08-A1

高级搜索树

伸展树：逐层伸展

我要一步一步往上爬

在最高点乘着叶片往前飞

邓俊辉

deng@tsinghua.edu.cn

局部性/Locality：刚被访问过的数据，极有可能很快地再次被访问

❖ 这一现象在信息处理过程中屡见不鲜...

[Pfaff-2004] Performance Analysis Of BSTs In System Software

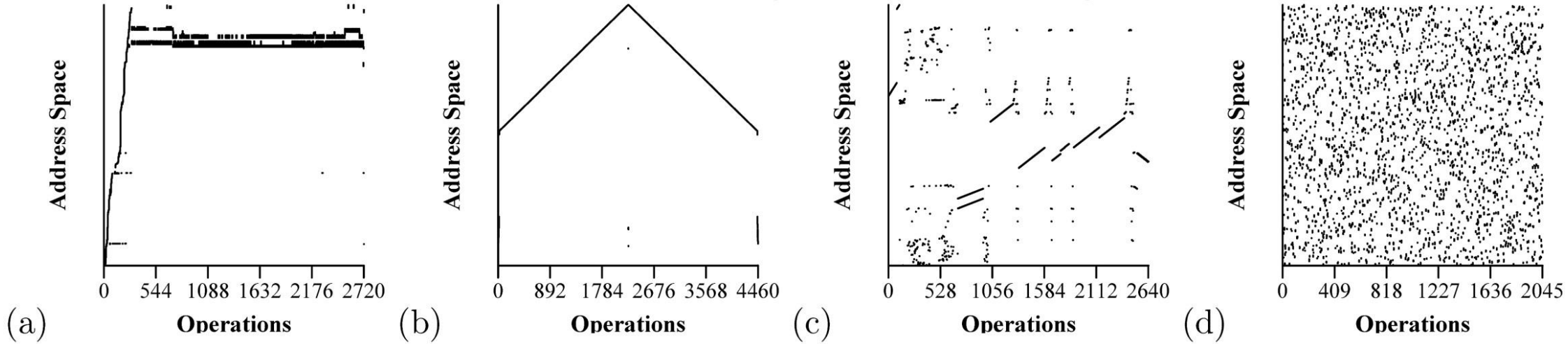


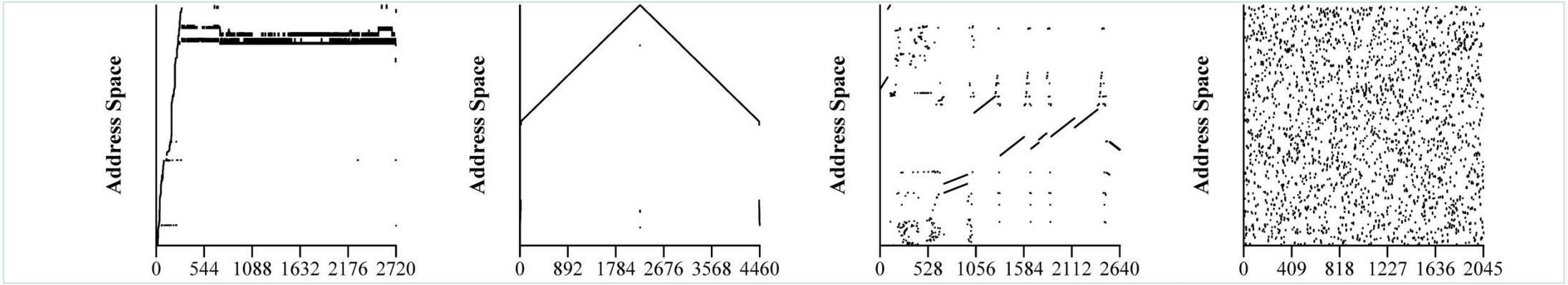
Figure 2: Call sequences in (a) Mozilla 1.0, (b) VMware GSX Server 2.0.1, (c) squid running under User-Mode Linux 2.4.18.48, and (d) random test sets. Part (b) omits one `mmap-munmap` pair for memory region 0x20000000 to 0x30000000 and (c) omits address space gaps; the others are complete.

BST的局部性

- 时间：刚被访问过的节点，极有可能很快地再次被访问

  空间：下一将要访问的节点，极有可能就在刚被访问过节点的附近

- 对AVL连续的m次查找（m >> n），共需 $\mathcal{O}(m \cdot \log n)$ 时间——能否利用局部性加速?



