Bottom-up Rebalancing Binary Search Trees by Flipping a Coin

Gerth Stølting Brodal

Aarhus University
Review

The paper considers an important problem...

The paper fails to solve the problem...
Unbalanced binary search trees – Insertions

1) Locate **insertion point** (empty leaf)
2) Create new leaf node
Unbalanced binary search trees

Inserting 1 2 3 4 5 6

**increasing** sequence gives linear depth

Inserting 3 5 2 4 1 6

**random permutation** gives expected logarithmic depth [Hilbard 1962]
Balanced binary search trees

- Structural **invariants** implying logarithmic depth
- Rebalance using **rotations**

**60s**
AVL-tree

\[ |h(v_{.left}) - h(v_{.right})| \leq 1 \]

**70s**
Red-black tree

No two adjacent red & root-to-leaf paths same #black

**80s**
Splay tree

Always rotate accessed node to the root

**90s**
Treap

Elements random priority, stored using heap order
## Balanced binary search tree insertions: rebalancing cost

<table>
<thead>
<tr>
<th>Reference</th>
<th>Name</th>
<th>Time</th>
<th>Rotations</th>
<th>Random bits</th>
<th>Space pr node</th>
</tr>
</thead>
<tbody>
<tr>
<td>[AVL62]</td>
<td>AVL-trees</td>
<td>$O_A(1)$</td>
<td>O(1)</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>[GS78]</td>
<td>Red-black trees</td>
<td>$O_A(1)$</td>
<td>O(1)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>[B79]</td>
<td>Encoded 2-3 trees</td>
<td>$O(lg\ n)$</td>
<td>$O(lg\ n)$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[ST85]</td>
<td>Splay trees</td>
<td>$O_A(lg\ n)$</td>
<td>$O_A(lg\ n)$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[A89,GR93]</td>
<td>Scapegoat</td>
<td>$O(lg\ n)$</td>
<td>$O(lg\ n)$</td>
<td>0</td>
<td>global $O(lg\ n)$</td>
</tr>
<tr>
<td>[SA96]</td>
<td>Treaps</td>
<td>$O_E(1)$</td>
<td>$O_E(1)$</td>
<td>$O_E(1)$</td>
<td>$O_E(1)$</td>
</tr>
<tr>
<td>[MR98]</td>
<td>Randomized BST</td>
<td>$O_E(lg\ n)$</td>
<td>$O_E(1)$</td>
<td>$O_E(lg^2\ n)$</td>
<td>$O(lg\ n)$</td>
</tr>
<tr>
<td>[S09]</td>
<td>Seidel</td>
<td>$O_E(lg^2\ n)$</td>
<td>$O_E(1)$</td>
<td>$O_E(lg^3\ n)$</td>
<td>0</td>
</tr>
<tr>
<td><strong>Open problem</strong></td>
<td></td>
<td>$O_E(1)$</td>
<td>O(1)</td>
<td>$O_E(1)$</td>
<td>0</td>
</tr>
</tbody>
</table>
Question asked in this paper

Does there exist a randomized rebalancing scheme satisfying...

1. No balance information stored
2. Worst-case $O(1)$ rotations
3. Most rotations near the insertion
4. Local information only
5. Expected $O(1)$ time
6. $O(1)$ random bits per insertion
7. Nodes expected depth $O(\lg n)$
Algorithm RebalanceZig

\[
\text{RebalanceZig}(v) \\
\text{while } v.p \neq \text{NIL and coin flip is tail do} \\
\quad v \leftarrow v.p \\
\quad \text{if } v.p \neq \text{NIL then} \\
\quad \text{RotateUp}(v)
\]

Fact 1  \text{RebalanceZig} takes expected O(1) time and performs \leq 1 rotation

Theorem 1  \text{RebalanceZig} on increasing sequence each node expected depth O(lg n), for 0 < p < 1 (p = Pr[tail])
Algorithm RebalanceZig extreme probabilities

\[ \text{RebalanceZig}(v) \]

while \( v.p \neq \text{NIL} \) and coin flip is tail do
  \( v \leftarrow v.p \)
  if \( v.p \neq \text{NIL} \) then
    \text{RotateUp}(v) 
\]

- \( \Pr[\text{tail}] = 1 \) never performs rotation
  i.e. unbalanced binary search tree

- \( \Pr[\text{tail}] = 0 \) always rotates up new leaf
  i.e. tree is always a path
RebalanceZig \((n = 1024)\)

- **Rotate onto path**
- **No rebalancing**

**Theorem 1**

Increasing is \(\Theta(\lg n)\) for \(0 < p < 1\)
Different insertion sequences

3 5 2 4 1 6  
(permutation)

1 2 3 4 5 6  
(increasing)

1 6 2 5 3 4  
(converging)

2 4 6 5 3 1  
(bitonic)

2 4 6 1 3 5  
(runs)

2 1 4 3 6 5  
(pairs)
Theorem 2
Converging is $\Theta(n)$

Theorem 3
Pairs is $\Theta(n)$ for $p \neq \frac{1}{2}$

Conjecture
Pairs is $\Theta(\sqrt{n})$ for $p = \frac{1}{2}$
“Proof” of Theorem 2

- For each pair the depth of the insertion point increases by $\geq 1$ with probability $\geq p(1-p)$:

If inserting $u$ rotates a (strict) ancestor of $u$ up, and inserting $v$ rotates $v$ up, then depth of insertion point * increases by inserting the pair.
Why ReblanceZig is bad on pairs

"Proof" of Theorem 3

- $p < 1/2$, odd numbers tend to be rotated on rightmost path, right path expected $\Theta(n)$ nodes
- $p > 1/2$, rightmost path tends to contain $O(1)$ nodes, left path expected $\Theta(n)$ nodes
RebalanceZig \( (n = 1024) \)

Find a randomized algorithm that handles **converging** sequences.
Algorithm RebalanceZigZag

\[
\text{RebalanceZigZag}(v) \\
\text{while } v.p \neq \text{NIL} \text{ and coin flip is tail do} \\
\quad v \leftarrow v.p \\
\text{if } v.p \neq \text{NIL} \text{ and } v.p.p \neq \text{NIL} \text{ then} \\
\quad \text{if } (v = v.p.l \text{ and } v.p = v.p.p.l) \text{ or } (v = v.p.r \text{ and } v.p = v.p.p.r) \text{ then} \\
\quad \quad \text{RotateUp}(v.p) \quad \triangleright \text{ zig-zig or zag-zag case} \\
\text{else} \\
\quad \text{RotateUp}(v) \quad \triangleright \text{ zig-zag or zag-zig case} \\
\quad \text{RotateUp}(v) \\
\]
Theorem 4
Increasing is $\Theta(\lg n)$ for $0 < p < 1$

Theorem 5
Converging is $\Theta(\lg n)$ for $0.62 < p < 1$
Theorem 6
Pairs is $\Theta(n)$ for $0 \leq p \leq 1$
Summary

RebalanceZig ($n = 1024$)

RebalanceZigZag ($n = 1024$)
Bottom-up Rebalancing Binary Search Trees by Flipping a Coin
Bottom-up Rebalancing
Balanced Binary Search Trees
by
Flipping a Coin