Module 4: Dictionaries - Enriched

T. Biedl  M. Petrick  O. Veksler
Based on lecture notes by many previous cs240 instructors

David R. Cheriton School of Computer Science, University of Waterloo

Winter 2019

References: Sedgewick 12.6, 13.2, CLRS Problem 13-4, Morin 8, BG85, Knuth71
Outline

1. Dictionaries and Balanced Search Trees
   - Insertion in Treaps
   - More AVL insertions
   - Scapegoat Trees
   - Search in Self-Adjusting Trees
   - Splay Trees
Outline

1. Dictionaries and Balanced Search Trees
   - Insertion in Treaps
   - More AVL insertions
   - Scapegoat Trees
   - Search in Self-Adjusting Trees
   - Splay Trees
Treap Insertion Example

(The lower numbers indicate the “priority” of the node.)

Example:

```
22/4

10/3

4/1
6/0

14/2
13/0
18/1

31/2
28/0

37/1
46/0
```
Outline

1. Dictionaries and Balanced Search Trees
   - Insertion in Treaps
   - More AVL insertions
   - Scapegoat Trees
   - Search in Self-Adjusting Trees
   - Splay Trees
AVL Insertion: Second example

Example:
Outline

1. Dictionaries and Balanced Search Trees
   - Insertion in Treaps
   - More AVL insertions
   - Scapegoat Trees
   - Search in Self-Adjusting Trees
   - Splay Trees
Scapegoat insertion

- First, insert KVP into $T$ with the usual BST insertion.
- We assume that this returns the path $P$ to the new leaf.
- Put a “token” on every node on $P$.
  (These are only needed for analysis, not for implementation.)
- If $|P| > \log_{1/\alpha}(n)$
  ▶ Find highest node $v$ on $P$ such that $\text{size}(v) > \alpha \cdot \text{size}(\text{parent}(v))$
  ▶ $p \leftarrow \text{parent}(v)$ (completely rebuilt sub-tree at $p$)
  ▶ Extract descendants $D$ of $p$ in in-order (sorted).
  ▶ Remove all tokens from $D$.
  ▶ Re-organize $D$ into a perfectly balanced tree:
    $|\text{size}(z.\text{left}) - \text{size}(z.\text{right})| \leq 1$ for all nodes $z$.
  ▶ This takes $O(|D|) = O(\text{size}(p))$ time.
  ▶ Can argue: This releases $\geq (2\alpha + 1)\text{size}(p)$ tokens.
Scapegoat Tree Insertion Example

The lower numbers indicates the subtree-size.
The stars indicate tokens.

Example:
Outline

1. Dictionaries and Balanced Search Trees
   - Insertion in Treaps
   - More AVL insertions
   - Scapegoat Trees
   - Search in Self-Adjusting Trees
   - Splay Trees
MTF-heuristic for binary search trees

Example: $BST$-search(60)

This can get quite unbalanced!
Double Left Rotation = Zig-zag Rotation

First, a right rotation at $p$. Second, a left rotation at $g$. 
Zig-zig Rotation

First, a left rotation at \( g \). Second, a left rotation at \( p \).
Compare to doing two single rotations

Seemingly minor change, but allows for amortized analysis.
Outline

1. Dictionaries and Balanced Search Trees
   - Insertion in Treaps
   - More AVL insertions
   - Scapegoat Trees
   - Search in Self-Adjusting Trees
   - Splay Trees
Splay Tree Insertion

\[ \text{SplayTree-insert}(r, k, v) \]

1. \( x \leftarrow \text{BST-insert}(r, k, v) \)
2. \( \text{while } (x \text{ is not the root}) \)
3. \( p \leftarrow \text{parent of } x \)
4. \( \text{if } (x \text{ is the left child of } p) \)
5. \( \text{if } (p \text{ is the root}) \)
6. \( \text{rotate-right}(p) \)
7. \( \text{else let } g \text{ be the parent of } p \)
8. \( \text{case} \)
9. \( \text{Zig-zig rotation} \) \[ \text{rotate-right}(g) \]
   \[ \text{rotate-right}(p) \]
10. \( \text{Zig-zag rotation} \) \[ \text{rotate-right}(p) \]
    \[ \text{rotate-left}(g) \]
11. \( \text{else ... } \) // symmetric case, \( x \) is right child
Splay Tree rotations

**Example:** $\text{SplayTree-search}(60)$