06-D3

二叉搜索树

AVL树：插入

邓俊辉

deng@tsinghua.edu.cn

名不正则言不顺，言不顺则事不成

单旋：黄色节点恰好存在其一

❖ 同时可有多个失衡节点

最低者g不低于x的祖父

❖ g经单旋调整后复衡

子树高度复原

❖ 更高祖先也必平衡

全树复衡

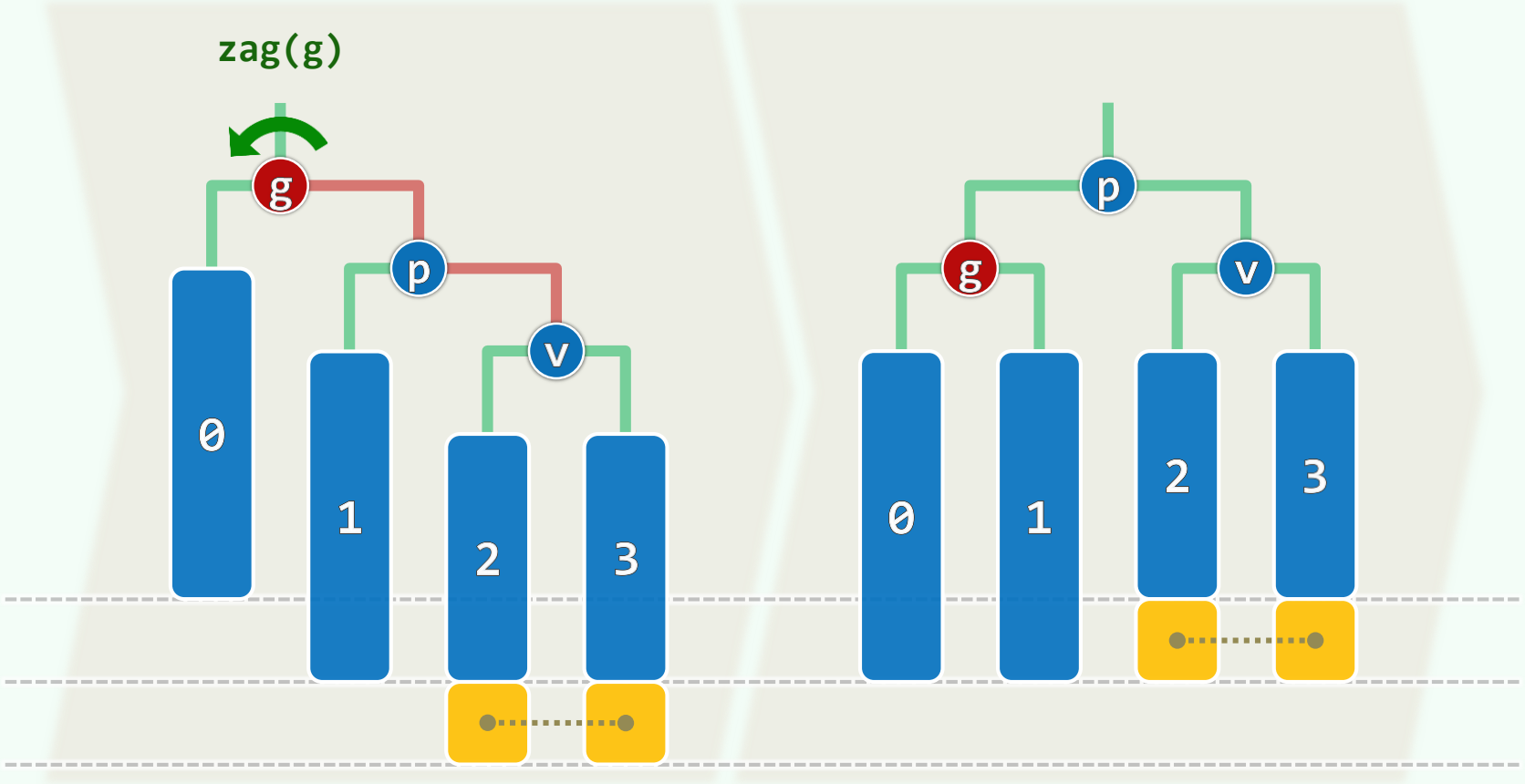
❖ 逐层上溯，便可找到g

❖ 确定名分：

- p = tallerChild(g)
- v = tallerChild(p)

❖ 无论p和v的方向是否一致

均可从容处理...



