

08-A2

高级搜索树

伸展树：双层伸展

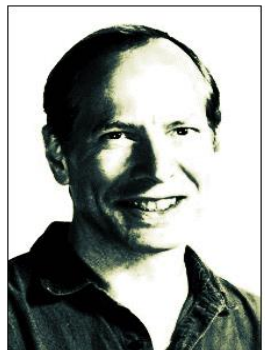
邓俊辉

deng@tsinghua.edu.cn

贾政道：“不用全打开，怕叠起来倒费事。” 詹光便与冯紫英一层一层折好收拾。

# 双层伸展

## ❖ Self-Adjusting Binary Trees

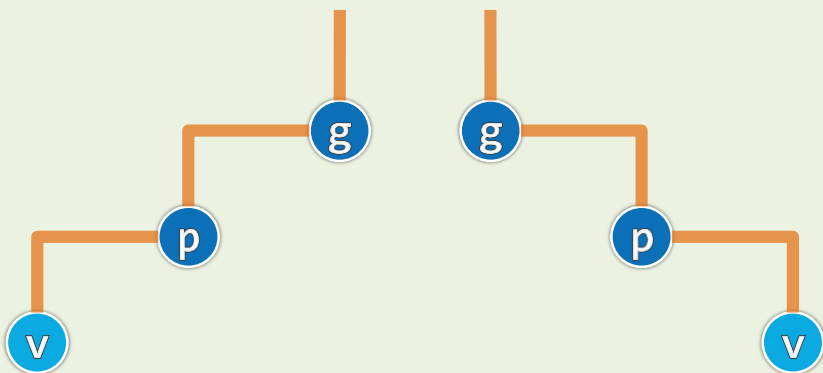


D. D. Sleator

R. E. Tarjan

J. ACM, 32:652-686, 1985

## ❖ 构思的精髓：向上追溯两层，而非一层



## ❖ 反复考察祖孙三代：

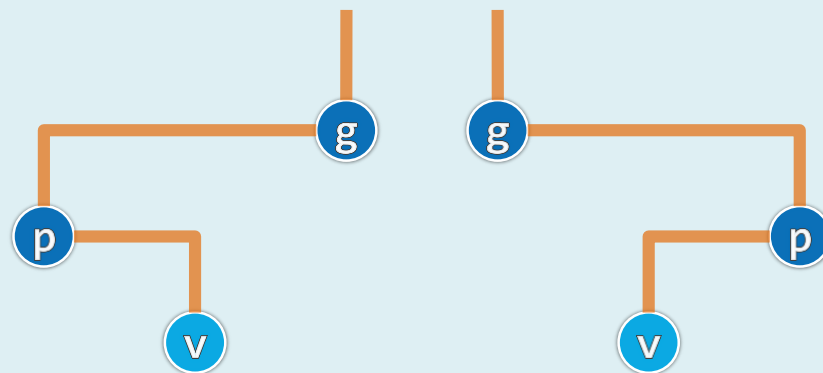
$g = \text{parent}(p)$ ,  $p = \text{parent}(v)$ ,  $v$

