

## 10 高级搜索树 | 10A1 伸展树 逐层伸展

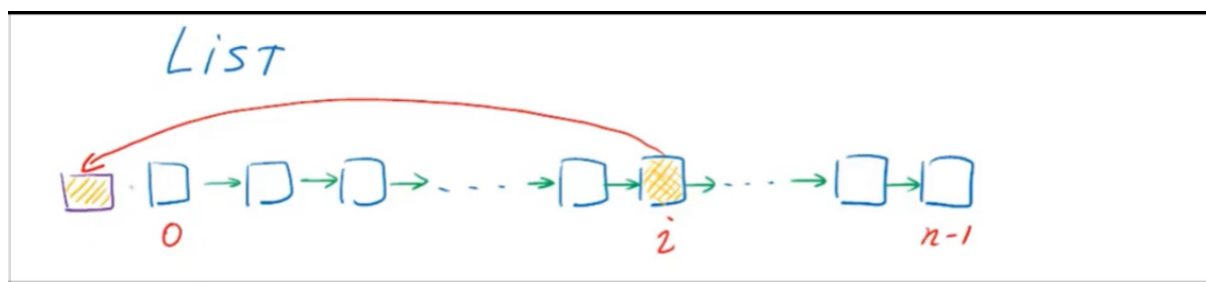
#数据结构邓神

### 局部性

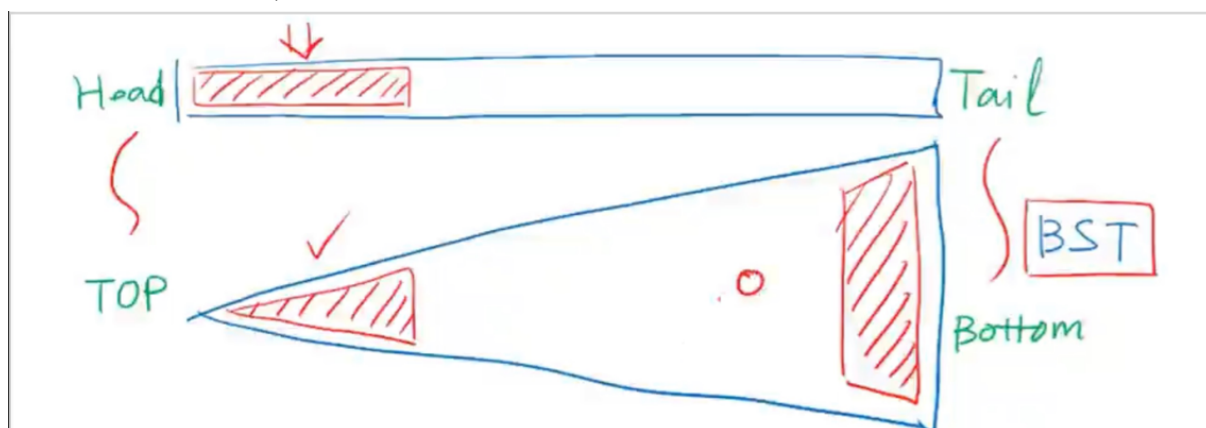
**局部性**

- ❖ **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 //BST就是这样的例子
- ❖ **BST**: 刚刚被访问过的节点, 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
- ❖ 连续的 $m$ 次查找 ( $m \gg n = |BST|$ ), 采用AVL共需 $O(m \log n)$ 时间
- ❖ 利用局部性, 能否更快? //仿效自适应链表

### 自适应调整



一旦访问了那个列表中的某个元素, 我们就直接把那个元素移动到链表的最前端。  
这样随着访问的进行, 排在越前面的就是访问的频率越多也能降低平均的访问的时间



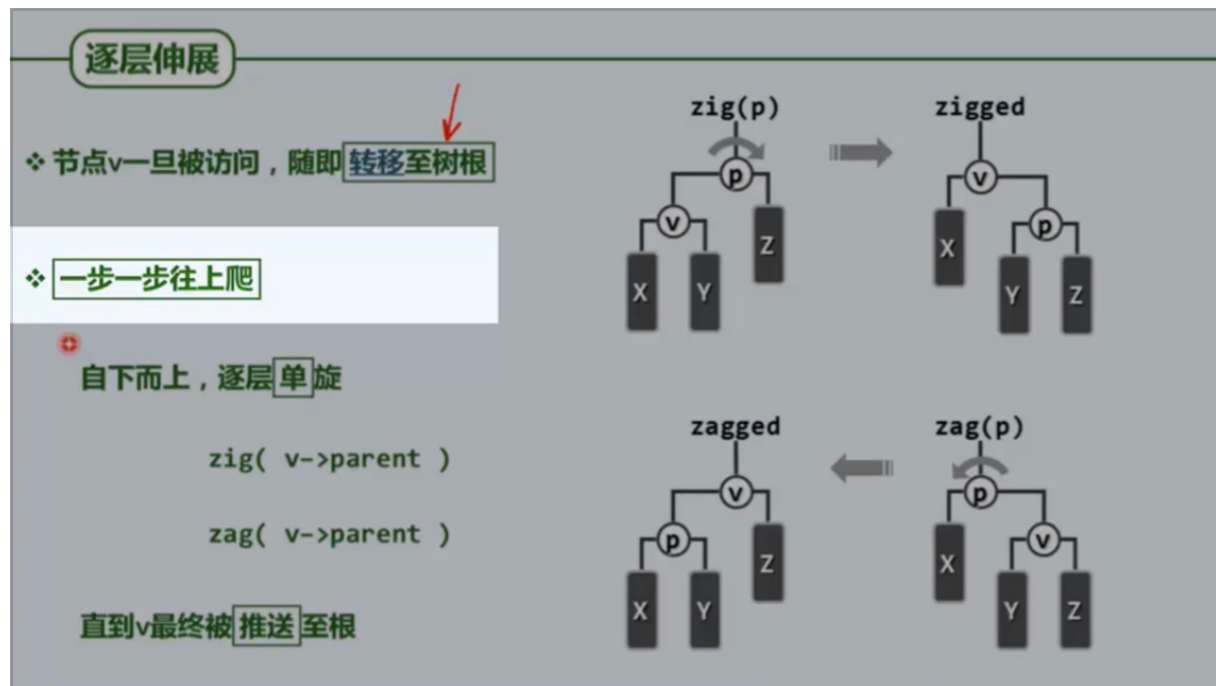
我们是否可以在BST中降低经常被访问元素的深度呢?

