Q1. Explain non linear data structure?
Ans: A non-linear data structure is a type of data structure where elements are not arranged sequentially or linearly, instead , they are connected with multiple ways . Allowing for complex relationship between elements. These are useful for hierarchical or network data

1. Tree:
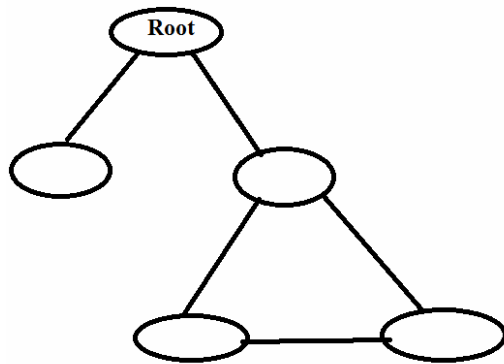
Tree Terminology:

Q1. Explain Tree in Data Structure?
Ans-->Tree is a non-linear data structure which organize data in hierarchical structure and this is a recursive definition
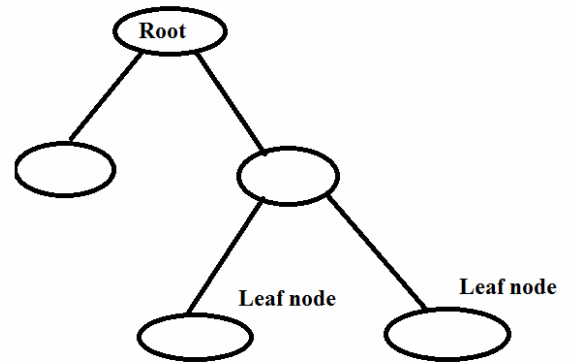OR
A tree is a connected graph without any circuits.
OR
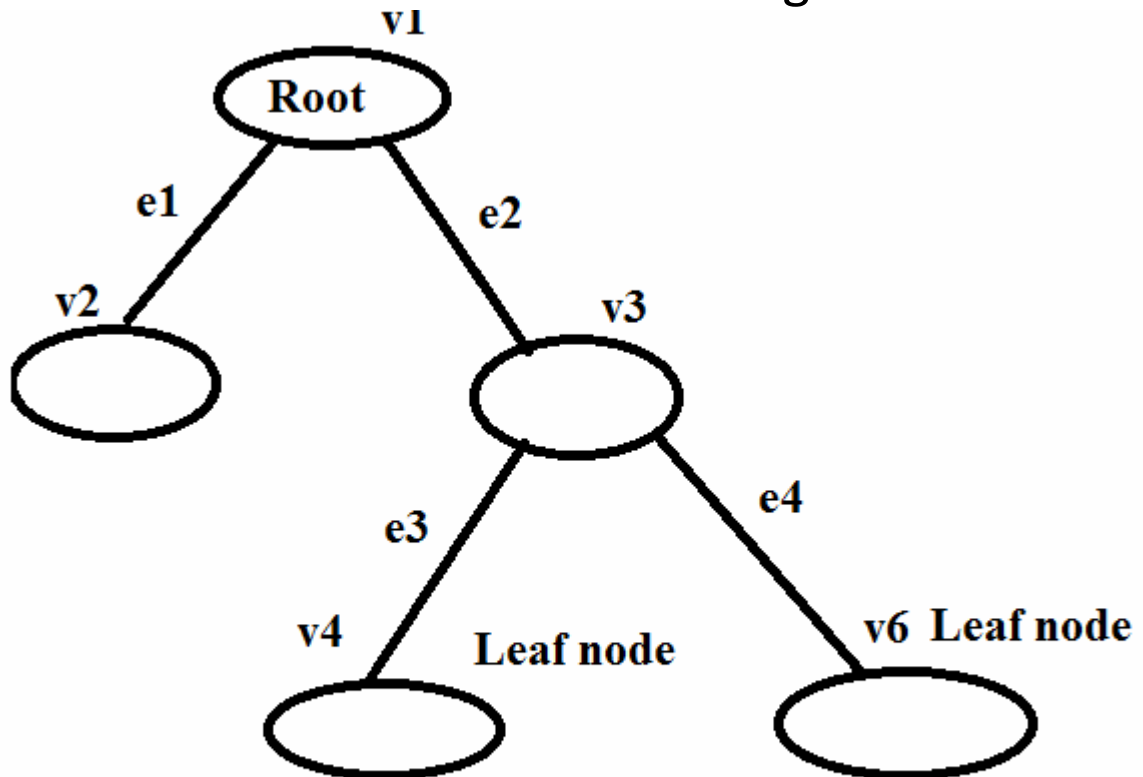If in a graph, there is one and only path between every pair of vertices. then the graph is tree.

Graph which is not a tree

Graph which is tree

Properties of trees:
1. A tree with n vertices has n-1 edges

Graph which is tree

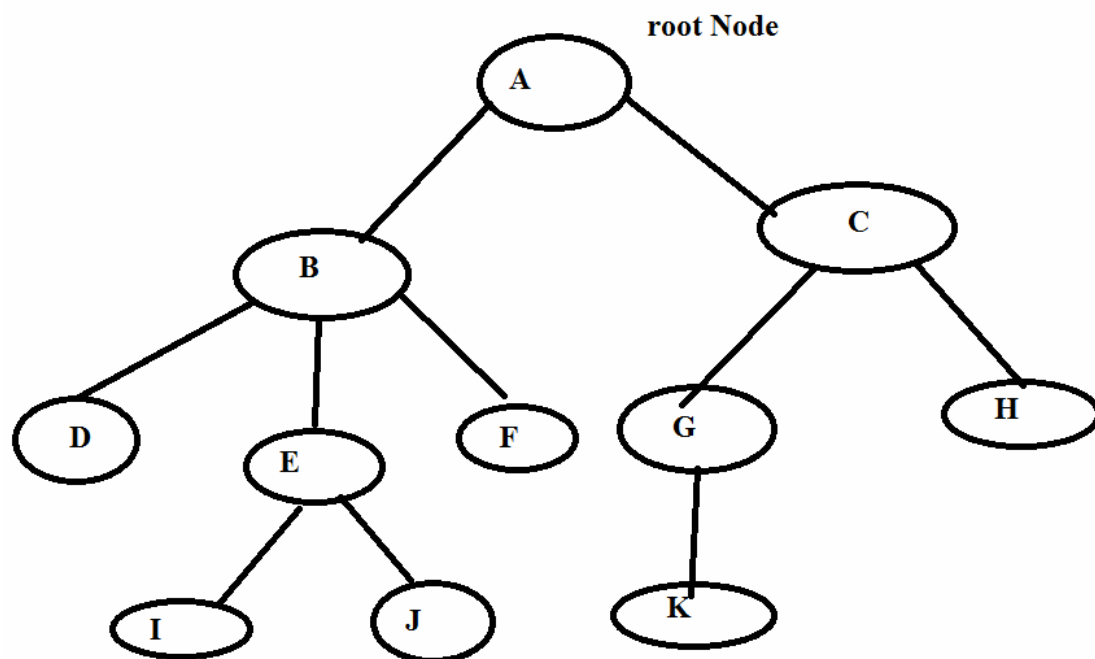## 2. A graph is a tree if and only if it is minimum connected