## ❖ 根据它们的相对位置，经两次旋转

使v上升两层，成为（子）树根

## ❖ 如此，性能的确会有改善？

## ❖ 具体地，应该如何旋转？



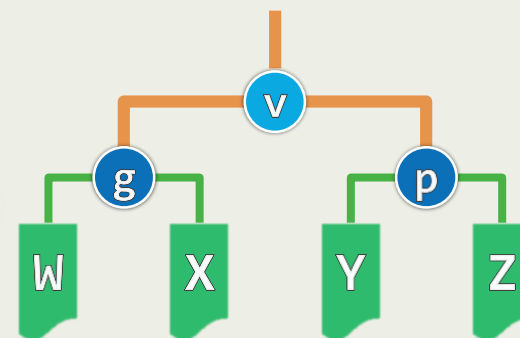
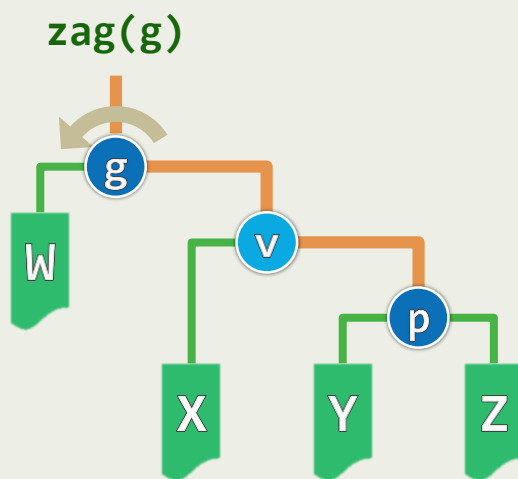
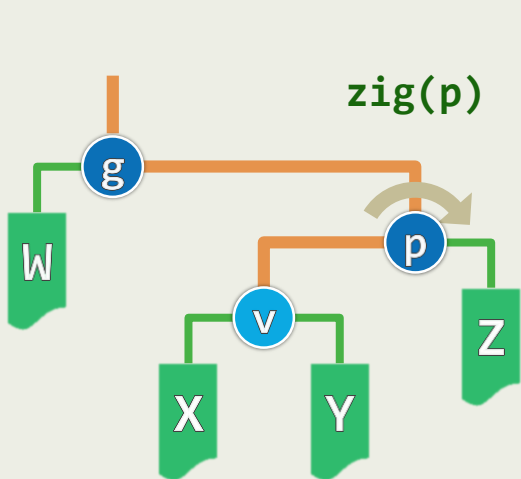
## zig-zag / zag-zig

