Balancing Trees

Basic terminology:

Balanced Tree: tree where subtrees heights differ ≤ 1

skew: skew of tree is height(right) $-$ height(left).

Exercise: Binary tree of height balanced nodes have a height $O\log_2(n)$

Proof: Recursion! For a height h tree, we have if $F(h)$ is # min vertices,

$$F(h) = 1 + F(h-1) + F(h-2) \geq 2F(h-2)$$

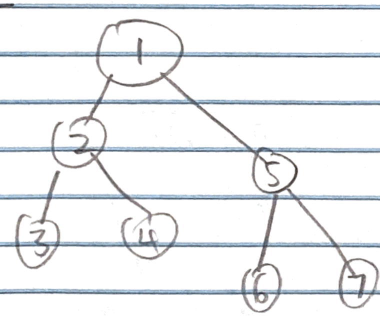
So $F(h) \geq 2^{h/2}$ hence $F(h) = O(\log_2 n)$

Rotations:

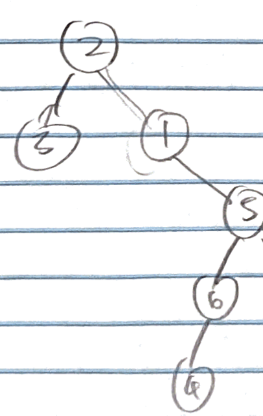
- we can preserve all structures of tree with a rotation, which is simply shifting the root pointer one to the left or right.

Example:

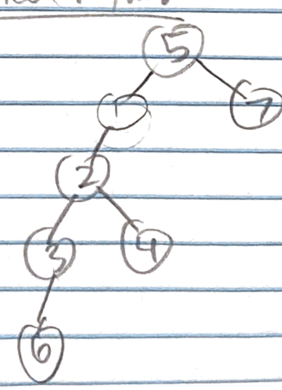
Tree:



Rotated left:



Rotated Right:



Exercise: Develop algorithm to height balancing a tree.

Steps:

1. Rotate left/right depending on skew to make sure $|skew| \leq 1$.
2. Apply method on two children

I think this works because it ensures the balanced condition by making subtrees balanced too.

Runtime Calculation:

For each layer, runtime is $O(n)$ since n different possible rotations. Each time, we cut # needed to rotate by half so

$$O(n \cdot \log_2(n)) = O(n)$$

is the runtime.