CS 225
Data Structures

February 17 – BST Balance
G Carl Evans
Height-Balanced Tree

What tree makes you happier?

Height balance:  \( b = \text{height}(T_R) - \text{height}(T_L) \)

A tree is height balanced if:
BST Rotation

We will perform a rotation that maintains two properties:

1.

2.
BST Rotation Summary

- Four kinds of rotations (L, R, LR, RL)
- All rotations are local (subtrees are not impacted)
- All rotations are constant time: O(1)
- BST property maintained

**GOAL:**

We call these trees:
AVL Trees

Three issues for consideration:
- Rotations
- Maintaining Height
- Detecting Imbalance
AVL Tree Rotations

Four templates for rotations:
Finding the Rotation on Insert

**Theorem:**
If an insertion occurred in subtrees $t_3$ or $t_4$ and a subtree was detected at $t$, then a __________ rotation about $t$ restores the balance of the tree.

We gauge this by noting the balance factor of $t$->**right** is ______.
Finding the Rotation on Insert

Theorem:
If an insertion occurred in subtrees \( t_2 \) or \( t_3 \) and a subtree was detected at \( t \), then a __________ rotation about \( t \) restores the balance of the tree.

We gauge this by noting the balance factor of \( t \rightarrow \text{right} \) is _______.

Diagram:
- Node \( t \) with subtrees \( t_1, t_2, t_3, t_4 \)
Insertion into an AVL Tree

```
struct TreeNode {
  T key;
  unsigned height;
  TreeNode *left;
  TreeNode *right;
};
```
Insertion into an AVL Tree

Insert (pseudo code):
1: Insert at proper place
2: Check for imbalance
3: Rotate, if necessary
4: Update height

```c
struct TreeNode {
    T key;
    unsigned height;
    TreeNode *left;
    TreeNode *right;
};
```

_insert(6.5)