❖ 此时的v按中序遍历次序**居中**

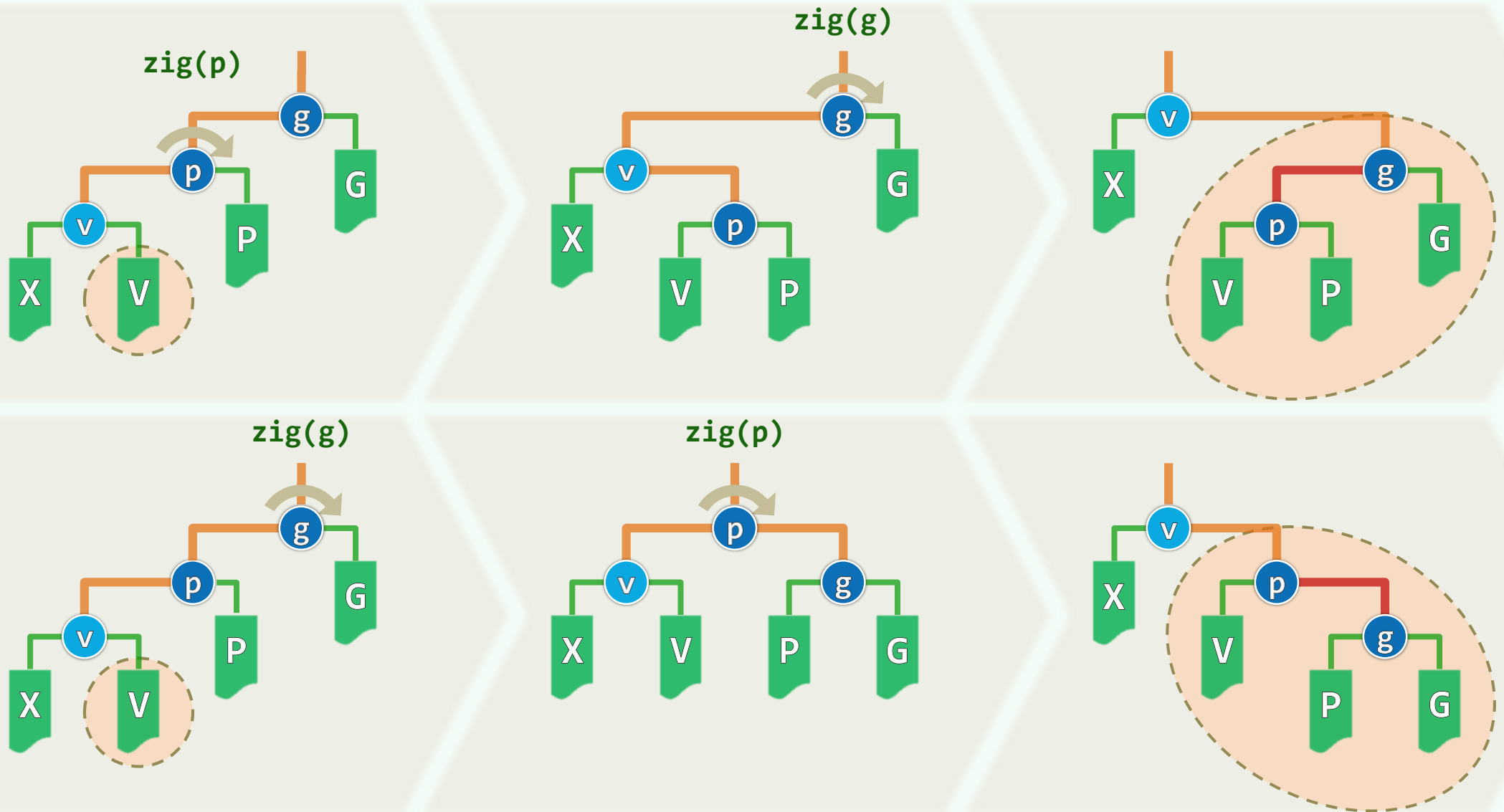
❖ 如此调整的效果，与**逐层调整**别无二致！

❖ 故若欲使之成为**根**，最终无非**一种姿势**

❖ 难道，就这样平淡无奇？



## zig-zig / zag-zag



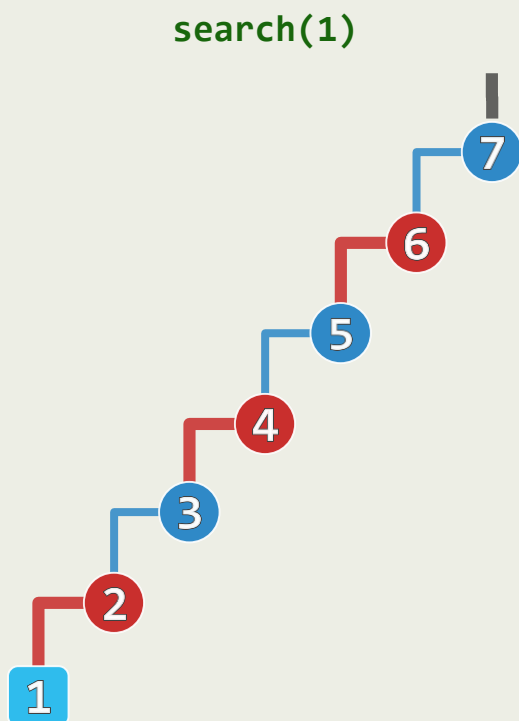
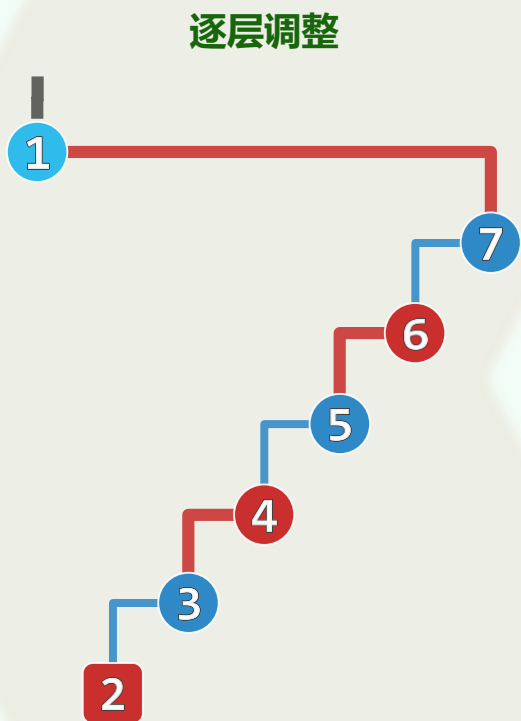
# zig-zig / zag-zag