3. Any connected graph with n vertices and n-1 edges is a tree.

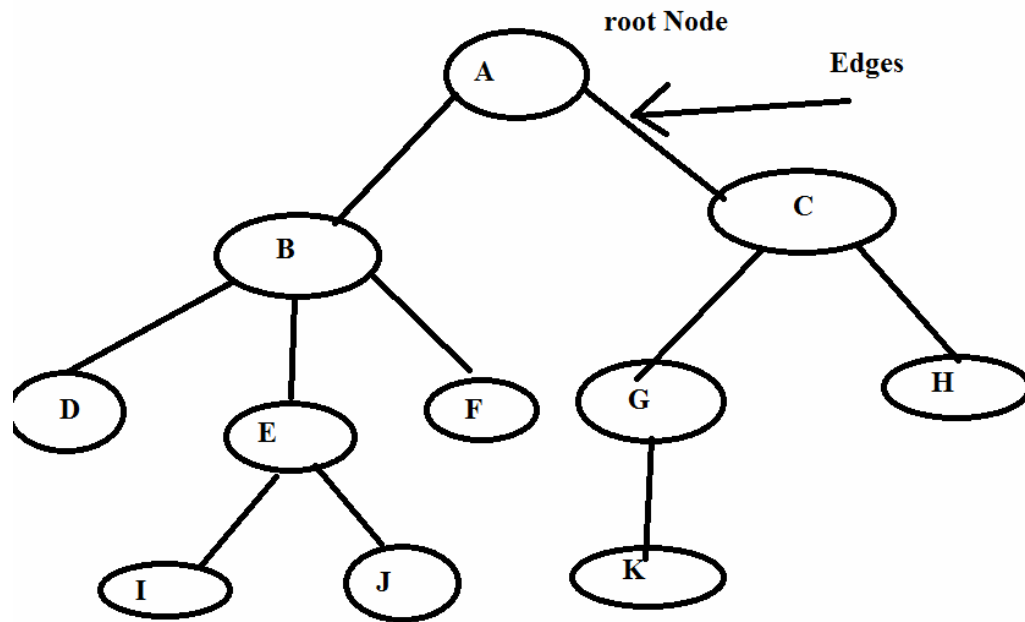
Q2. Explain Basic Terminology of tree

Ans-->

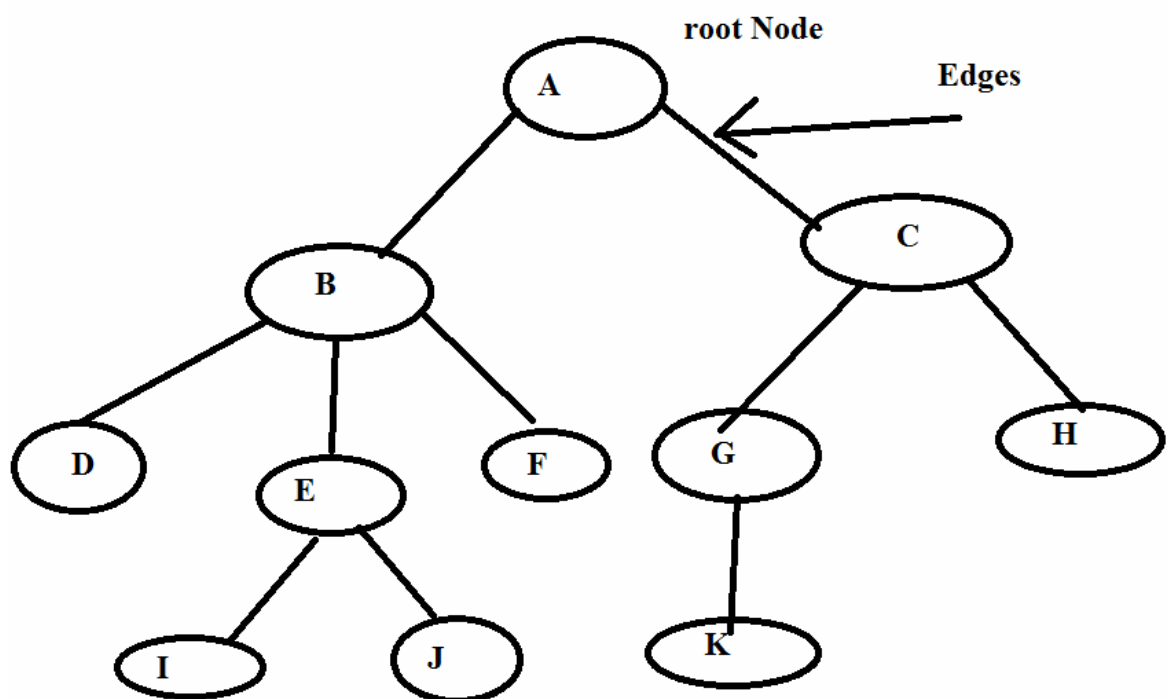**1. ROOT:** Root node is the origin of tree data structure. it is the first node

There must be only one root node and we can never have multiple root nodes in a tree.

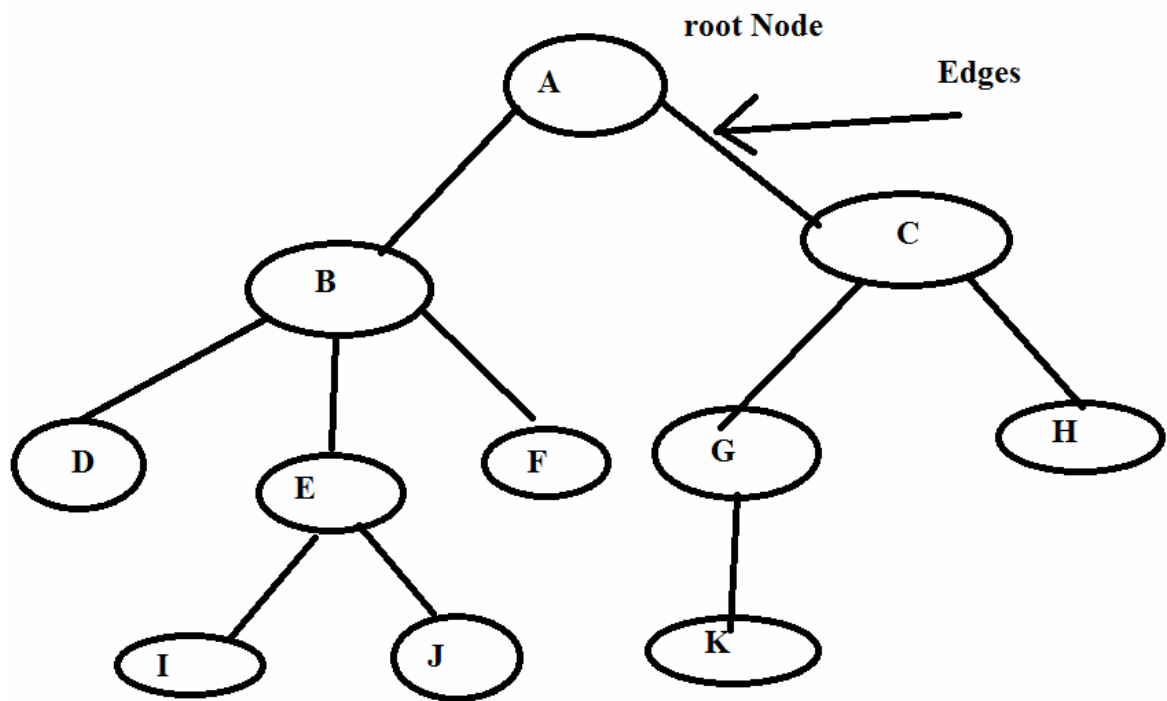2. Edge: The connecting link between only two nodes is called edges.

