AVL Trees

Definition 0.1

The **balance factor** of a node is the left child's height - right child's height

- What do different values of the balance factor mean?
  1. 0: Perfectly Balanced
  2. 1: Leaning left but ok
  3. neg 1: Leaning right but ok
  4. >= 2: unlabaced to far left
  5. <= neg 2 unbalanced to far right

Definition 0.2

In **AVL Trees**, the $|\text{node.balancefactor}| \leq 1$. We chose 1 and not 0 because there are some situations where it is straight up impossible to have a perfectly balanced tree.

AVL's help us hit the sweet spot of allowing only a little imbalance while keeping operations efficient.

Single Rotations

These are used when adding or removing a single node to a tree if it becomes imbalanced.

To perform a rotation when a node has a balance factor of −2: look at the right child first and see if it has a balance factor of 0, −1. Then we can do a simple, single rotations.

Theorem 0.1

```
Algorithm leftRotation
1) Node B <- A's right child
2) A's right child <- B's left child
3) B's left child <- A
4) (ALWAYS FIRST BEFORE 5) Update the height & BF of A
5) Update the height & BF of B
6) Return B
```

Right rotations are the mirrored opposite of left rotations.

When to use which rotation?

Use left rotation when
- node.BF = −2
- node.right.BF = −1, 0

Use right rotation when (? not sure)
- node.BF = 2
- node.right.BF = 1, 0

> A nodes balance factor shouldn't go past 2 without a rotation happening