❖ 节点访问之后，对应路径的长度随即**折半**

❖ 最坏情况不致持续发生！

//含羞草般的折叠效果

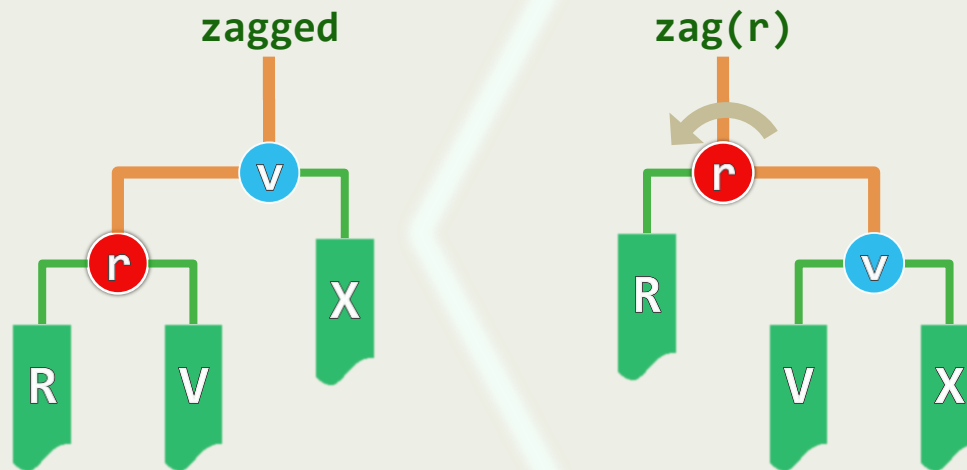
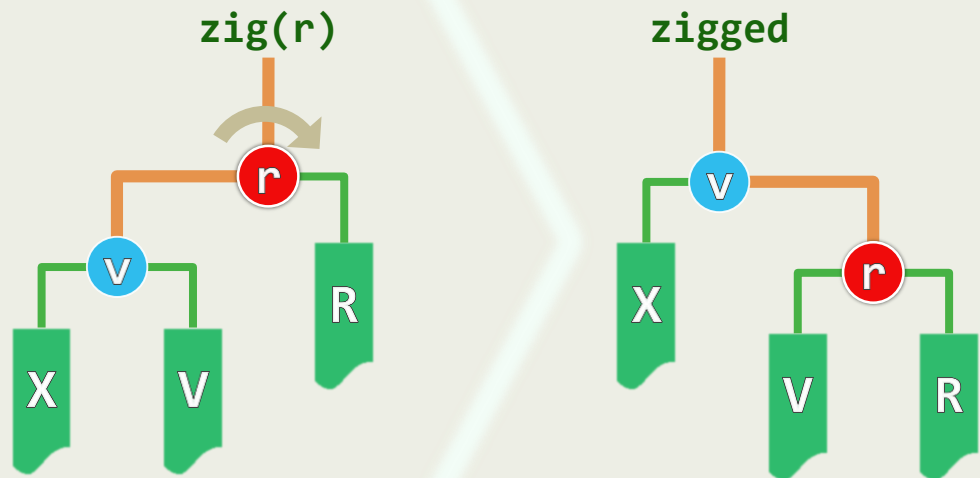
习题[8-2]：伸展操作分摊仅需  $\mathcal{O}(\log n)$  时间



## zig / zag

❖ 要是v只有父亲，没有祖父呢？

❖ 此时必有 $v.parent() == T.root()$



❖ 只需做**单次**旋转： $zig(r)$ 或 $zag(r)$

❖ 好在，这种情况至多（在最后）出现**一次**