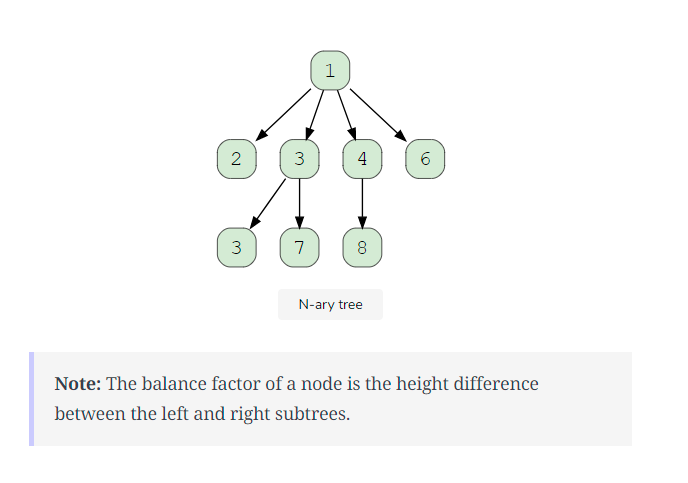
**Types of Trees**

There are many types of trees that we can use to organize data differently within a hierarchical structure. The tree we use depends on the problem we are trying to solve. Let’s take a look at the trees we can use in Java. We will be covering:

* N-ary trees
* Balanced trees
* Binary trees
* Binary Search Trees
* AVL Trees

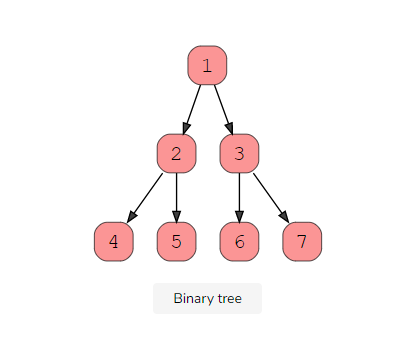
**N-ary Tree**

In N-ary tree, a node can have child nodes from 0-N. For example, if we have a 2-ary tree (also called a Binary Tree), it will have a maximum of 0-2 child nodes.



### Balanced Tree

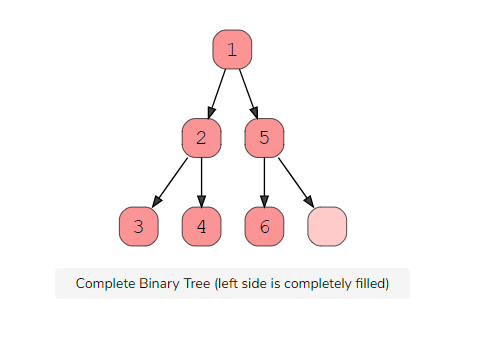
A balanced tree is a tree with almost all leaf nodes at the same level, and it is most commonly applied to sub-trees, meaning that all sub-trees must be balanced. In other words, we must make the tree height balanced, where the difference between the height of the right and left subtrees do not exceed **one**. Here is a visual representation of a balanced tree.



There are three main types of binary trees based on their structures.

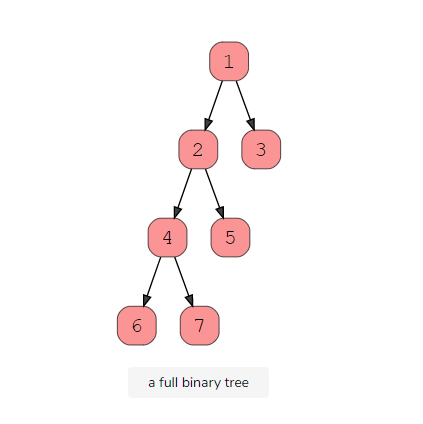
#### ****1. Complete Binary Tree****

A complete binary tree exists when every level, excluding the last, is filled and all nodes at the last level are as far left as they can be. Here is a visual representation of a complete binary tree.



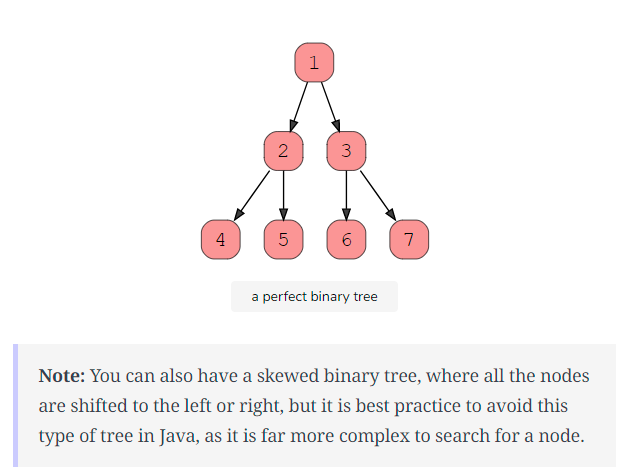
#### ****2. Full Binary Tree****

A full binary tree (sometimes called proper binary tree) exits when every node, excluding the leaves, has two children. Every level must be filled, and the nodes are as far left as possible. Look at this diagram to understand how a full binary tree looks.



#### ****3. Perfect Binary Tree****

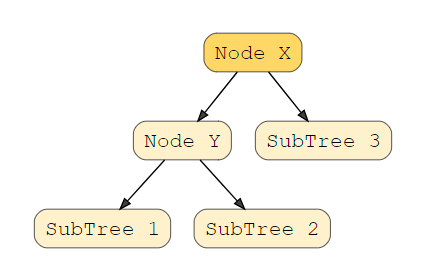
A perfect binary tree should be both full and complete. All interior nodes should have two children, and all leaves must have the same depth. Look at this diagram to understand how a perfect binary tree looks.



### Binary Search Trees

A Binary Search Tree is a binary tree in which every node has a key and an associated value. This allows for quick lookup and edits (additions or removals), hence the name “search”. A Binary Search Tree has strict conditions based on its node value. It’s important to note that every Binary Search Tree is a binary tree, but not every binary tree is a Binary Search Tree.

What makes them different? In a Binary Search Tree, the left subtree of a subtree must contain nodes with fewer keys than a node’s key, while the right subtree will contain nodes with keys greater than that node’s key. Take a look at this visual to understand this condition.



In this example, the node Y is a parent node with two child nodes. All nodes in subtree 1 must have a value less than node Y, and subtree 2 must have a greater value than node Y.

### AVL Trees

AVL trees are a special type of Binary Search tree that are self-balanced by checking the balance factor of every node. The balance factor should either be **+1**, **0**, or **-1**. The maximum height difference between the left and right sub-trees can only be **one**.

If this difference becomes more than one, we must re-balance our tree to make it valid using rotation techniques. These are most common for applications where searching is the most important operation. Look at this visual to see a valid AVL tree.

