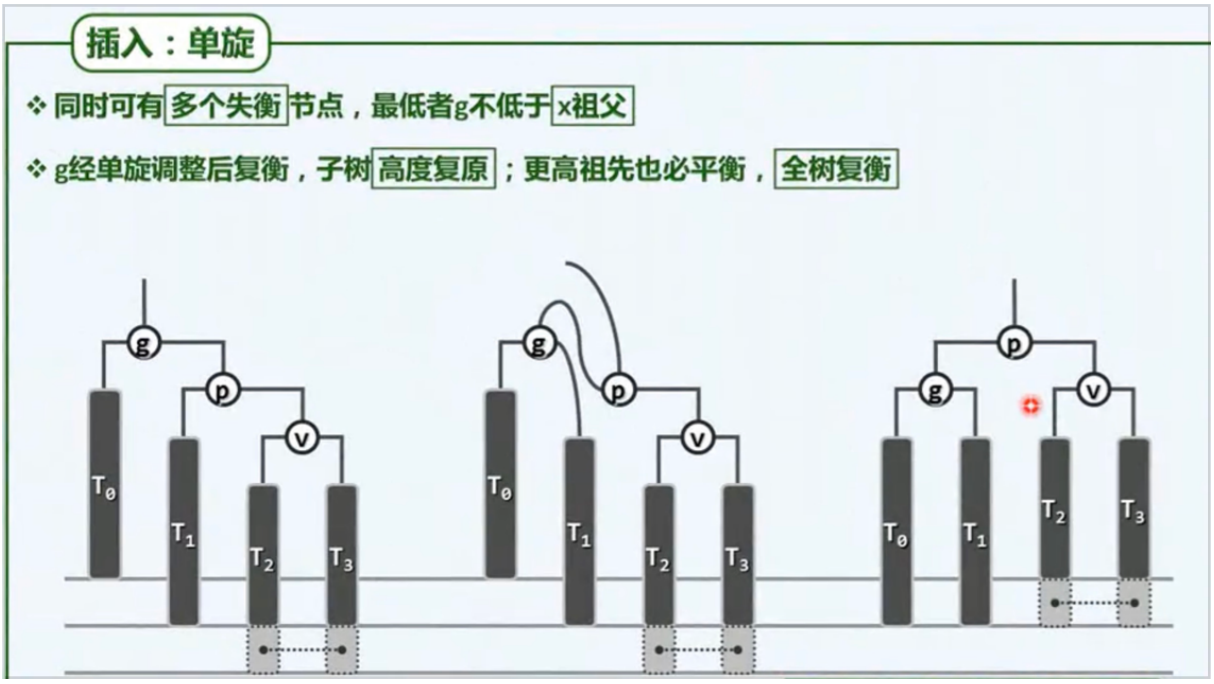
08D3 插入

#数据结构邓神

单旋



旋转套路：注意旋转后叶子各个节点左右前后顺序不发生变化，始终是把处于中间节点向上移作为跟节点，其余保持相对次序即可

两个虚线连接是代表两个只能插入一个

插入新节点之前，高度为2，插入的瞬间高度变化为3，平衡后高度又回到2表示回归了原始的状态，因为平衡因子只跟高度相关。

一次zag旋转，只设计到局部的常数个节点，时间消耗为O(1)