3. Parent: The Node which has a branch form it to any other node is called as parent node.
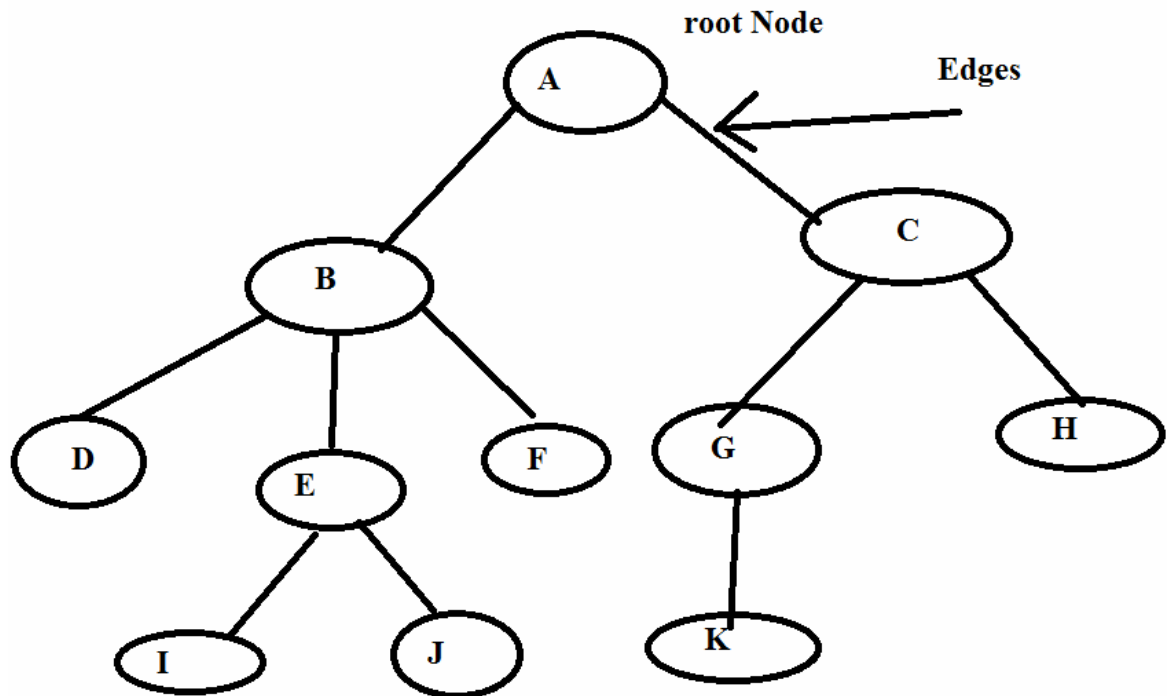

Example: A,B,C,E and G are parent Nodes

4. Child: The node don't have any other node is called child node.

1. Here B and C are child node of A
2. G and H are child node of C
3. K is the only one child of G node
4. I and J is child node of E

5. Siblings: Nodes which belongs to the same parent are called as sibling. in other words node with the same parent is called sibling node

1. B and C are sibling
2. D, E and F are sibling
3. G and H are sibling
4. I and J are sibling

6. Degree: The total number of child of a node is called as degree of that node degree of a tree is the highest degree of a node among all the node in a tree.
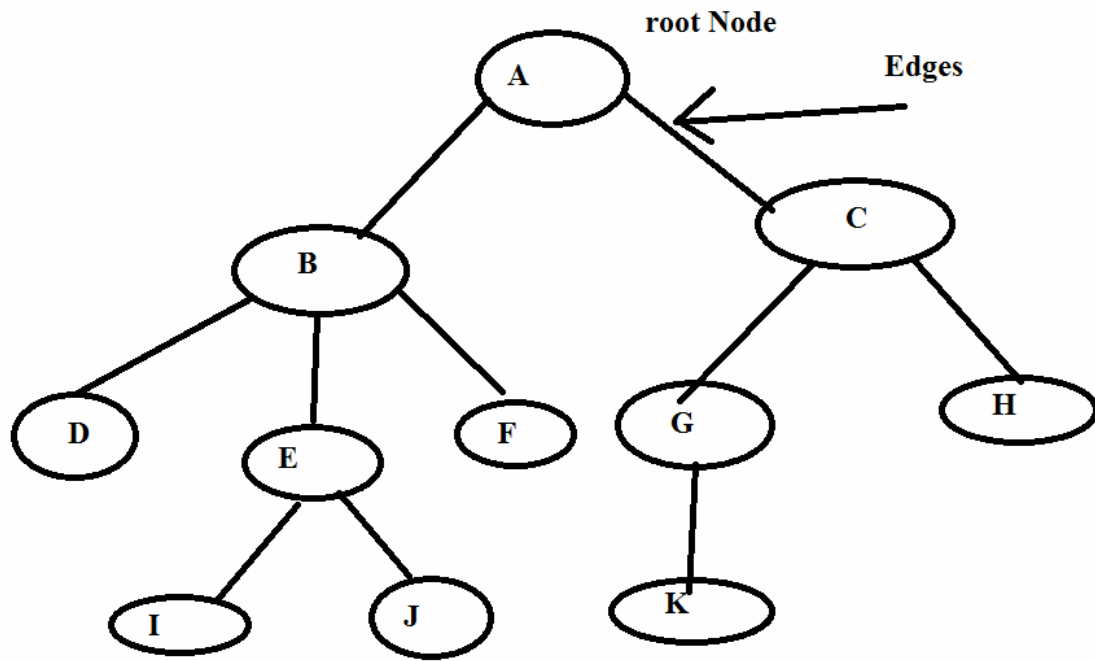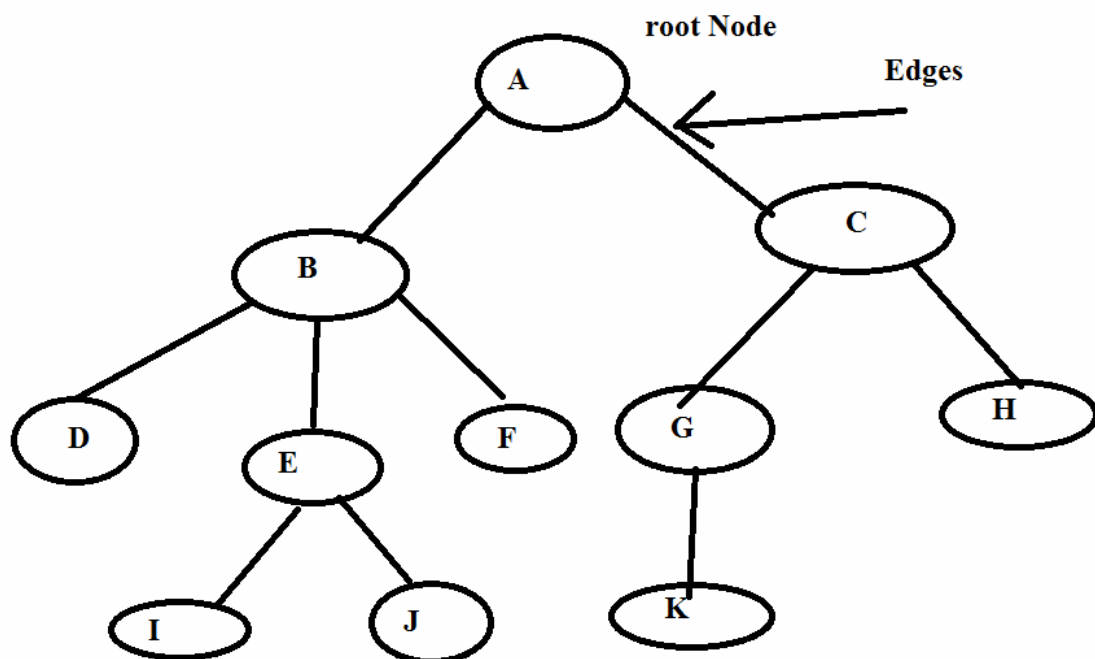
here
1. Degree of B: 3
2. Degree of A: 2
3. Degree of F: 0

