**Binary Search Tree**

**Balanced Tree**

**AVL**

1. **Tree:**

-A tree <T> (Tree) is :

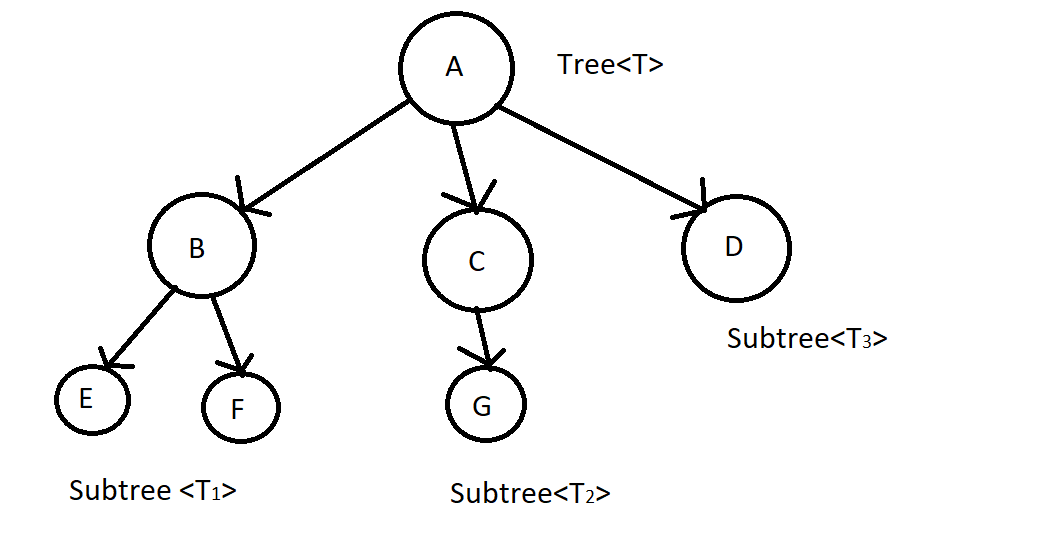
+A set of values ​​of the same type, elements of the same value are stored by a node, called node p1,p2,...

+ If a tree has no elements, it is called an empty tree.

+If the tree is not empty, there are some caveats:

. Each tree has only 1 root node. The root node is the starting node of the tree.

.The remaining nodes are divided into non-intersection sets (Ti ,Ti+1,…). Each <Ti> is called a subtree of <T> tree .Example:

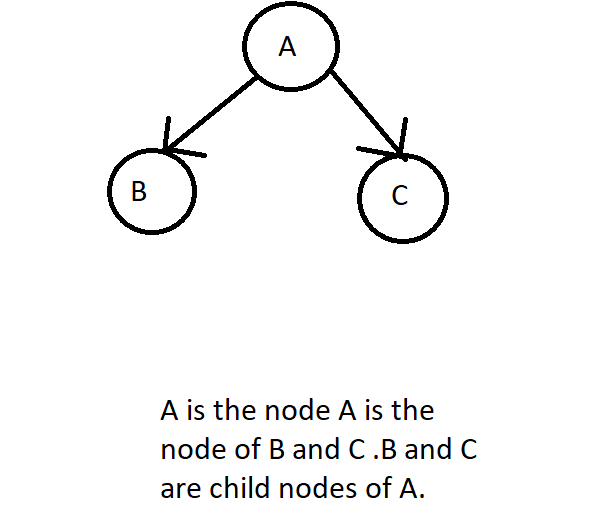


-Properties:

+The root node has no parent node.

+Leaf node (external node):is a node that does not have any child nodes.

+ Parent node: is the node that we will be processed to do something before reaching another node that it can go to. Child node: It is the node that when the parent node has been processed, it comes to it.Example:



+Each node has only one node parent.

+Each node can have multiple children.

+Trees do not count cycles . That is, when a node 1 node goes to a child node, the child node cannot have any path in the tree to go against the node above. If possible it is called a graph.

+Node: is an element in tree. Node can contain any data. Each node can contain many types of data. For example, 1 node can contain 1 data as name, another data as age,... And importantly, all nodes must be the same in terms of data but may differ in the values ​​in the data that the node holds. That's why we say a tree is a data structure that collects elements of the same type.

