Overview of Trees

This lesson is a quick overview of trees, their types, and some important formulas to compute the height and number of nodes in a tree.

We'll cover the following

Binary Trees

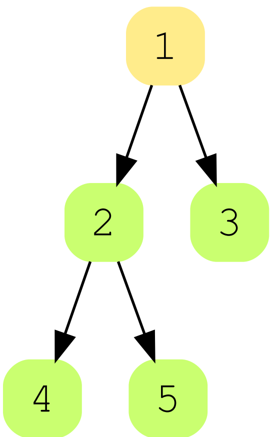
Definition: A tree having a max. of two children at each internal node

Types: *Perfect, Full, Complete, Skewed*

Total number of nodes: $2^{(h+1)} - 1$

Total number of leaf nodes: $2^h or \frac{(n+1)}{2}$

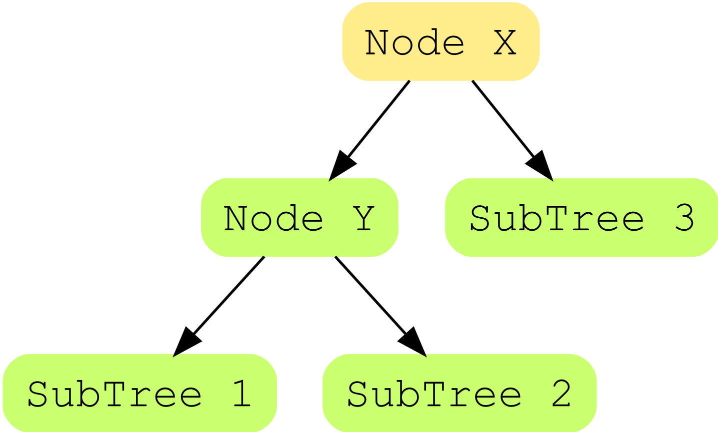
Height: $log_2(n + 1) - 1$



Binary Search Trees

Definition: Every node has a value greater than/equal to all the node values in its left sub-tree, and has a value less than all the node values in its right sub-tree. Mathematically,

$$Keys(SubTree1) < Keys(Y) < Keys(Subtree2) < Keys(X) < Keys(SubTree3)$$

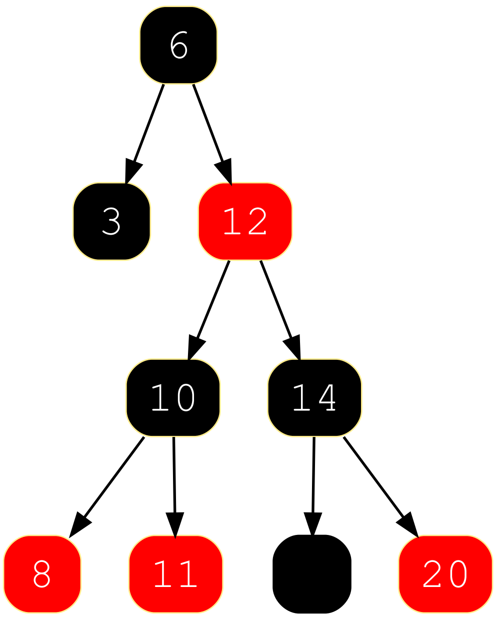


Red Black Trees

Definition: A tree where every node is colored as red or black, no two adjacent nodes have *red* coloring, and root and null nodes are considered *black*.

Height: $h <= 2log_2(n+1)$

Minimum number of nodes: $(h+1)+2\left(\sum_{i=0}^{y})2^{i}-1\right)$, where y is equal to: $floor(h/2)$



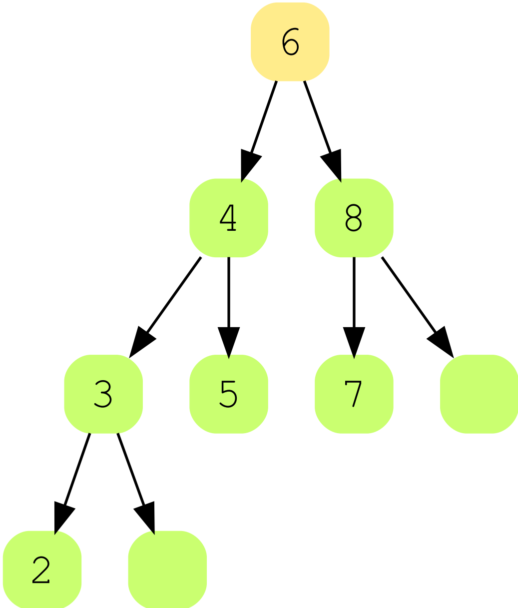
AVL Trees

Definition: For each node, the height of the left and right *sub-trees* differ by a max. of one.

Minimum number of nodes: $N(h) = 1 + N(h-1) + N(h-2)$

Maximum number of nodes: $N - 1 + 2^{log(N-1)+2}$

Height: $O(log_2 n)$



2-3 Trees

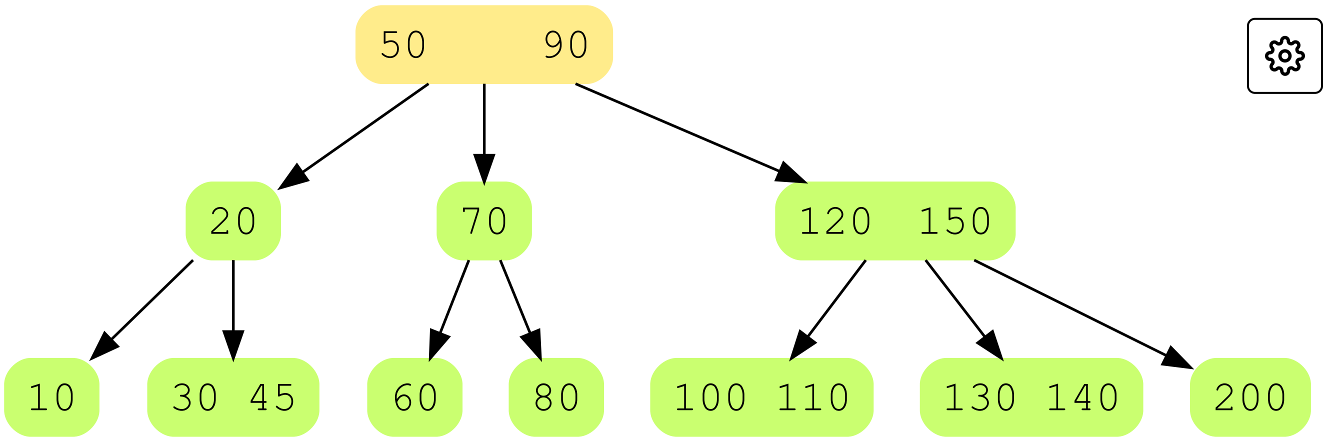
Definition: A balanced and ordered tree where each node can have a max. of two keys (X and Y) and three children, such that,

$LChild.Key < X < MChild.Key < Y < RChild.Key $

Maximum number of nodes: $ 3^h$

Height: $log_4(n+1) - 1 < h < log_2(n+1) - 1$

Types: *2-3-4 Trees*



Now let's apply what we have learned so far on some exciting coding challenges!

← Back

Next →

2-3-4 Trees

Challenge 1: Find the Minimum Value i...

Mark as Completed