Full Binary Tree

* Each node either is a leaf or is an internal node with exactly two non-empty children

Complete Binary Tree

* If the height (number of levels) of the tree is d, the first (top) d-1 levels are completely full. Namely, all the positions are taken/occupied.
* The Bottom level has nodes that are filled from left to right.
* There is NO GAP on the bottom level, all the nodes are adjacent on the bottom level being filled from left to right.

B Trees or B+ Trees

R Tree, R\* Tree,

Height-Balanced Binary Tree (Balanced)

* An empty tree is height-balanaced
* A non-empty binary tree T is balanced if:
  + Left subtree of T is balanced
  + Right subtree of T is balanced
  + The height difference between left subtree and right subtree is no more than 1.

*Logic to check if a Tree is Balanced or not*

* Check all subtrees in the diagram, to see if all of the subtrees satisfy the height difference condition.

Boolean isBalanced ( Node root) {  
 Int lh, rh;

If (root == null) return true;

Lh = height (root.left);

Rh = height (root.left);

if(Math.abs(lh - rh) <= 1 && isBalanced(root.left) && isBalanced(root.right))

Return true;

Return false;

}

Int height (Node node) {

If(node == null)

Return 0;

Int left = height(root.left);

Int right = height(root.right);

if(left > right) return right + 1;

Retrrn left + 1;

}

Int height (node node) {

If (node == null) return 0;

Return 1 + max(height(node.left), height(node.right));