+Branch: link 2 nodes together.

+ Sibling nodes: is a node with the same parent node.

+Degree of node: number of children of a node.

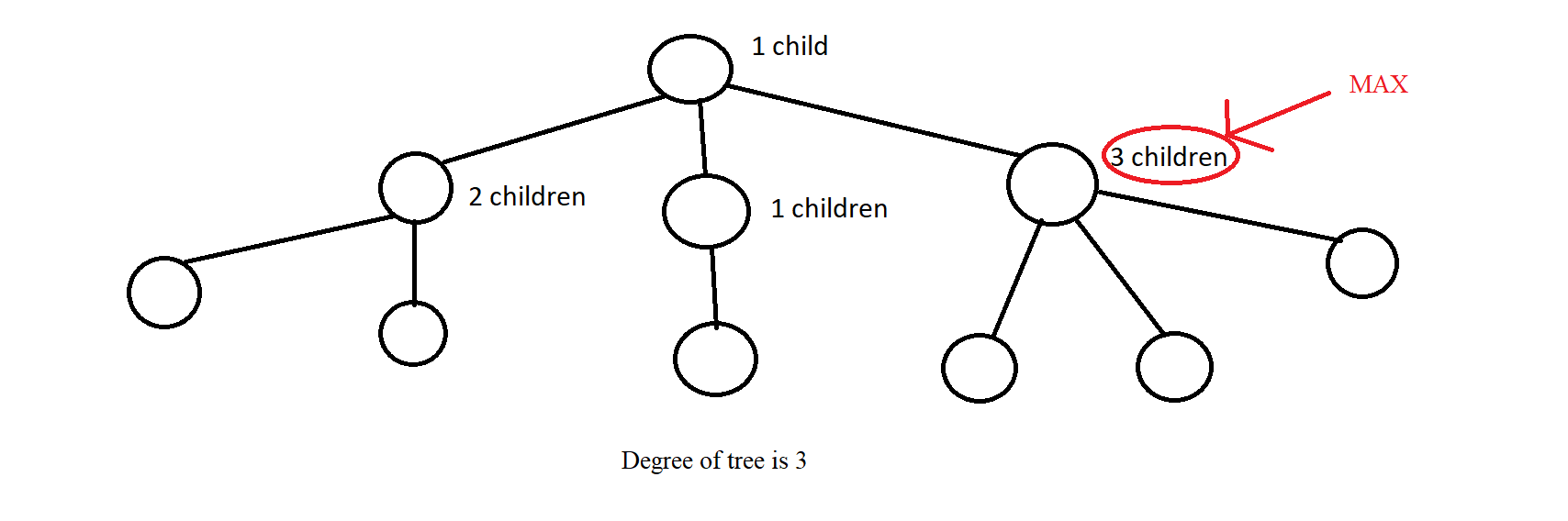
+Internal node: is a node that has both a parent node and a child node.

+Subtree: is a subtree of the original tree. Also starts with a node.