双旋

❖ 同时可有多个失衡节点

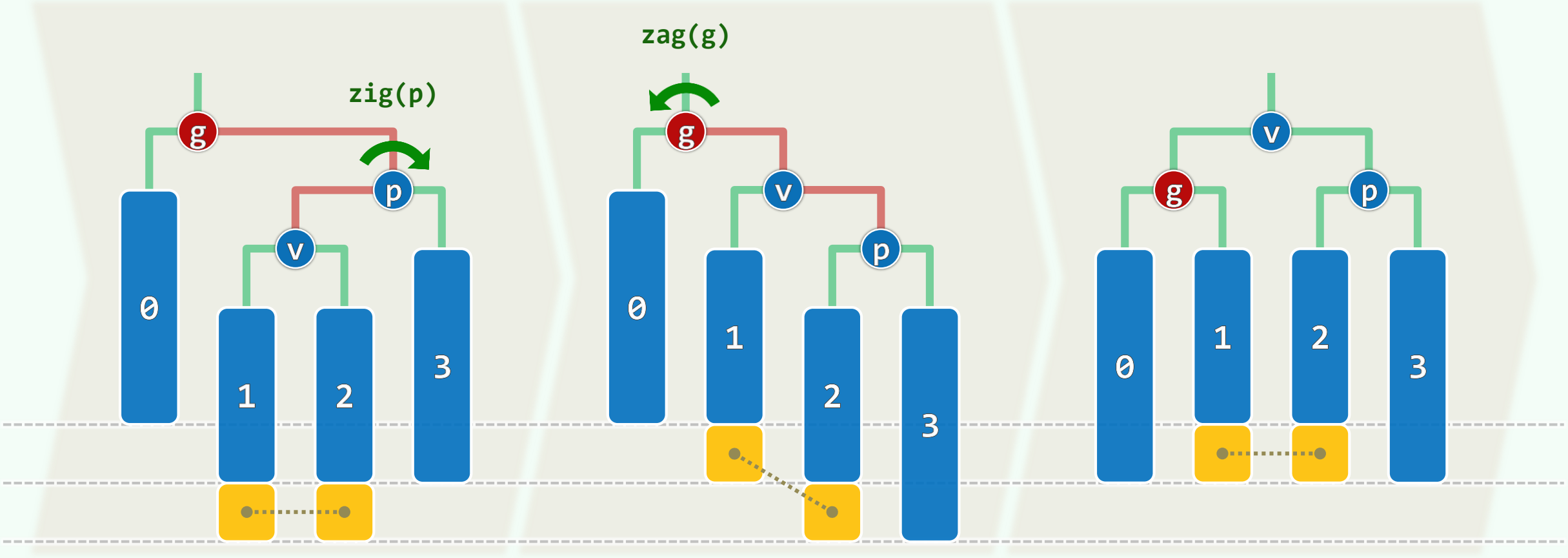
最低者g不低于x的祖父

❖ g经单旋调整后复衡

子树高度复原

❖ 更高祖先也必平衡

全树复衡



实现

```
template <typename T> BinNodePosi<T> AVL<T>::insert( const T & e ) {
   BinNodePosi<T> & x = search( e ); if ( x ) return x; //插入失败
   BinNodePosi<T> xx = x = new BinNode<T>( e, _hot ); _size++; //则创建新节点
   ---------------------------------------------------------------
   for ( BinNodePosi<T> g = _hot; g; g->updateHeight(), g = g->parent ) //逐层上溯
      if ( ! AvlBalanced( g ) ) { //一旦发现失衡祖先g，则
         rotateAt( tallerChild( tallerChild( g ) ) ); //通过调整恢复平衡
         break; //并随即终止（局部子树复衡后，高度必然复原；所有祖先亦必复衡）
      }
   ---------------------------------------------------------------
   return xx; //插入成功
} //至多会做O(1)次调整
```