7. Internal Node: The node which has at least one child is called as internal node
They are also called non-terminal nodes.
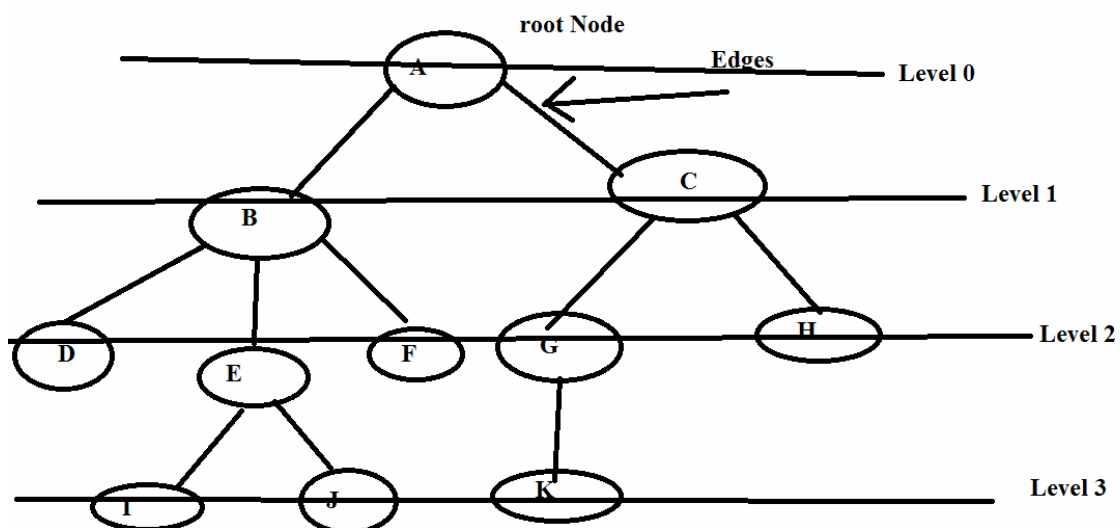Every non-leaf node is an internal node
Example
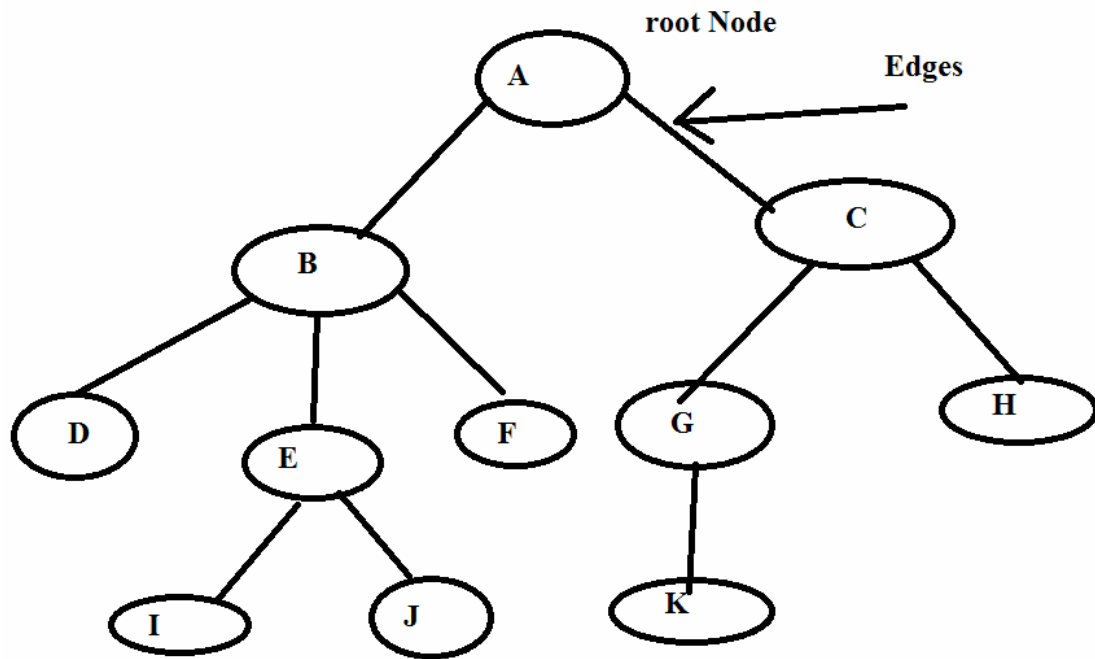
Example: A,B,C E and G