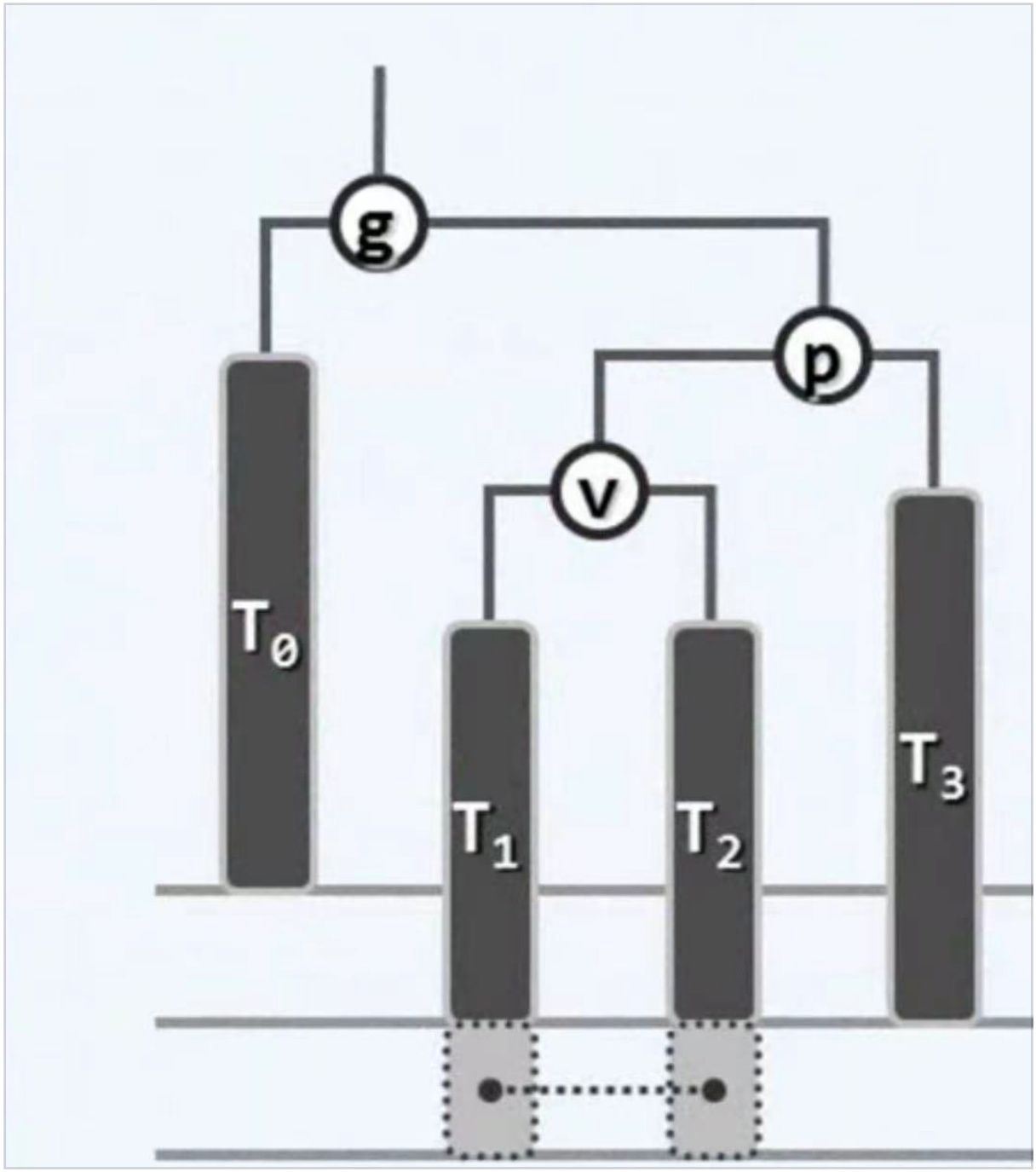
这种旋转是一种特定的情况：特点是gpv三个节点的方向一致，一方都是另一方的右/左子树