- 自适应链表：节点一旦被访问，随即移动到最前端

  模仿：BST的节点一旦被访问，随即调整到树根

- 难点：如何实现这种调整？调整过程自身的复杂度如何控制?

逐层伸展

- 节点v一旦被访问

  随即被推送至根

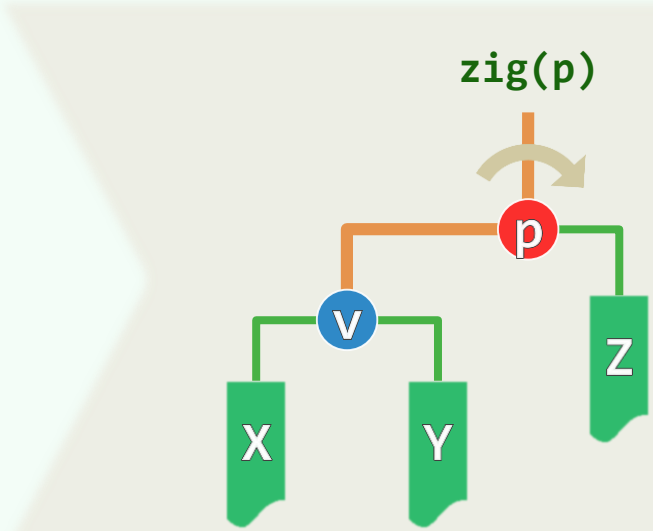
- 与其说“推”，不如说“爬”

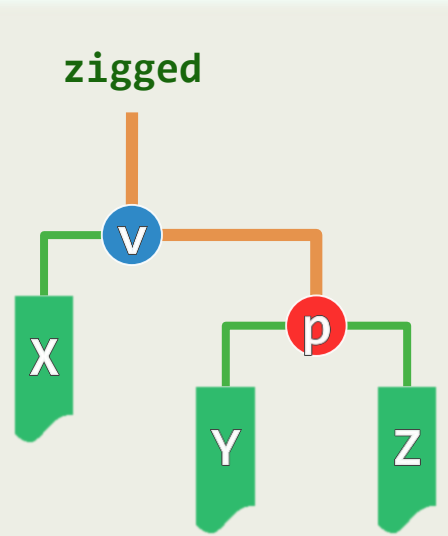
  一步一步地往上爬

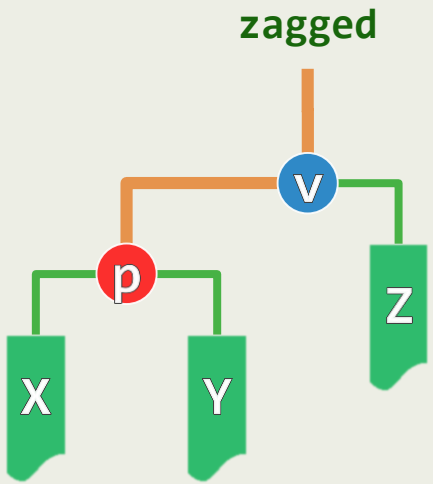
- 自下而上，逐层旋转

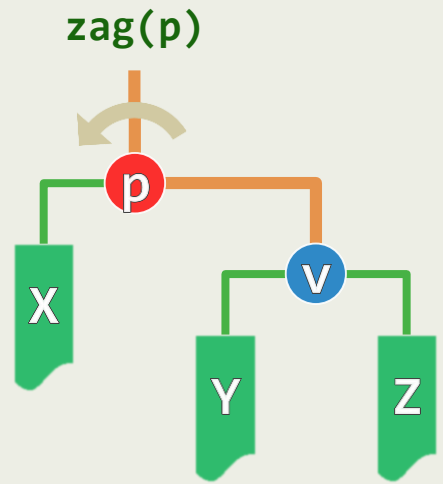
  - zig(v->parent)

  - zag(v->parent)









