

二叉搜索树

AVL树：插入

08 - D3

邓俊辉

deng@tsinghua.edu.cn

没有什么是一次旋转解决不了的

如果有，那就两次

单旋

❖ 同时可有多个失衡节点

最低者g不低于x的祖父

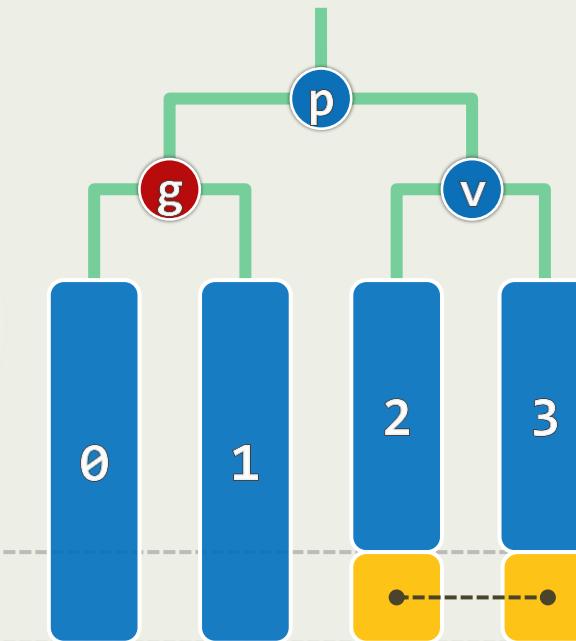
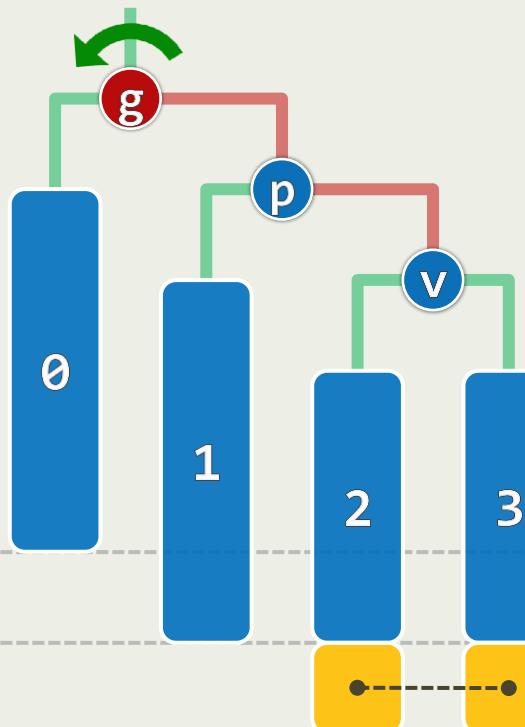
❖ g经单旋调整后复衡

子树高度复原

❖ 更高祖先也必平衡

全树复衡

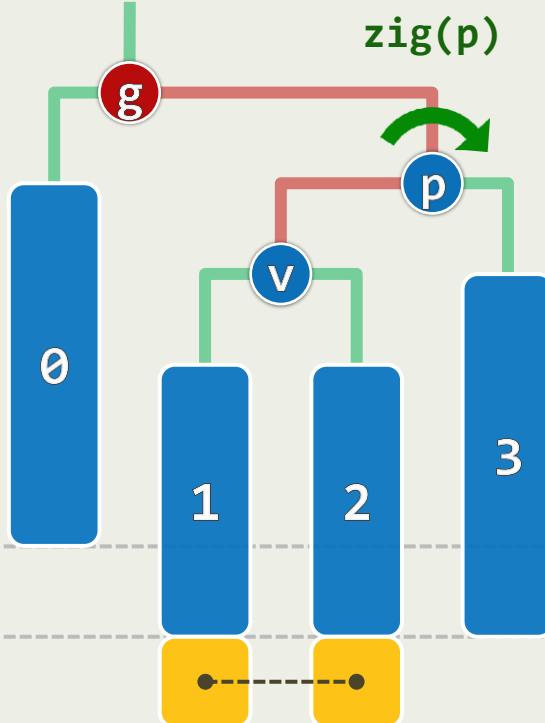
zag(g)



双旋

❖ 同时可有多个失衡节点

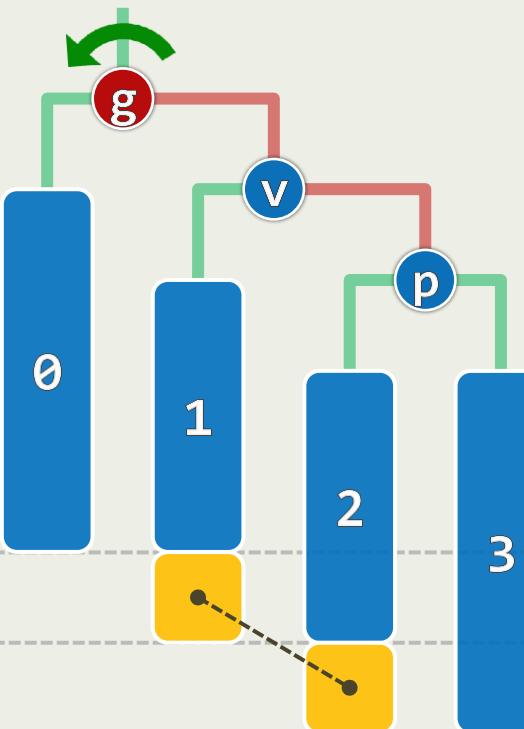
最低者g不低于x的祖父



❖ g经单旋调整后复衡

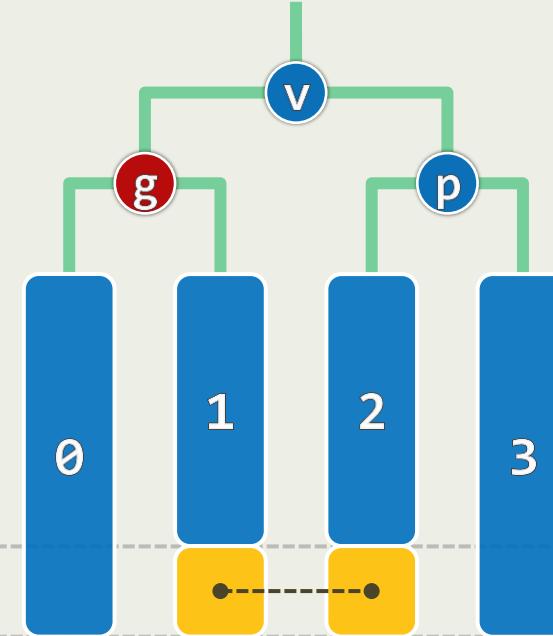
子树高度复原

zag(g)



❖ 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; //若目标尚不存在  
    BinNodePosi(T) xx = x = new BinNode<T>( e, _hot ); _size++; //则创建新节点  
    // 此时，若x的父亲_hot增高，则祖父有可能失衡。故以下从_hot起，向上逐层检查各代祖先  
    for ( BinNodePosi(T) g = _hot; g; g = g->parent )  
        if ( ! AvlBalanced( *g ) ) { //一旦发现g失衡，则通过调整恢复平衡  
            FromParentTo(*g) = rotateAt( tallerChild( tallerChild( g ) ) );  
            break; //g复衡后，局部子树高度必然复原；其祖先亦必如此，故调整结束  
        } else //否则（在依然平衡的祖先处），只需简单地  
            updateHeight( g ); //更新其高度（平衡性虽不变，高度却可能改变）  
    return xx; //返回新节点：至多只需一次调整  
}
```