8. Leaf Node: The node which does not have any child is called as leaf node
The leaf nodes are also called as External node/ terminal node

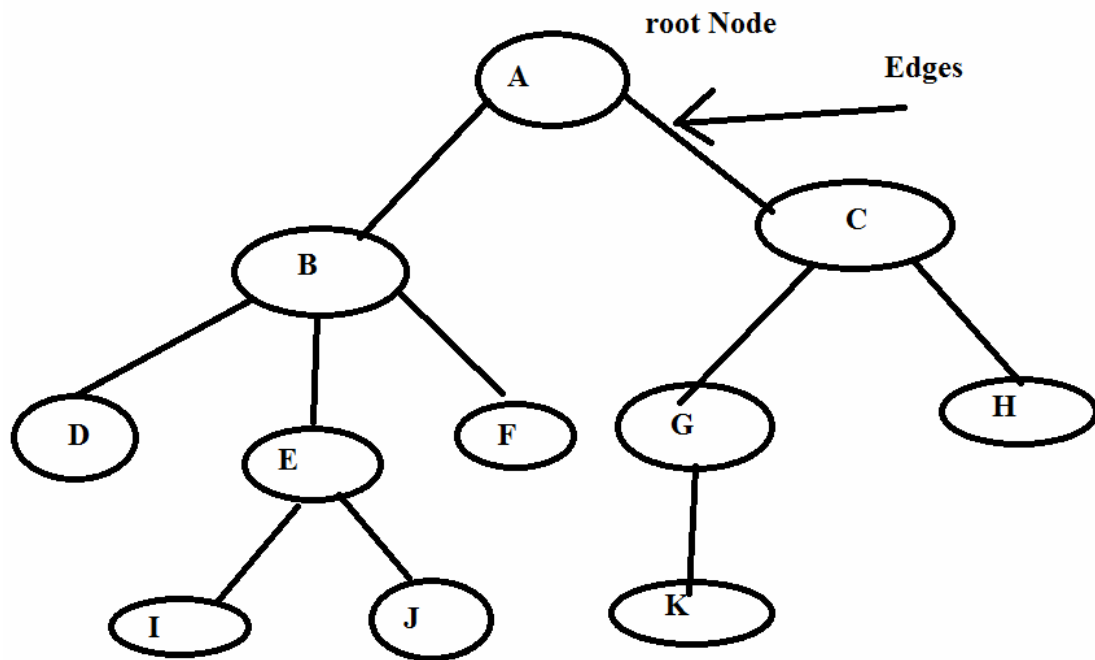**Example" D,I,J,F,K and H are leaf Nodes.**

9. Level: In a tree, each step from top to bottom is called level and level count starts with '0' and incremented by one at each level or step

10. Height : The total number of edges from leaf node to a particular node in the longest path is called height of that node
Example:
Height of the tree=Height of the root node

1. Height of K : 0
2. Height of B: 2
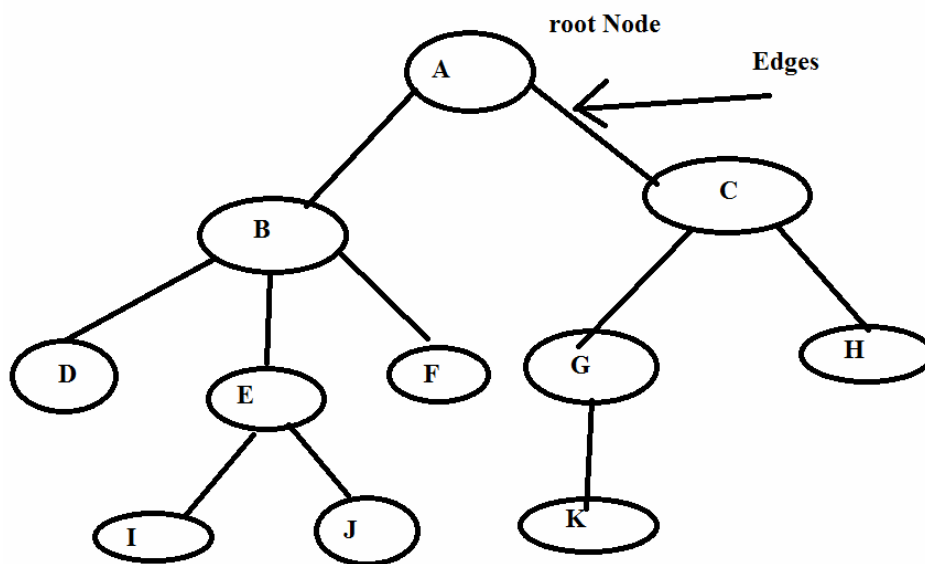3. Height of A : 3
4. Height of G:1
5. Height of H: 0
Height of tree is: 3(Height of root node)
Node: Height of all leaf node is 0.

14. Depth:

The total number of edges form root node to particular node is called as depth of that node
Depth of tree= Total Number of edges from root node to a leaf node in the longest path
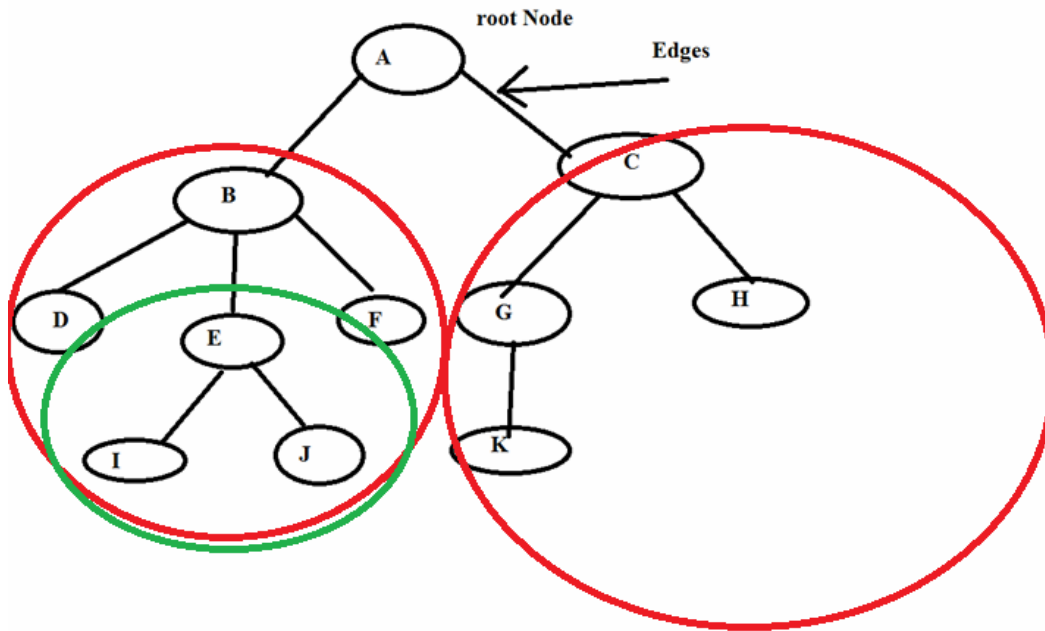

Depth OF Node A: 0
Depth OF B: 1
Depth OF D:2
Depth OF K: 3
Depth of tree: 3

13: sub tree: In a tree, each child from a node form a sub tree recursively. Every child node will form a sub tree on its parent node.

