前者被后者打断,但

在开始X2之前,等效于t和Ty未曾存在过

## 效率: 分摊分析

- ❖ goAlongVine()最多需调用 $\Omega(n)$ 次; 单次调用最多需做 $\Omega(n)$ 次push()
  - 既然如此,难道总体将需要...o(n²)时间?
  - 事实上这个界远远不紧,更精准的分析可给出♂(n)的界!
- ❖ 为此,需<mark>纵观整个遍历过程中所有对goAlongVine()的调用...</mark>
- ❖ 观察结论: 算法的时间复杂度, 渐近地不超过辅助栈的push()和pop()次数
  - pop(): 仅在主算法中执行; 每迭代一步恰好一次, 全程不过Ø(n)次
  - push(): 仅在goAlongVine()中执行;尽管次数不定,但累计应与pop()一样多 //Aggregate
- ❖ 更多的实现: travIn\_I2() + travIn\_I3() + travIn\_I4()