

高级搜索树

伸展树：逐层伸展

10.-A2

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...

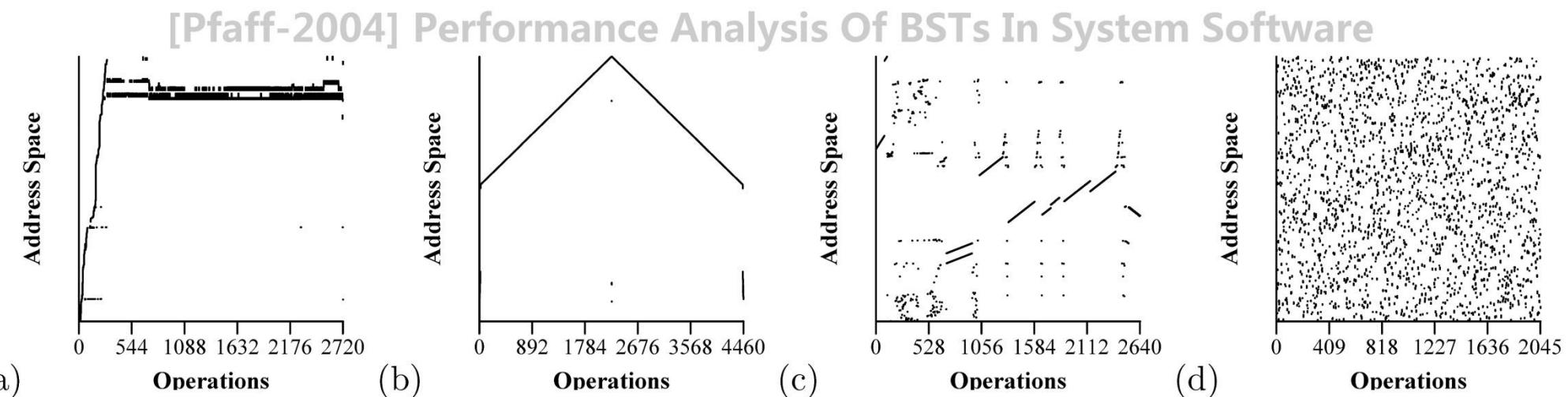


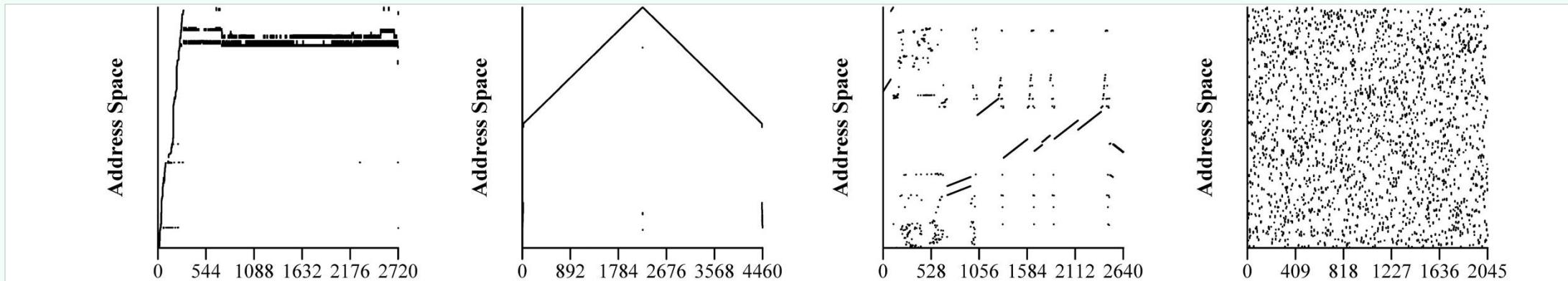
Figure 2: Call sequences in (a) Mozilla 1.0, (b) VMware GSX Server 2.0.1, (c) squid running under User-Mode Linux 2.4.18.48, and (d) random test sets. Part (b) omits one `mmap-munmap` pair for memory region `0x20000000` to `0x30000000` and (c) omits address space gaps; the others are complete.

局部性

❖ BST：刚被访问过的**节点**，极有可能**很快地再次被访问**

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** ($m \gg n$)，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