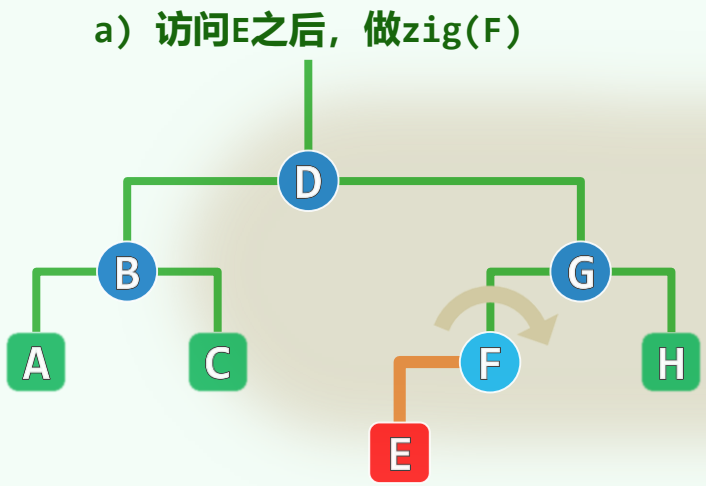
实例

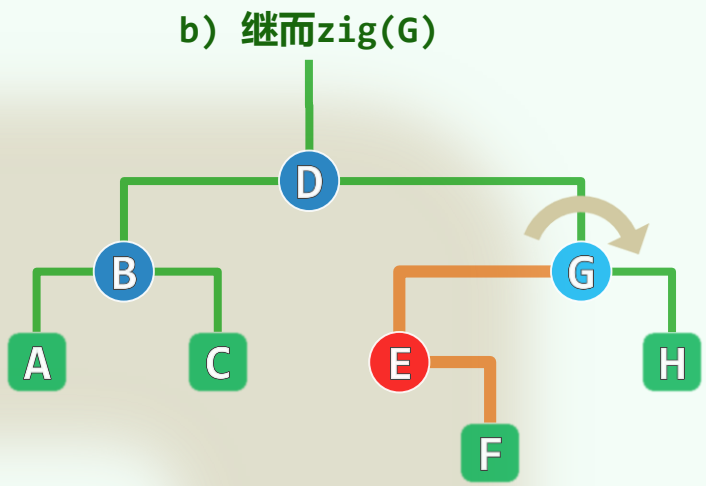
- 伸展过程的效率

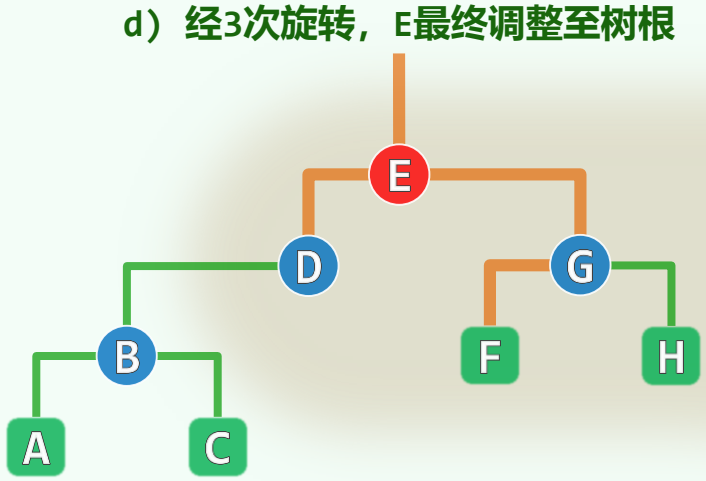
  是否足够地高?

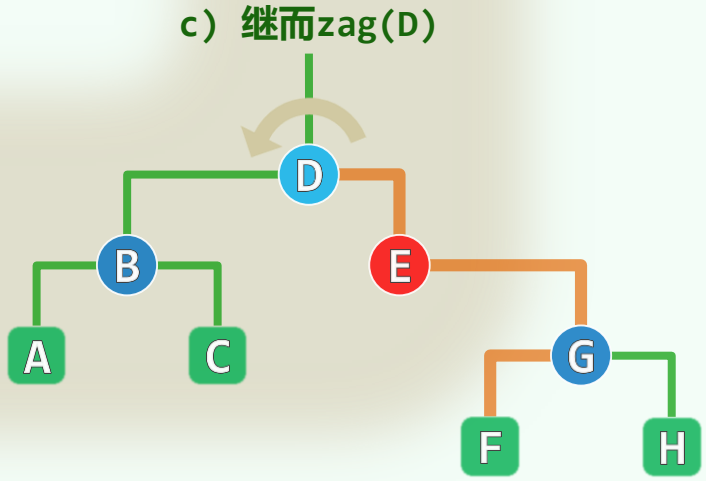
- 这取决于

  - 树的初始形态和
  - 节点的访问次序









最坏情况

- 旋转次数

  呈周期性的算术级数

- 每一周期累计$\Omega(n^2)$

  分摊 $\Omega(n)$

- 怎么破?