Other/Better Criteria?
- Expected case: Some keys more popular than others
- Self-adjusting: Tree adapts as popularity changes

How to design/analyze?
- Splay Tree: A self-adjusting binary search tree
  - No rules! (yay anarchy!)
  - No balance factors
  - No limits on tree height
  - No colors/levels/priorities
- Amortized efficiency:
  - Any single op - slow
  - Long series - efficient on avg.

Intuition: Let T be an unbalanced BST; suppose we access its deepest key

Recap: Lots of search trees
- Unbalanced BSTs
- AVL Trees
- 2-3, Red-black, AA Trees
- Treaps & Skip lists

Focus: Worst-case or randomized expected case

Lesson: Different combinations of rotations can:
- bring given node to root
- significantly change (improve) tree structure.

Splay Trees I

Splay Trees II

Final

Tree's height has reduced by ~ half!

Idea I: Rotate "a" to top (Future accesses to "a" fast)

Idea II: Rotate 2 at a time - upper + lower

Still unbalanced!

→ Tree restructures itself

→ Tree restructures itself
ZigZig(p): [LL case]

Subtrees A, C move up↑

ZigZig(p): [LR case]

Subtrees C, E of p move up↑

Zig(p): [L case]

Node p ← find x by standard BST search while (p ≠ root)
   if (p is child of root) zig(p)
   else /* p has grand parent */
      if (p is LL or RR grand child) zigZig(p)
      else /* p is LR or RL grand child */ zigZag(p)

Splay (Key x):

Example: splay(3)

Subtree A moves up↑

C unchanged

Final ↓

Splay Trees II
**Splay Trees**

- **Analysis**: 
  - Amortized analysis
  - Any one op might take $O(n)$
  - Over a long sequence, average time is $O(n \log n)$ each
  - Amortized analysis is based on a sophisticated potential argument
  - Potential: A function of the tree's structure
    - Balanced $\Rightarrow$ Low potential
    - Unbalanced $\Rightarrow$ High potential
  - Every operation tends to reduce the potential

- **Delete ($x$)**: 
  - $\text{splay}(x)$ [x now at root]
  - $p = \text{root}$
  - if ($p$.key $\neq x$) error!
  - $\text{splay}(x)$ in $p$'s right subtree
  - $q = p$.right [q's key is $x$'s successor]
  - $q$.left = $p$.left
  - root = $q$

- **Splay Trees are Amazingly Adaptive!**

- **Dynamic Finger Theorem**
  - Keys: $x_1, \ldots, x_n$. We perform accesses $x_{i_1}, x_{i_2}, \ldots, x_{i_m}$
  - Let $\Delta_j = i_j - i_{j-1}$, distance between consecutive items
  - Thm: Total access time is $O(m + n \log n + \sum_{j=1}^{m} (1 + \log \Delta_j))$

- **Static Optimality**
  - Suppose key $x_i$ is accessed with prob $p_i$. ($\sum p_i = 1$)
  - Information Theory:
    - Best possible binary search tree answers queries in expected time $O(H)$ where $H = \sum p_i \log \frac{1}{p_i} = \text{Entropy}$
  - Given a seq. of $m$ ops on splay tree with keys $x_1, \ldots, x_n$, where $x_i$ is accessed $g_i$ times. Let $p_i = g_i/m$. Then total time is $O(m \sum p_i \log \frac{1}{p_i})$