### 逐层伸展

节点 $v$ 一旦被访问, 随即转移到树根(每次上升一层)

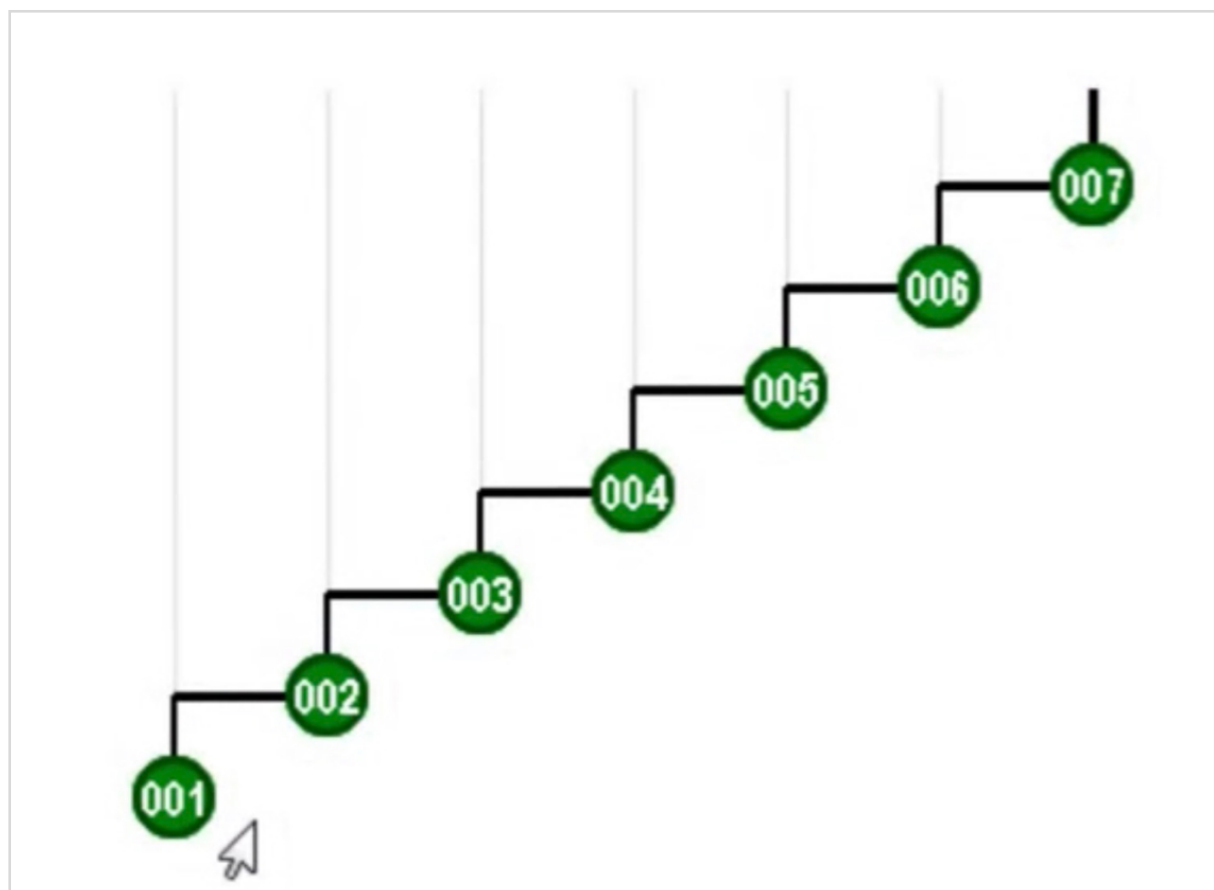
还是要借助到zig和zag转换

一步一步往上爬

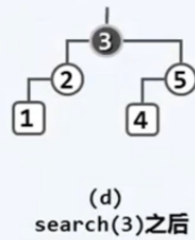
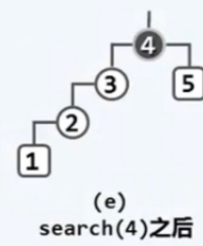
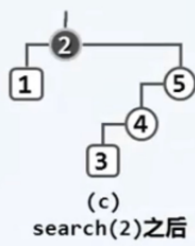
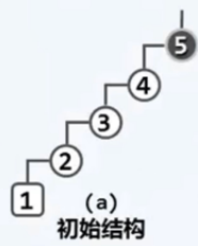


最坏情况：

就是排成一条：  $O(n)$



## 最坏情况



Data Structures (Spring 2014), Tsinghua University

❖ 旋转次数呈周期性的算术级数演变：每一周期累计  $\Omega(n^2)$ ，分摊  $\Omega(n)$ ！

Woooooooooooo

与  $\log n$  相差相差很大，而且与list和Vector毫无区别