Extra AVL Tree Slides

General Single Rotation
- Height of subtree same as it was before insert!
- Height of all ancestors unchanged.

Insert into Z, increasing height

General Double Rotation
- Height of subtree still the same as it was before insert!
- Height of all ancestors unchanged.

Height of an AVL tree

**Theorem:** Any AVL tree with \( n \) nodes has height less than 1.441 \( \log n \).

**Proof:** Given an \( n \)-node AVL tree, we want to find an upper bound on the height of the tree.

Fix \( h \). What is the smallest \( n \) such that there is an AVL tree of height \( h \) with \( n \) nodes?

Let \( W_h \) be the set of all AVL trees of height \( h \) that have as few nodes as possible.

Let \( S(h) \) be the number of nodes in any one of these trees.

\[ S(0) = 1, \quad S(1) = 2 \]

Suppose \( T \in W_h \), where \( h \geq 2 \). Let \( T_L \) and \( T_R \) be \( T \)'s left and right subtrees. Since \( T \) has height \( h \), either \( T_L \) or \( T_R \) has height \( h-1 \). Suppose it’s \( T_R \).

By definition, both \( T_L \) and \( T_R \) are AVL trees. In fact, \( T_R \in W_{h-1} \), or else it could be replaced by a smaller AVL tree of height \( h-1 \) to give an AVL tree of height \( h \) that is smaller than \( T \).

Similarly, \( T_L \in W_{h-2} \).

Therefore, \( S(h) = 1 + S(h-2) + S(h-1) \).

**Claim:** For \( h \geq 0 \), \( S(h) \geq \varphi^h \), where \( \varphi = (1 + \sqrt{5}) / 2 \approx 1.6 \).

**Proof:** The proof is by induction on \( h \).

**Basis step:** \( h = 0 \). \( S(0) = 1 = \varphi^0 \).

**Induction step:** Suppose the claim is true for \( 0 \leq m \leq h \), where \( h \geq 1 \).
Then:
\[ S(h+1) = 1 + S(h-1) + S(h) \]
\[ \geq 1 + \varphi^{h-1} + \varphi^h \quad \text{(by the i.h.)} \]
\[ = 1 + \varphi^{h-1} (1 + \varphi) \quad \text{(by math)} \]
\[ = 1 + \varphi^{h+1} \quad \text{(using } 1+\varphi = \varphi^2 \text{)} \]
\[ > \varphi^{h+1} \quad \text{Thus, the claim is true.} \]

From the claim, in an \( n \)-node AVL tree of height \( h \),
\[ n \geq S(h) \geq \varphi^{h} \quad \text{(from the Claim)} \]
\[ h \leq \log_{\varphi} n \quad \text{(by math – log of both sides)} \]
\[ = (\log n) / (\log \varphi) \]
\[ < 1.441 \log n \]

**AVL tree: Running times**

- **find** takes \( O(\log n) \) time, because height of the tree is always \( O(\log n) \).
- **insert**: \( O(\log n) \) time because we do a find \( (O(\log n) \text{ time}) \), and then we may have to visit every node on the path back to the root, performing up to 2 single rotations \( (O(1) \text{ time each}) \) to fix the tree.
- **remove**: \( O(\log n) \) time. Left as an exercise.

**AVL Insert Algorithm**

- **Recursive**
  1. Search downward for spot
  2. Insert node
  3. Unwind stack, correcting heights
    a. If imbalance \#1, single rotate
    b. If imbalance \#2, double rotate

- **Iterative**
  1. Search downward for spot, stacking parent nodes
  2. Insert node
  3. Unwind stack, correcting heights
    a. If imbalance \#1, single rotate and exit
    b. If imbalance \#2, double rotate and exit

**Why use a stack?**

**Single Rotation Code**

```cpp
void RotateRight(Node root) {
    Node temp = root.right
    root.right = temp.left
    temp.left = root
    root.height = max(root.right.height(), root.left.height()) + 1
    temp.height = max(temp.right.height(), temp.left.height()) + 1
    root = temp
}
```

**Double Rotation Code**

```cpp
void DoubleRotateRight(Node root) {
    RotateLeft(root.right)
    RotateRight(root)
}
```

**Double Rotation Completed**

- **First Rotation**
- **Second Rotation**