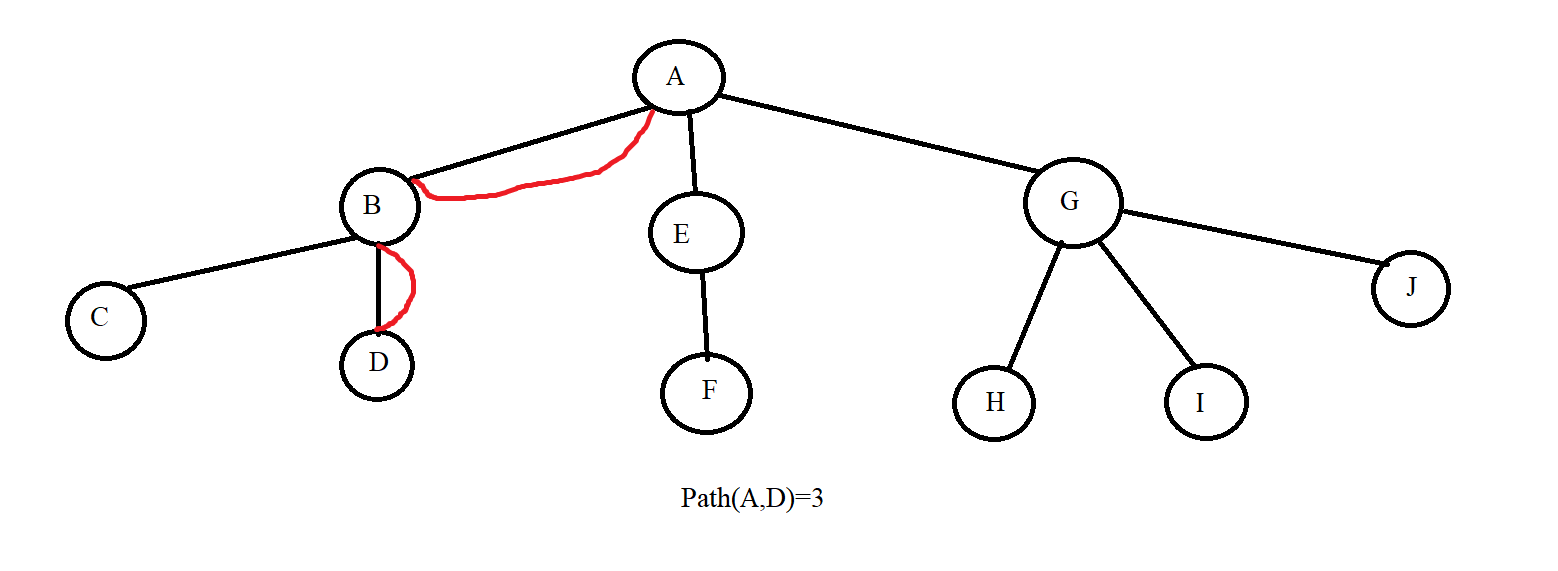
+Degree of tree: is the maximum value of the degree of node in the tree. When we interpret it with a formula, we can write:

Degree (<T>) = max {degree (pi) / pi ∈ <T>}

Example:



+Path between node pi to node pj: is a series of nodes from pi to pj and on those nodes there must be branches to connect 2 nodes together.Example:

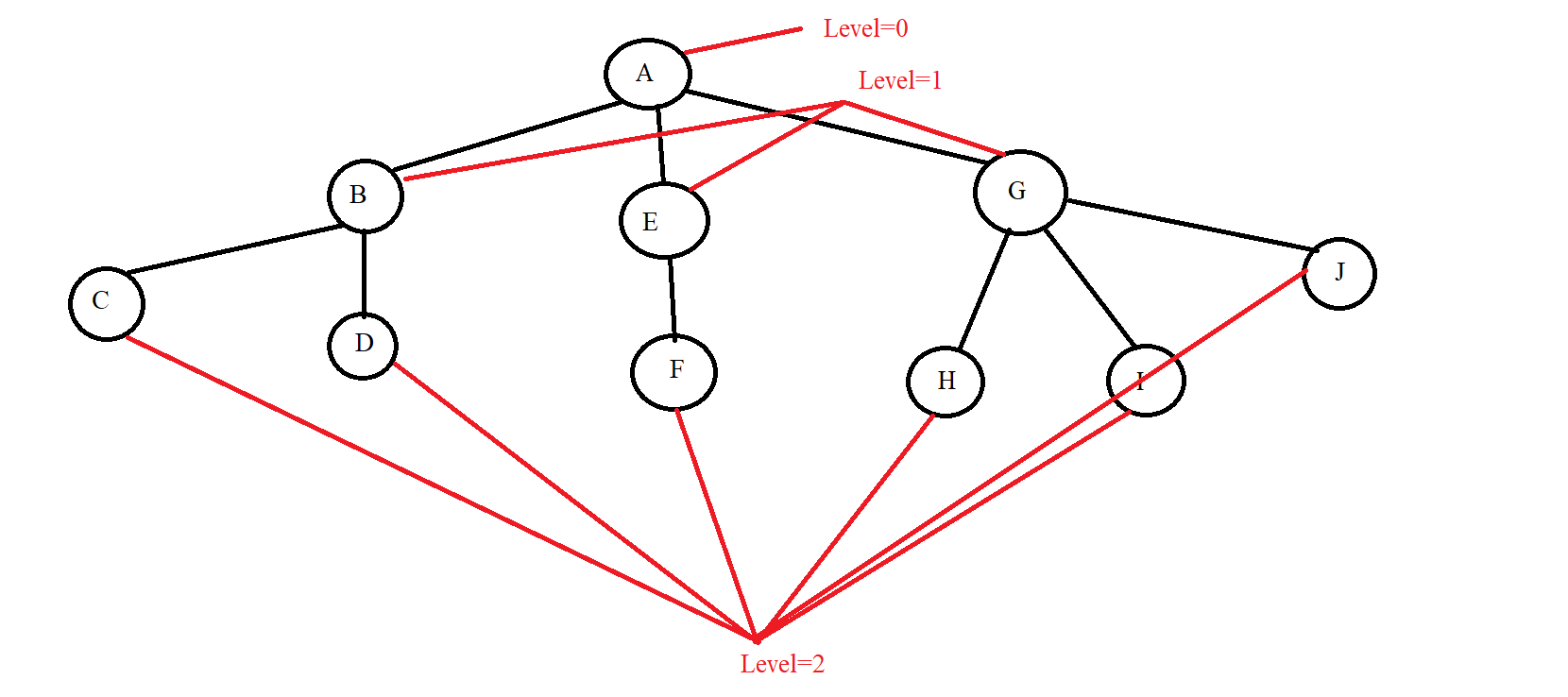


+Level: Just like how to calculate the floor of a whole tower, we apply it to trees as well. Where the ground floor is the root with level =0 and the upper floors are considered the bottom nodes of a node, the level is calculated as the parent node's level plus 1. From this we can deduce the formula:

Level(p) = 0 if p = root

Level(p) = 1 + level(parent(p)) if p! = Root

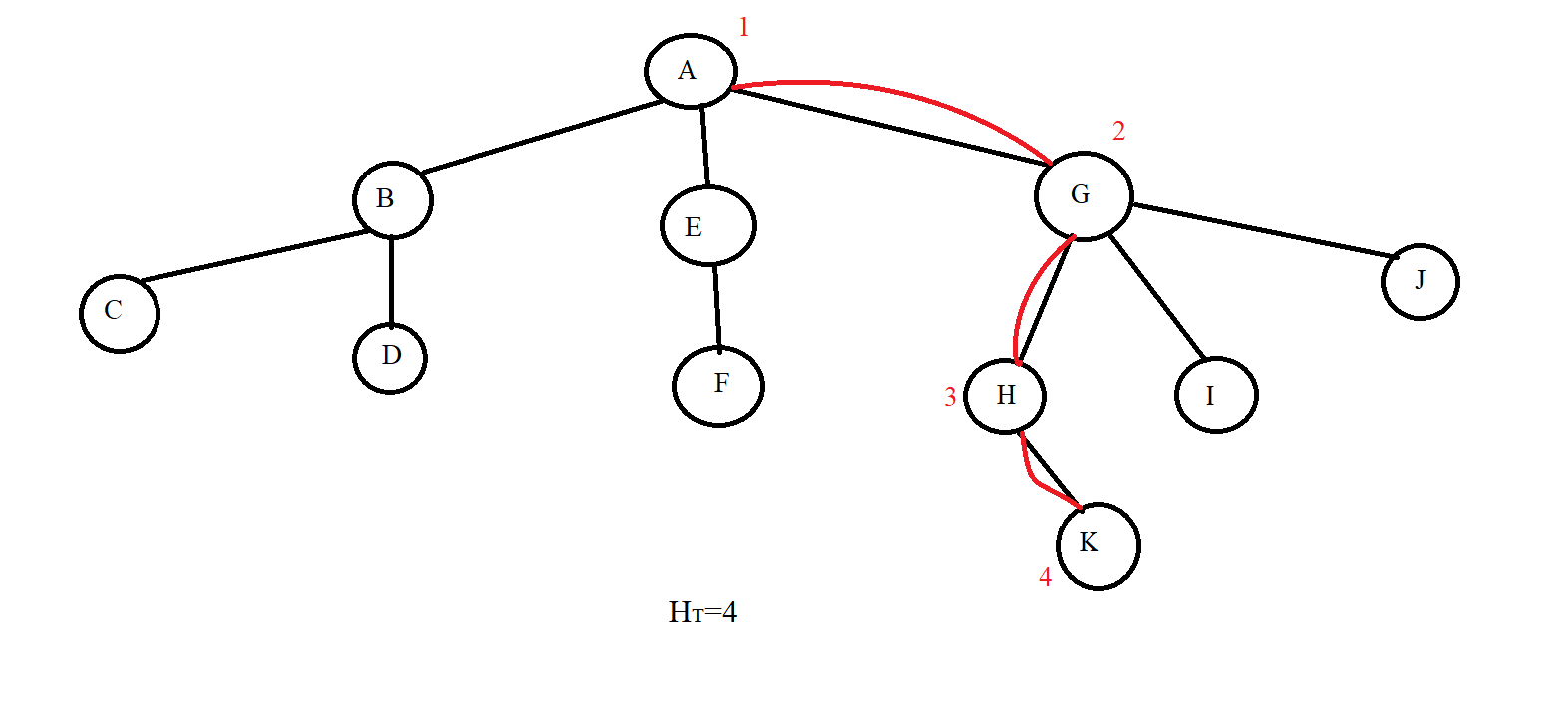
Example:



