6: AVL Trees

CSE326 Spring 2002

April 11, 2002

--- Balanced Trees ---

![Balanced Trees Diagram]

- Same data
- Different tree structure

--- Trees Containing 1, 2, 3, 4 ---
General Restructuring Primitives

Rotate Left

Rotate Right

Double Rotation

Strategy

Recipe for a Balanced Tree

1. Insert/Delete as for BST
2. Check if tree balanced
3. If not, use rotations to rebalance
What is Balanced?

A balanced tree has nearly the same height on both sides.

\[
\begin{align*}
\text{LeftHt}(v) &= \begin{cases} 0 \text{ if } \text{Height(LChild}(v)) > 0 \\
1 + \text{Height(LChild}(v)) \text{ otherwise}
\end{cases} \\
\text{RightHt}(v) &= \begin{cases} 0 \text{ if } \text{Height(RChild}(v)) > 0 \\
1 + \text{Height(RChild}(v)) \text{ otherwise}
\end{cases}
\end{align*}
\]

and AVL We Go!

- Balance(v) = RightHt(v) - LeftHt(v)
- T is an AVL tree iff every node \( v \in T \) satisfies
  \[-1 \leq \text{Balance}(v) \leq 1\]

Are AVL Trees Really Balanced?

How bad can an AVL tree get?

- What's the worst \( h \) for a given \( n \)?
- What's the worst \( n \) for a given \( h \)?
AVL Analysis

\[ W_h = \{ \text{The smallest AVL trees of height } h \} \]
\[ = \{ \text{All AVL trees with } w_h \text{ nodes of height } h \} \]

\[ W_0 = \{ \circ \}, \quad w_0 = 1 \]

\[ W_1 = \left\{ \begin{array}{c}
\text{trees with one node}\end{array} \right\}, \quad w_1 = 2 \]

\[ W_2 = \left\{ \begin{array}{c}
\text{trees with two nodes} \end{array} \right\}, \quad w_2 = 4 \]

\[ W_3 = \left\{ \begin{array}{c}
\text{trees with three nodes} \end{array} \right\}, \quad w_3 = 7 \]

Worst of the Worst

Any \( T \in W_h \) must have some \( W_{h-1} \) as a child tree

- If both height \(< h - 1 \) or \( > h - 1 \), then \( T \) not height \( h \)
- If child height \( h - 1 \) but not in \( W_{h-1} \), could get smaller tree by replacing it with \( U \in W_{h-1} \)

Still Bad

The other child must be in \( W_{h-2} \)

- Height either \( h - 1 \) or \( h - 2 \), or \( T \) not height \( h \), or not AVL
- If height \( h - 1 \) could make smaller by using \( h - 2 \) instead
- If not \( \in W_{h-2} \), could make smaller by replacing it with \( U \in W_{h-2} \)
How Bad is Bad?

Trees in $W_2, W_3$ and $W_4$

Bad to the Bone

$w_h = 1 + w_{h-2} + w_{h-1}$

$= F_{h+3} - 1 = 2^O(h)$,

where $F_k$ is the $k$-th Fibonacci number

Hence $h = O(\log n)$ for any AVL tree

Background

- AVL \equiv Adelson-Velskii and Landis
- Russian Scientists
- Invent a data structure, your name will do down in history too!
How Do We Insert?

- Insert on empty tree: \( \emptyset \Rightarrow \emptyset \)

- We’re off to a good start...

Easy Inductive Steps

Oops
The Two Oops

Outside Case
(we just did this one)

Inside Case

Inside Oops

Case A

Case B
What Happens to the Heights?

Inside Rotate Cases

Outside Rotate

Easy Cases

Inside Rotate Cases
Performing an Insertion

1. Descend tree like BST insert
2. Insert Node
3. Retrace steps up tree, at each node
   (a) Update balance
   (b) Rotate if necessary

Analysis of Insertion

Consider first rotating node $v$
- Everything balanced below $v$
- Rotating at $v$ makes $\text{Height}(v)$ as it was before the insertion
- Hence nothing above $v$ will see height change
- So we only need to rotate once!

Waving My Hands

Deletion
1. Delete node as for BST, removing one node (either deleted, or inorder successor/predecessor)
2. If balance of removed’s parent goes from $\pm 1 \rightarrow 0$, $\text{Height}(p)$ changes and grandparent may become unbalanced
   - Rotate to rebalance
3. Grandparent’s parent may now be unbalanced...
4. Continue back path to last balance = 0 node

Running Time?
AVL Trees

- Find: $O(\log n)$
- Insert: $O(\log n)$
- Delete: $O(\log n)$