Pre Order Tree Traversal: ROOT LEFT RIGHT

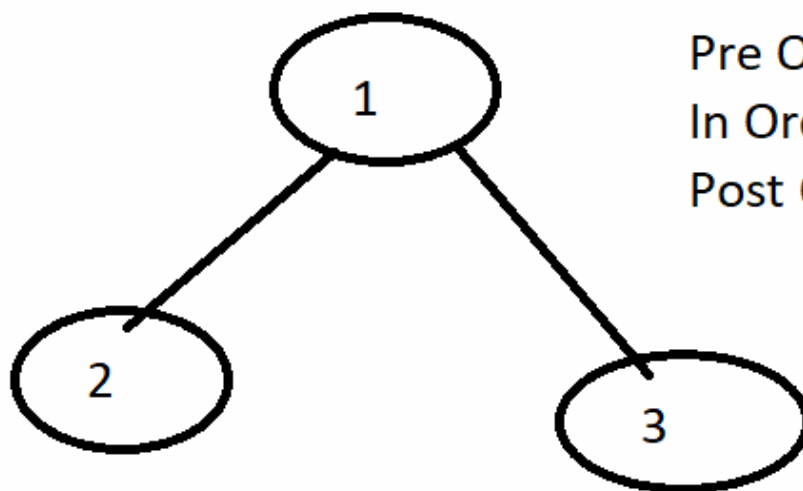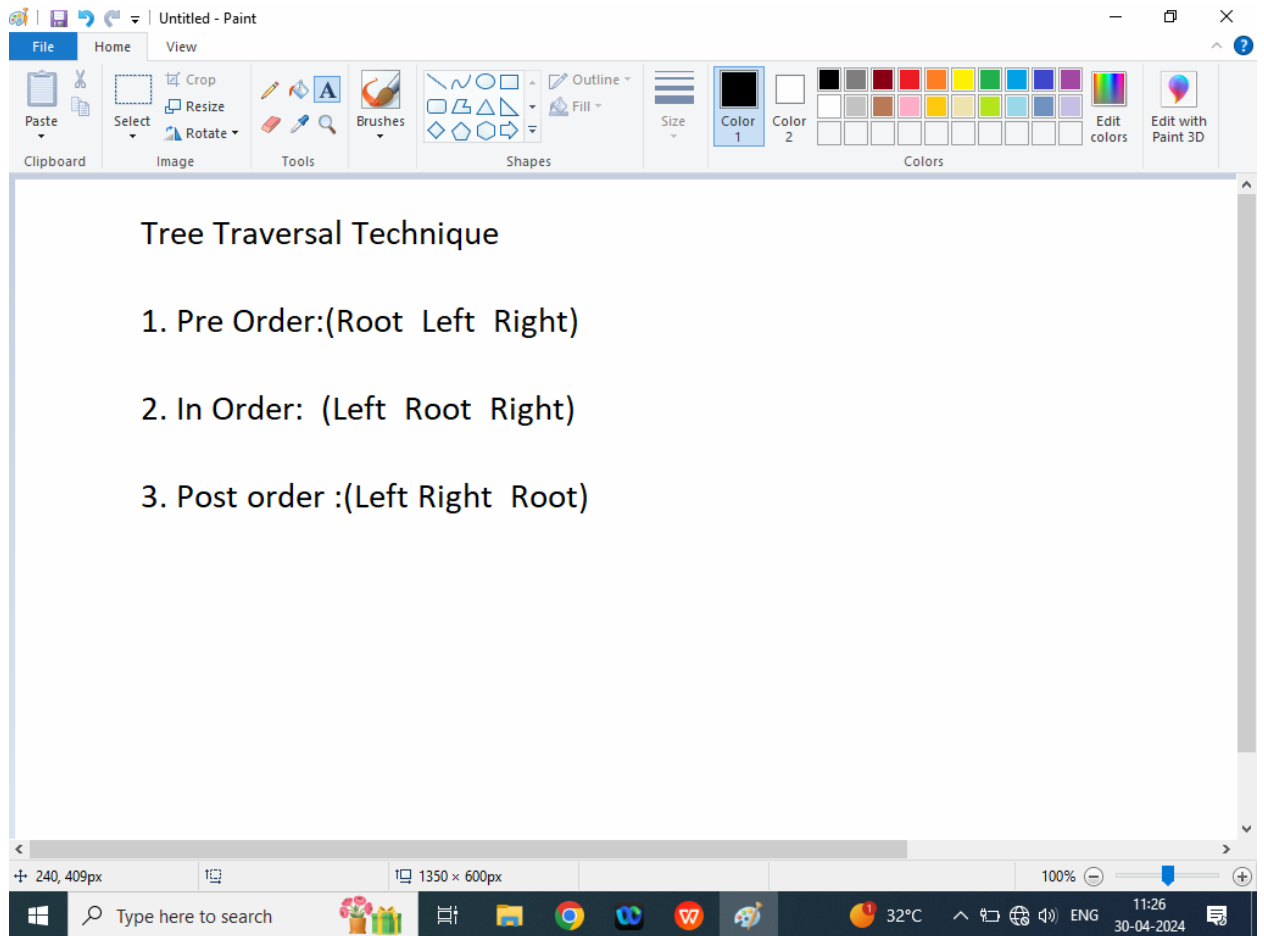In Order Tree Traversal: LEFT ROOT RIGHT

