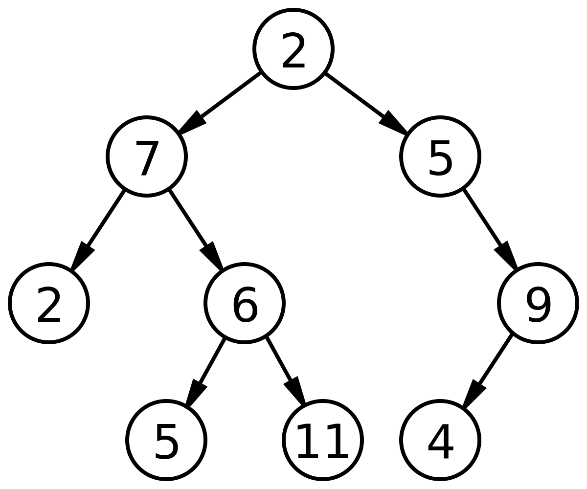
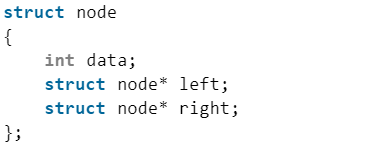
*Binary Trees*

A binary tree is a *hierarchical* and *non-linear* data structure with a maximum of two children for each parent. Every node in a binary tree has a left and right reference along with the data element. Any kind of searching, inserting or deleting of a node has an average *O(h) time complexity*.

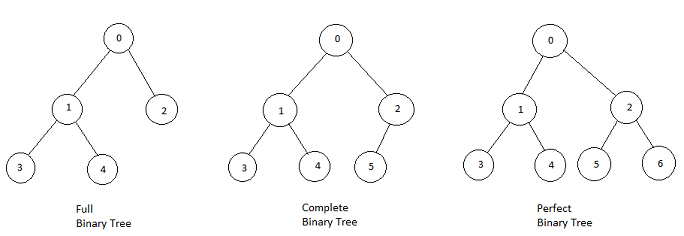




***Full Binary Tree****:* a binary tree where every node has 0 or 2 children. We can also say is a binary tree in which all nodes except leaf nodes have two children.

***Complete Binary Tree:*** a binary tree all the levels are completely filled except possibly the last level and the last level has all keys as left as possible. (between 2h+1 - 1 and 2h nodes)

***Perfect Binary Tree:***a binary tree in which all the internal nodes have two children and all leaf nodes are at the same level. (2h+1 - 1 nodes)



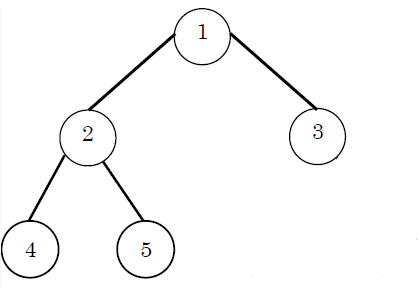
*min heigth = floor(log(n)), max heigth = n (degenerate tree)*

*Tree Traversals*

1. ***Depth First Traversals****:*

1.1. Inorder (SRD) : 4 2 5 1 3   
1.2. Preorder (RSD) : 1 2 4 5 3   
1.3. Postorder (SDR) : 4 5 2 3 1

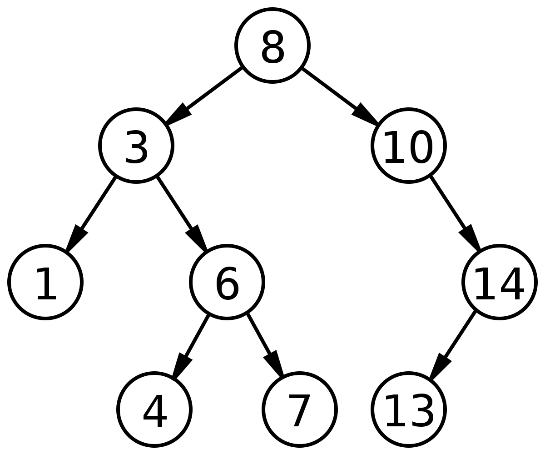
2. ***Breadth First Traversal*** : 1 2 3 4 5



*Binary Search Trees*

Binary Search Tree is a node-based binary tree data structure which has the following properties:

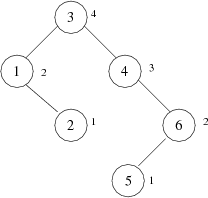
* The left subtree of a node contains only nodes with keys *lesser* than the node’s key.
* The right subtree of a node contains only nodes with keys *greater* than the node’s key.
* There must be *no duplicate* nodes.



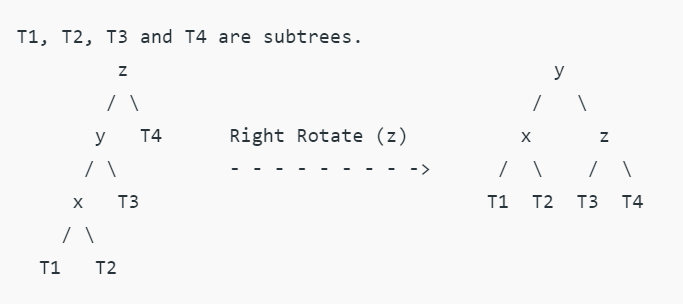
By traversing the tree in *inorder* we produce a *sorted* list of the nodes. The average time complexity of searching an element in a BTS is ***O(log n)*** for a balanced tree, and worst case ***O(n)*** for a degenerate binary tree.

*AVL Trees*

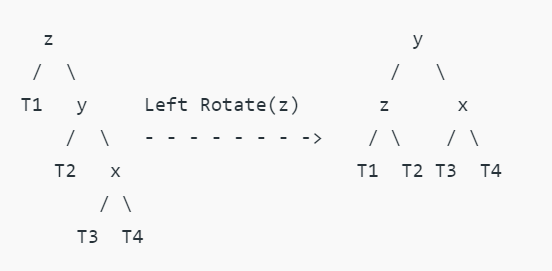
AVL tree is a *self-balancing* binary search tree where the difference between heights of left and right subtrees **cannot be more than one** for all nodes. The heigth of and AVL tree is *log(n)* and therefore every searching, inserting or deleting has an *O(log n) time complexity*. An AVL can perform 4 possible ***balancing operations*** also called ***rotations*** that do not alter the property of the BTS:



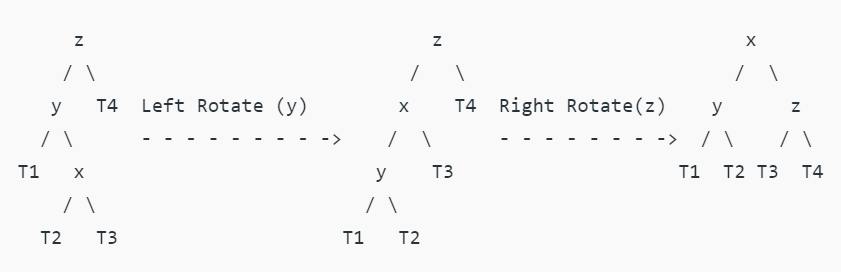
1. Left Left Case



1. Right Right Case



1. Left Right Case



1. Right Left Case