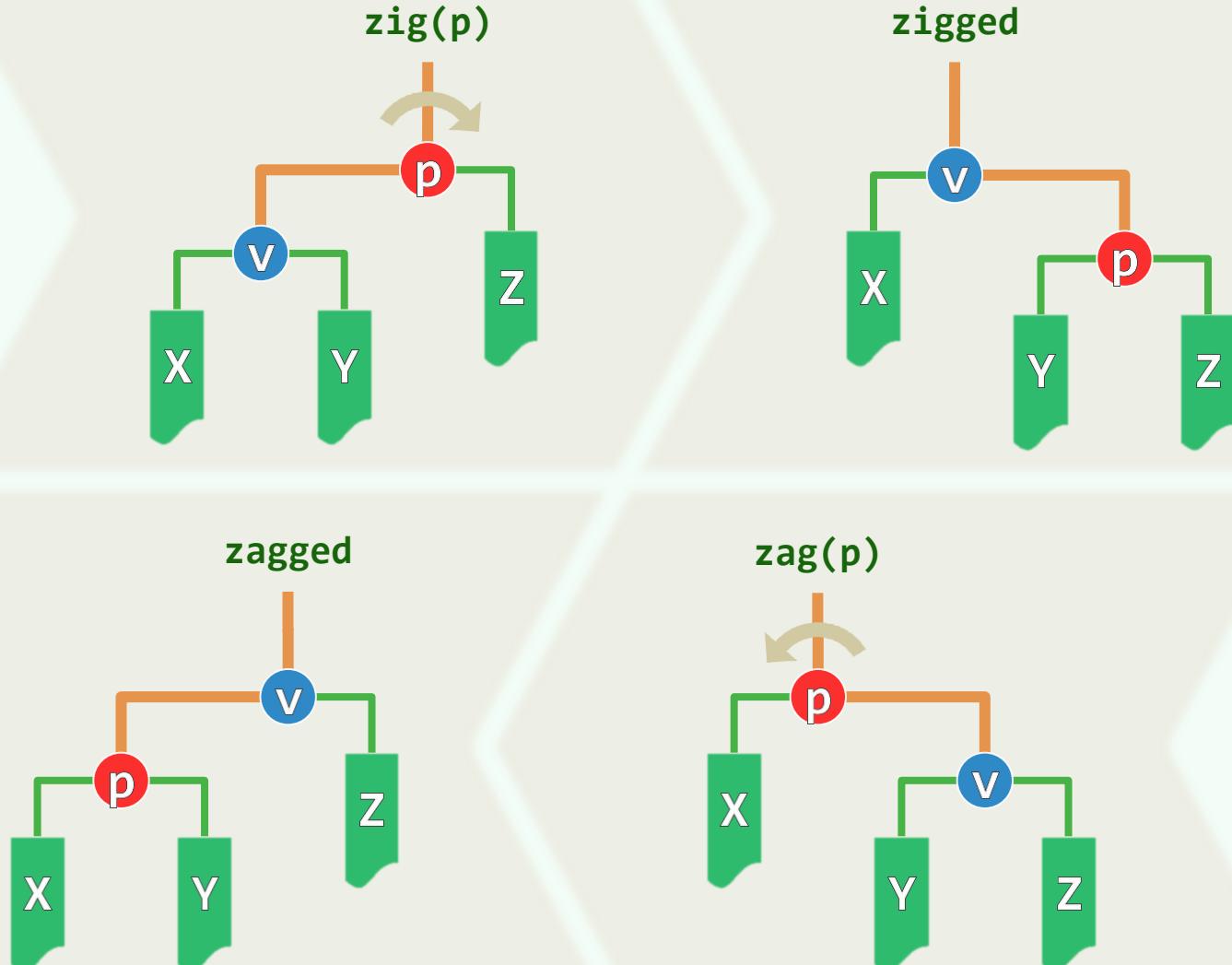
随即被**推送至根**

❖ 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

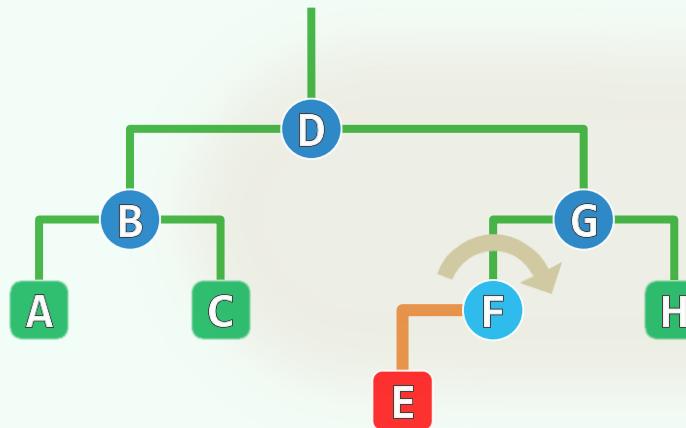
是否足够地高？

◆ 这取决于

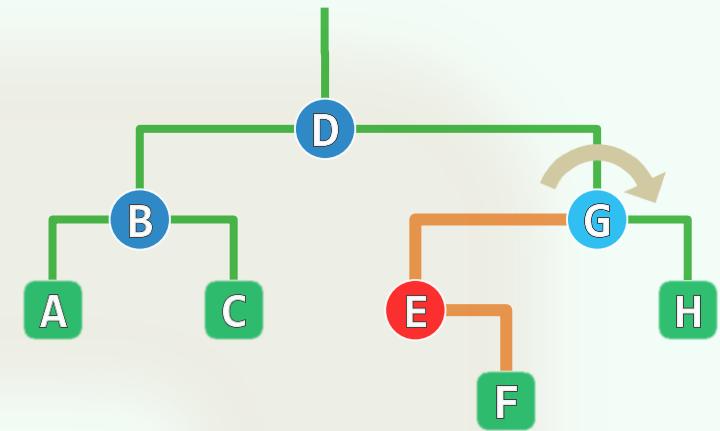
- 树的初始形态和

- 节点的访问次序

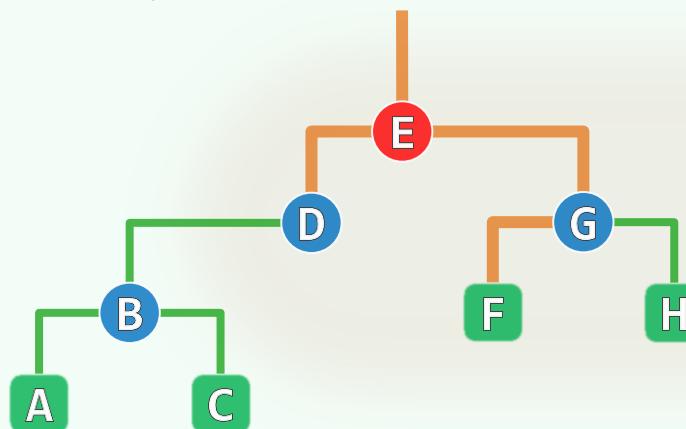
a) 访问E之后，做zig(F)



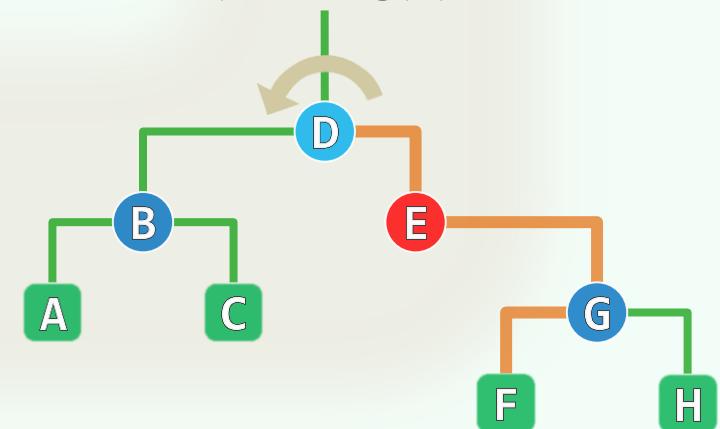