+Height of tree (hT): Maximum value of path from root to leaf node. We have the mathematical formula:

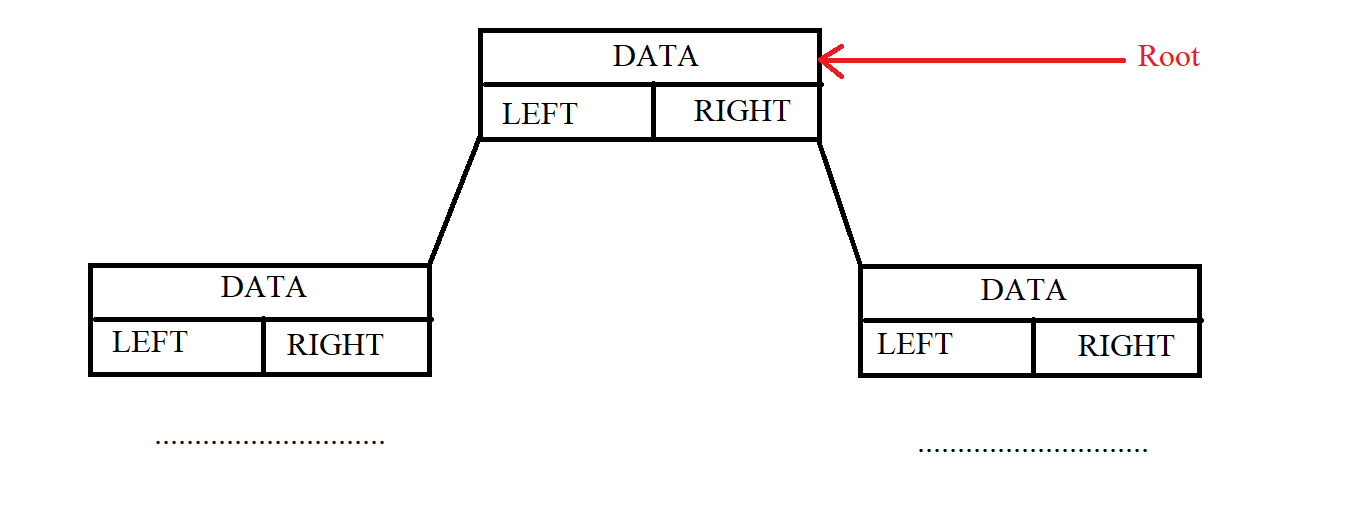
hT = max {Path(root, pi) | pi is the leaf node ∈ <T>}

Example:



1. **Binary tree :**

- A tree <T> (Tree) is a binary tree when the degree of the tree is 2.