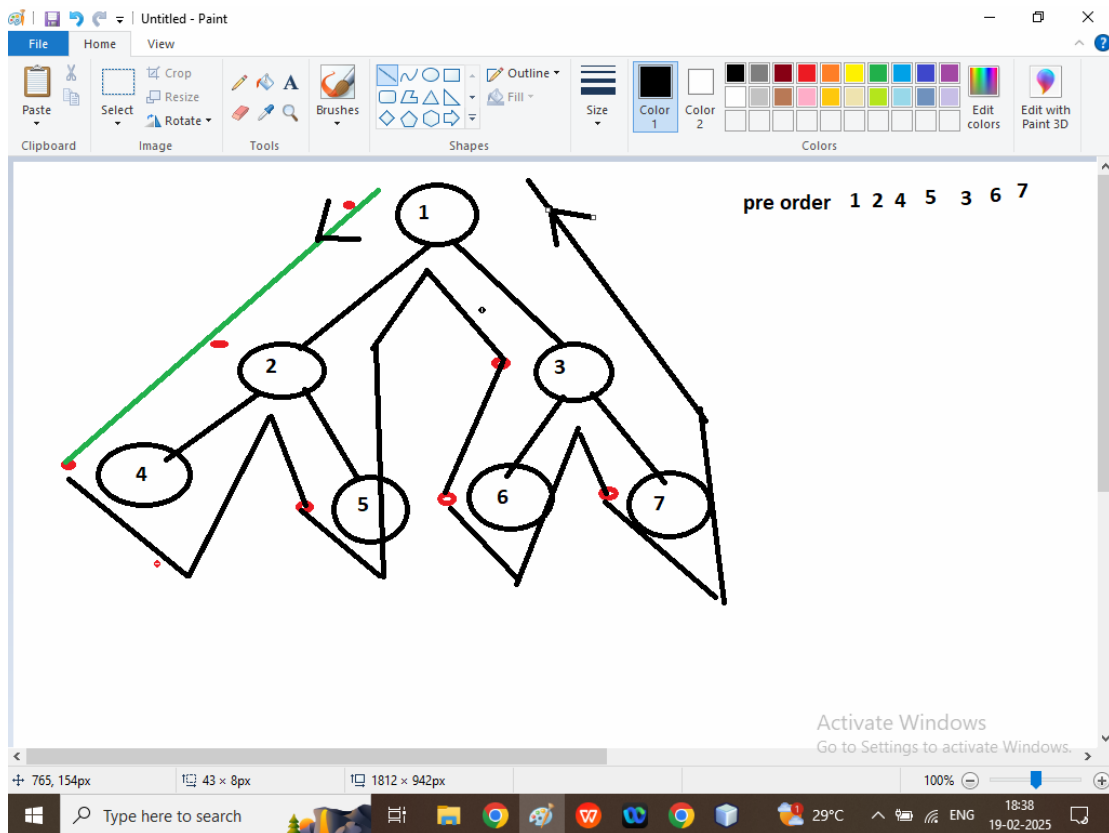Post Order Tree Traversal: LEFT RIGHT ROOT
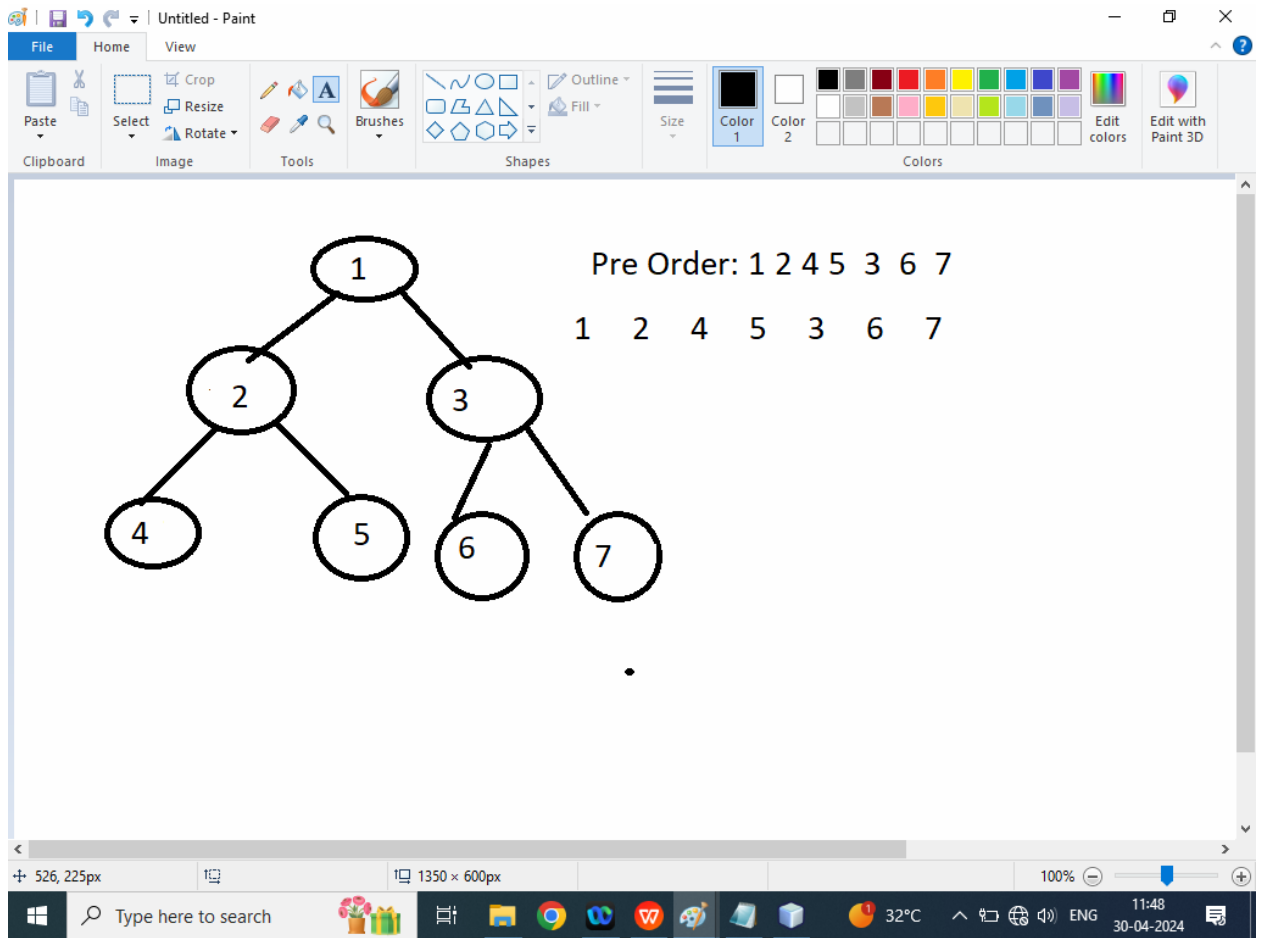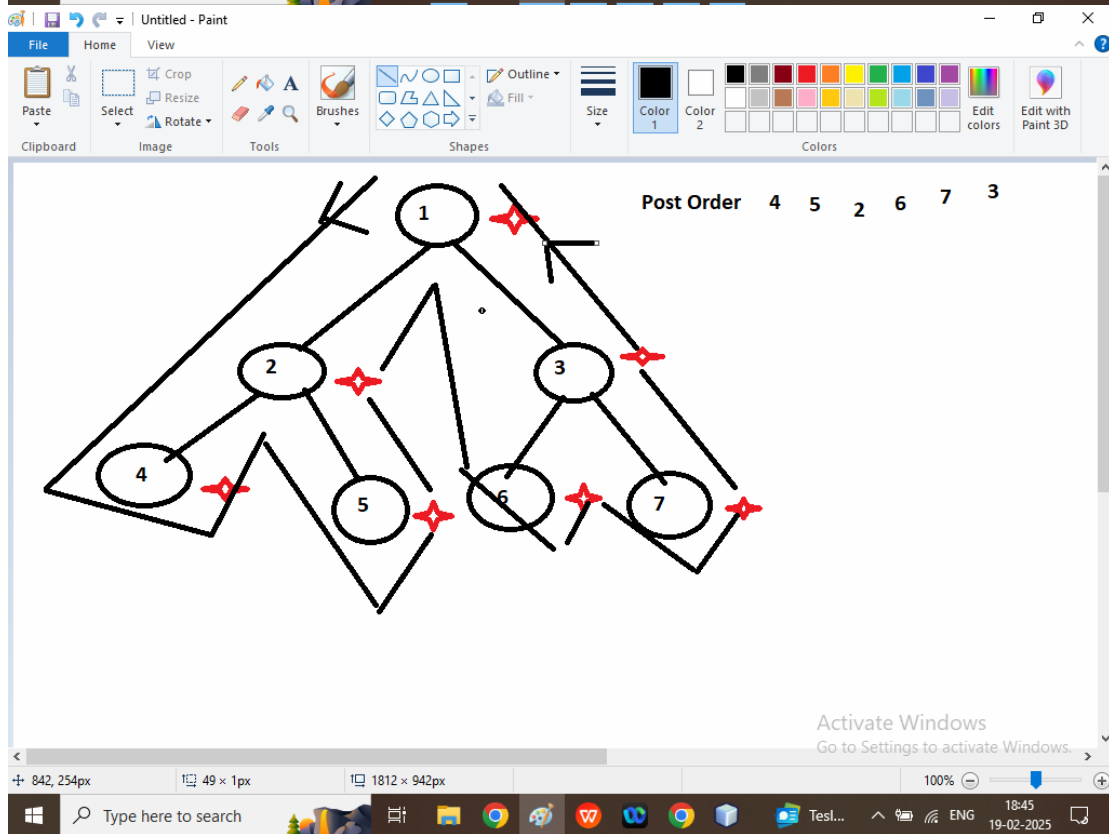
pre order: 1 2 3
in order: 2 1 3
post order: 2 3 1