b) 继而zig(G)



d) 经3次旋转，E最终调整至树根



c) 继而zag(D)



最坏情况

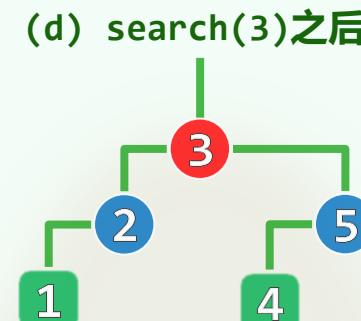
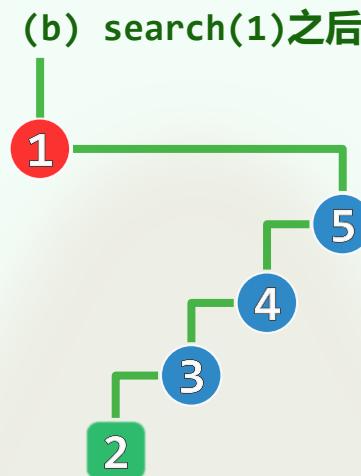
◆ 旋转次数

呈周期性的算术级数

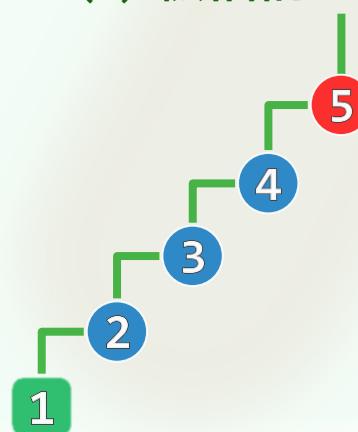
◆ 每一周期累计 $\Omega(n^2)$

分摊 $\Omega(n)$

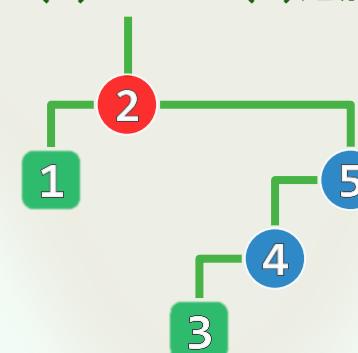
◆ 怎么破？



(a) 初始结构



(c) search(2)之后



(e) search(4)之后