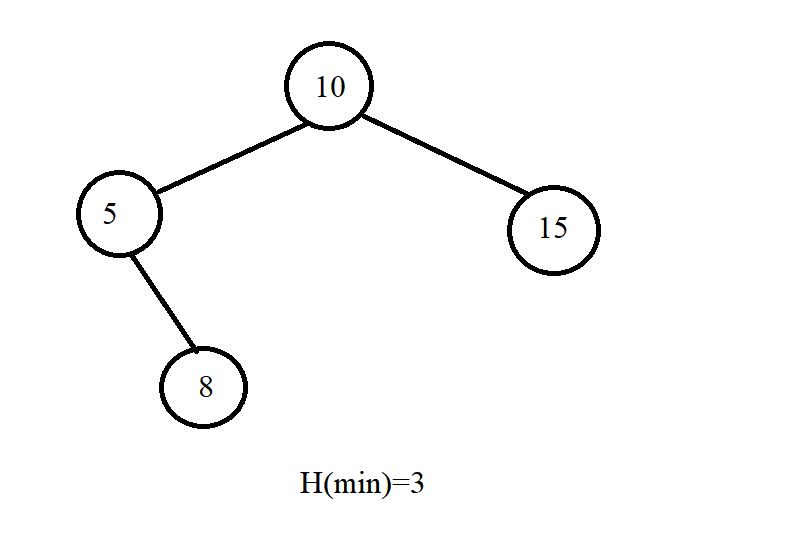


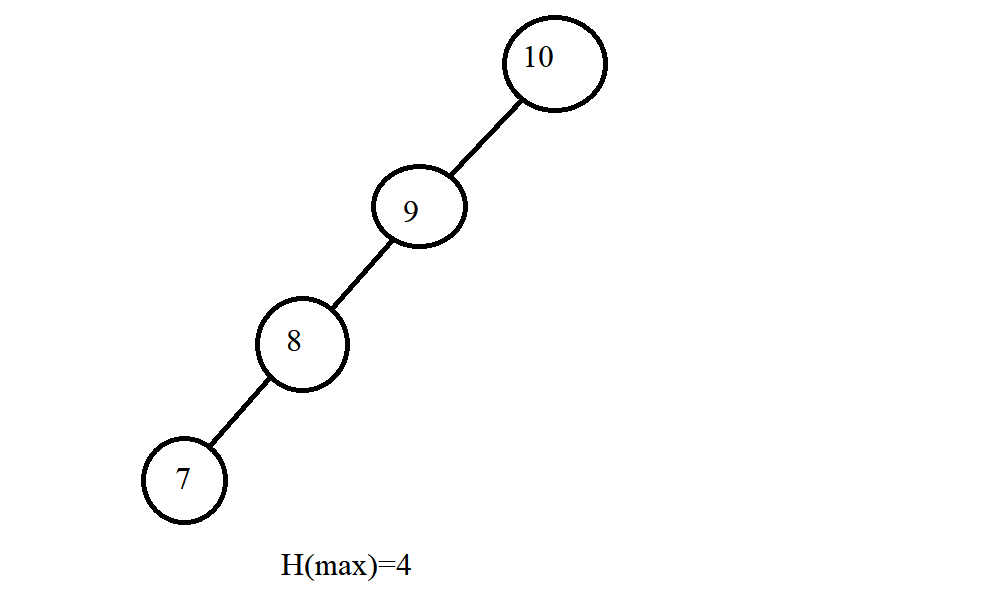
- The height of a binary tree has N nodes:

hT(max) = N

hT(min) = + 1

Example:with N=4





- There are 2 ways to organize a binary tree:

+Array.

+ Structure pointers.

\* Here it is best to use pointers because it is easier to use pointers to delete, insert and search. Pointers make it easier to know the child node of a node.