Tree Traversal Technique

1. Pre Order:(Root Left Right)

2. In Order: (Left Root Right)

3. Post order :(Left Right Root)
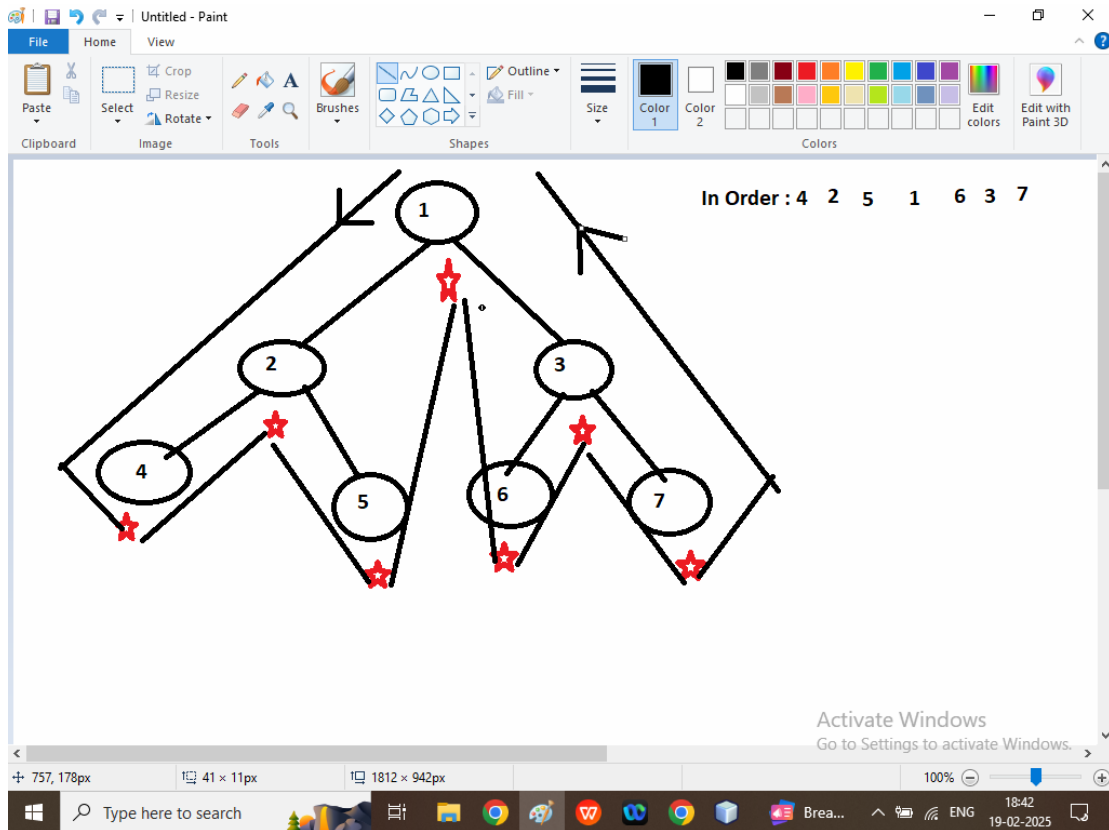

Pre Order: 1 2 3
In Order: 2 1 3
Post Order: 2 3 1

Pre Order: 1 2 4 5 3 6 7

1   2   4   5   3   6   7

pre order 1 2 4 5 3 6 7

In Order : 4 2 5 1 6 3 7

Post Order 4 5 2 6 7 3

Pre Order: Root Left Right

In Order: Left Root Right

Post Order: Left Right Root


```java
class Node{
int data;
Node left;
Node right;
}
public Node(int data){
this.data=data;
left=null;
right=null;
}
public static void main(String args[]){
Node t1=new Node(10);
}

}
```

```
class Node{
int data;
Node left;
Node right;
}
public Node(int data){
this.data=data;
left=null;
right=null;
}
public static void main(String args[]){
Node t1=new Node(10);
}

}
```
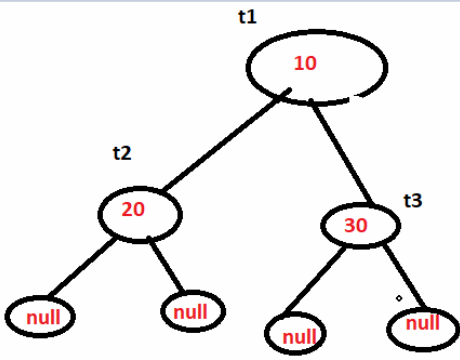


```
class Node{
int data;
Node left;
Node right;
}
public Node(int data){
this.data=data;
left=null;
right=null;
}
public static void main(String args[]){
Node t1=new Node(10);

Node t2=new Node(20);
  t1.left=t2;
Node t3=new Node(30);

    t1.right=t3;
```

2. Graph:

Representation:
1. Using Array
2. Using List