同时向右 叫做 zagzag

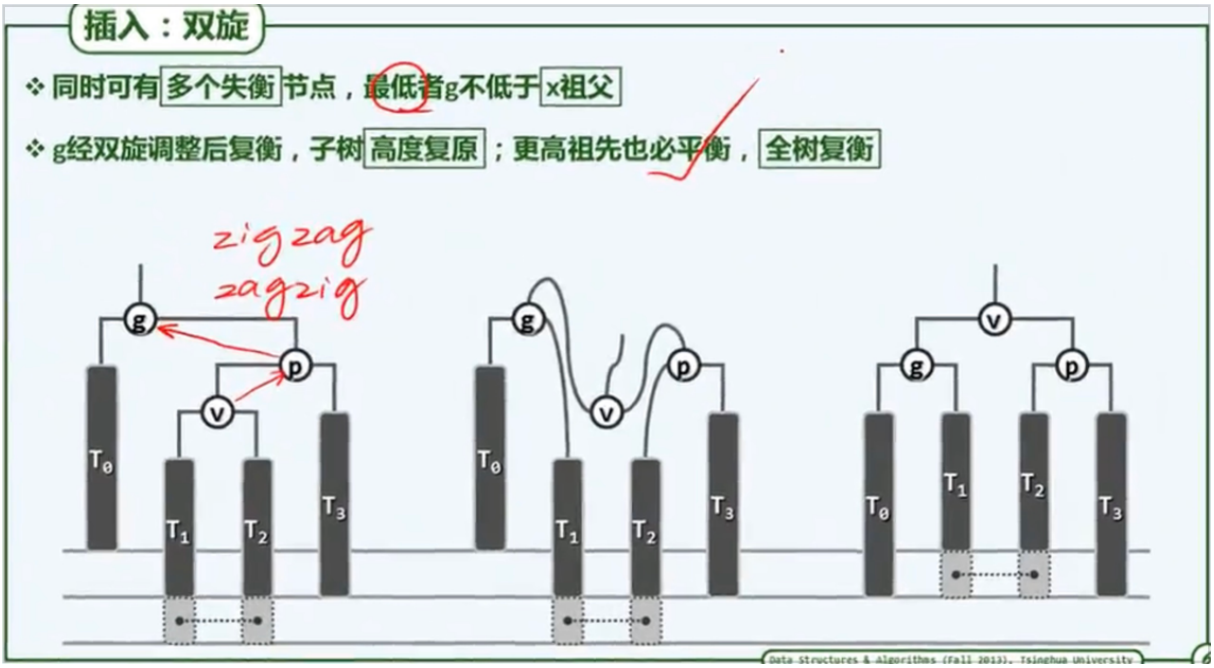
一致向左 叫做 zigzig

不难发现如果方向不一致要不就是 zagzig 或者 zigzag

双旋

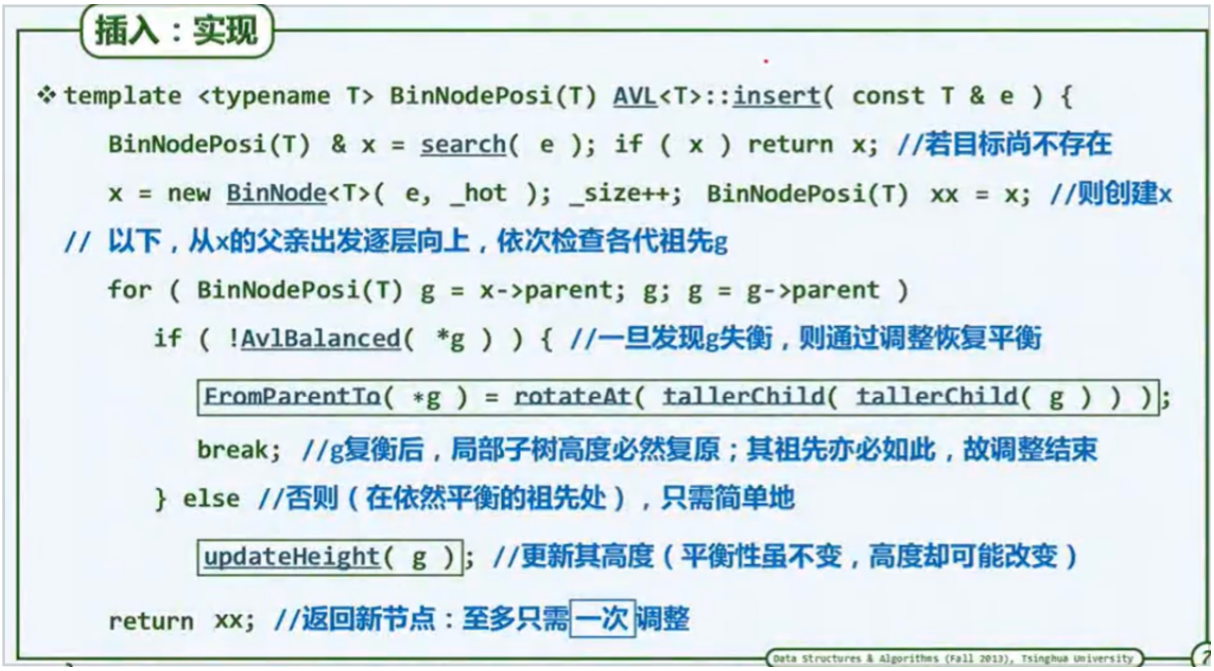


看到这种样子我们能否把他转换为单旋转那种朝向一个方向后在交由单旋转处理呢?



RPReplay_Final163913050...
2021年12月10日

实现




```
template <typename T> BinNodePosi<T> AVL<T>::insert(const T & e){
    BinNodePosi<T> & x = search(e);
    if (x){
        return x;
    }
    x = new BinNode<T>(e,_hot);
    _size++;
```

```
    BinNodePosi<T> xx = x;
    for (BinNodePosi<T> g = x->parent; g ; g = g->parent) {
        if (!AvlBalanced(*g)) {
            FromParentTo(*g) = rotateAt(tallerChild(tallerChild(g)));
                                                // 两个taller就是求祖父的位置
            break;
        }
        else {
            updateHeightAbove(g);
        }
    }
    return xx;
}
```