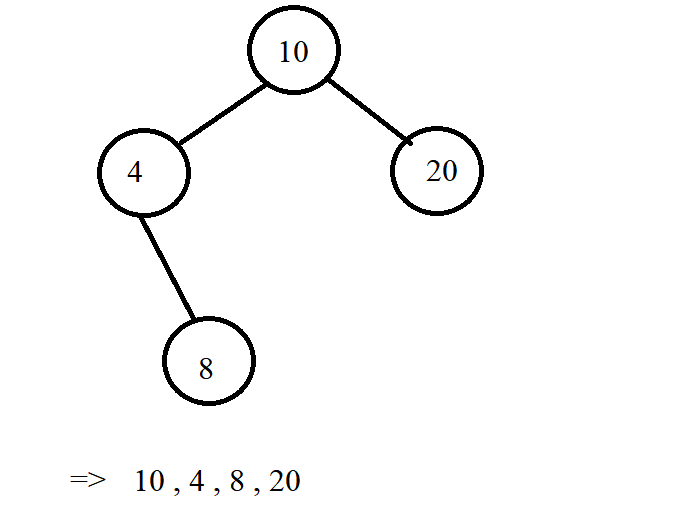


- Traverse in Tree: We have many different ways of traversing such as NLR, LRN, RNL, ... but we only consider 3 basic traversals: NLR, LRN and LRN.

+ Pre-Order (NLR): It will process the current node and then recursively call the left node and finally the right node. Put the case we want in the buttons in we have the following examples



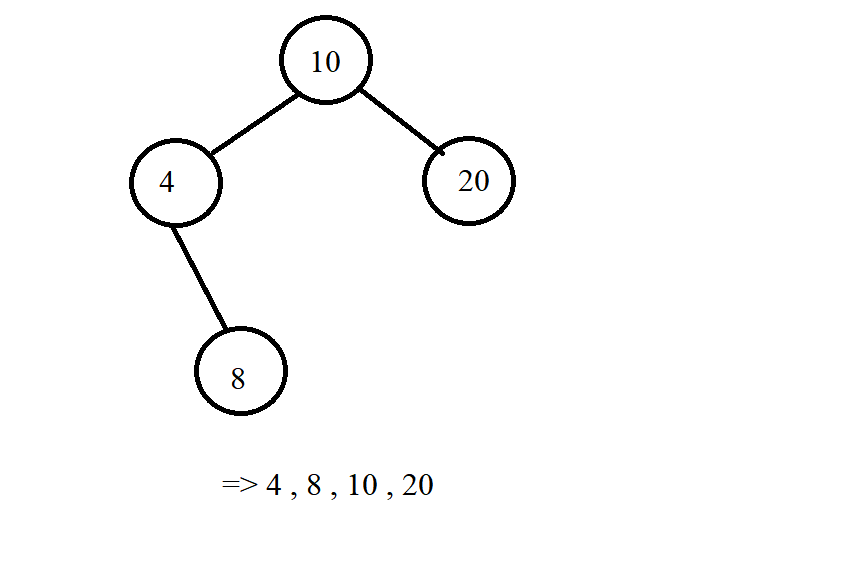
Example:



+ In-Order (LNR) : It will recursively call the left node. Then it will process the current node and finally recursively call the right node.



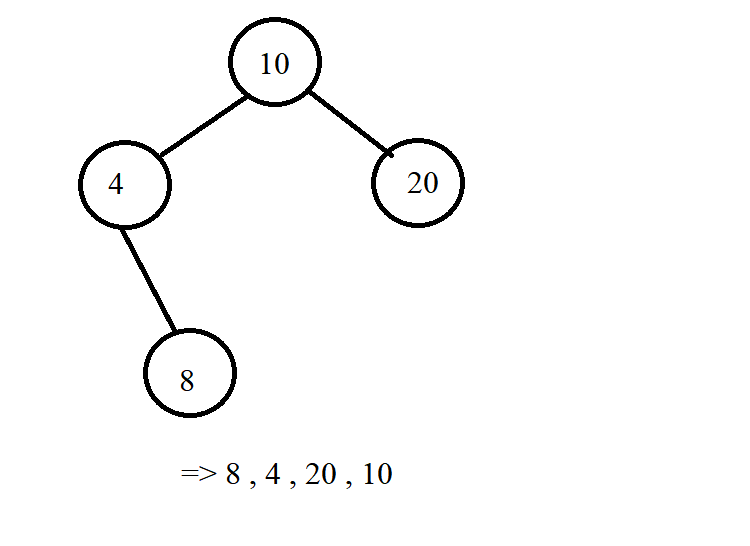
Example:



+Post-Order (LRN): We will recursively call the left node, go to the right node and finally process the current node.



Example:



\*Note: In addition to printing nodes like the examples above, we can change the place (cout<< pCurr->Data << " ") to whatever command we want to handle the query , the subject we want.

1. **Binary search tree (BST):**