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

Splay Tree: A self-adjusting binary search tree
- No rules! (yay anarchy!)
  - No balance factors
  - No limits on tree height
- Amortized efficiency:
  - Any single op - slow
  - Long series - efficient on avg.

Intuition: Let $T$ be an unbalanced BST, suppose we access its deepest key $a$

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

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

Intuition: Let $T$ be an unbalanced BST, suppose we access its deepest key $a$

...final result: $a$
ZigZig(p):  [LL case]  

Subtrees A, C move up ↑

ZigZag(p):  [LR case]  

Subtrees C, E of p move up ↑

Zig(p):  [L case]  

Subtree A moves up ↑ C unchanged

Splay Trees II

Example:

splay(3)  10  RL ZigZag

Final  3

Node p ← splay(x)  
if (p.key == x) Error!!
q ← new Node(x)
if (p.key < x)
p.left ← p
p.right ← p.right
p.right ← null
else ... (symmetrical)...
root ← q

find(x):
root ← splay(x)
if (root == null) return null
if (root.key == x) return root.value
else return null

insert(x):
add node q
splay(x):
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 $p_i$. ($\sum_i p_i = 1$)
- Information Theory: Best possible binary search tree answers queries in expected time $O(H)$ where $H = \sum p_i \log \frac{1}{p_i}$ = Entropy

Static Optimality Theorem:
Given a seq. of $m$ ops. on splay tree with keys $x_1, \ldots, x_n$, where $x_i$ is accessed $q_i$ times. Let $p_i = \frac{q_i}{m}$. Then total time is $O(m \sum p_i \log \frac{1}{p_i})$ = Entropy.

Splay Trees are Amazingly Adaptive!

Balance Theorem: Starting with an empty dictionary, any sequence of $m$ accesses takes total time $O(m \log n + n \log n)$ where $n = \max$ entries at any time.

Analysis:
- Amortized analysis
- Any one op might take $O(n)$
- Over a long sequence, average time is $O(\log n)$ 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

Splay Trees III:
- lu - root
- del(x):
splay(x) [x now at root]
p = root
if (p.key $\neq$ x) error!
splay(x) in p's right subtree
q = p.right [q's key is x's successor] q.left = p.left [q.left = null] root = q

Delete(x):
splay(x) [x now at root]
p = root
if (p.key $\neq$ x) error!
splay(x) in p's right subtree
q = p.right [q's key is x's successor] q.left = p.left [q.left = null] root = q
Announcements - 10/13

- HW 2 - Due Tue, 10/18 start of class

- Midterm 1 - Thu, in class
  - Bring your IDs

No guarantees

Diagram:

```
A → C → E
  ↓     
  b     d
  |     |   LR rotate at f
  v     v
A → C → E
  ↓     
 J     G
```

Legend:

- 1: work example
- 2: short answer
- 3: pseudo code
- 4: proof (induction)
- 5: ???

No guarantees