Unbalanced Tree

Insert 1, 2, 3, 4, 5, 6, 7 into a BST

Insert 4, 2, 6, 1, 3, 5, 7 into a BST

From the second BST, remove 2, insert 9, insert 8, insert 11, remove 8, remove 3

Moral: even a balanced tree can become unbalanced after a number of insertions and removals

(Why is a balanced tree more desirable?)

When is a tree said to be balanced?
Balanced Tree

A perfect binary tree of height $h$ has exactly $2^{h+1} - 1$ internal nodes

Only trees with $n = 1, 3, 7, 15, 31, 63, \ldots$ internal nodes can be balanced

Need another definition of “balance condition”
Want the definition to satisfy the following criteria

1. height of tree of $n$ nodes = $O(\log n)$
2. balance condition can be maintained efficiently: $O(1)$ to rebalance a tree

We will look at AVL-tree and B-tree
AVL Tree

Adel’son-Vel’skiï-Landis (AVL) tree’s balance condition:
A non-empty binary tree is AVL balanced if both \( T_l \) and \( T_r \) are AVL balanced and

\[
|h_l - h_r| \leq 1, \tag{1}
\]

where \( h_l \) is height of \( T_l \) and \( h_r \) height of \( T_r \)

Preiss Th. 10.2, p. 316: The height of an AVL tree with \( n \) internal nodes is \( \Theta(\log n) \)

The AVL balance condition satisfies criterion 1 of balance condition
Can the balance condition be maintained in \( O(1) \)?
AVL Tree Insertion, Removal, and Balance Factor

Insert and remove operations exactly the same as for BST, but must “re-balance” the tree if AVL balance condition is violated after insertion/removal

Define “balance factor” \((B_i)\) of a node \(i\) as \((h_{l_i} - h_{r_i})\):

- if the tree rooted at node \(i\) is AVL balanced, \(|B_i| \leq 1\)
- if \(T_{r_i}\) is deeper, \(B_i < -1\)
- if \(T_{l_i}\) is deeper, \(B_i > 1\)
If $|B_i| > 1$, there are four cases to consider depending on the direction of the imbalance from the unbalanced node: LL, LR, RL, and RR.

For example:

**LL:** a new node is added to $u_l$: $B_u = 0 \rightarrow 1 \Rightarrow B_r = 2$, which violates the AVL balance condition, and the tree rooted at $r$ is now unbalanced.

Sugih Jamin (jamin@eecs.umich.edu)
Rebalancing AVL Tree

Want: rebalancing operation to be $O(1)$ time complexity

**Preiss Th. 10.3, p. 322**: When an AVL tree becomes unbalanced, *exactly* one single or double rotation is required to balance the tree.

AVL rotations: from the node $r$ that has become unbalanced ($|B_r| > 1$), do LL, RR, LR, or RL rotation depending on the direction of the imbalance from $r$.

Rotate counter to the direction of traversal.

For double rotations, reverse the effect of the last traversal first.

Must retain BST property at all times.
Single LL Rotation

- L: rotate right: make $r$ the right child of $u$
- L: rotate right: make $u_r$ the left child of $r$

Note Errata for Preiss Fig. 10.8
Single RR Rotation

- R: rotate left: make $r$ the left child of $v$
- R: rotate left: make $v_l$ the right child of $r$
Double LR Rotation

- **R**: rotate left: do an RR rotation on \( u \)
- **L**: rotate right: do an LL rotation on \( r \)

\[
\begin{align*}
&Br = 2 \\
&Bu = -1 \\
&Bw = -1 \\
\end{align*}
\]

\[
\begin{align*}
&Br = 2 \\
&Bu = -1 \\
&Bw = -1 \\
\end{align*}
\]

\[
\begin{align*}
&Br = 2 \\
&Bu = 1 \\
&Bw = 1 \\
\end{align*}
\]

\[
\begin{align*}
&Br = 0 \\
&Bu = 1 \\
&Bw = 1 \\
\end{align*}
\]

Sugih Jamin (jamin@eecs.umich.edu)
Double RL Rotation

- L: rotate right: do an LL rotation on $v$
- R: rotate left: do an RR rotation on $r$

After LL Rotation

After RR Rotation