Tree and Binary Tree

Definition

- A **tree** is a widely used abstract data type
- A tree data structure can be defined recursively as a collection of nodes (starting at a root node), where each node is a data structure consisting of a value, together with a list of references to nodes (the "children"), with the constraints that no reference is duplicated, and none points to the root.

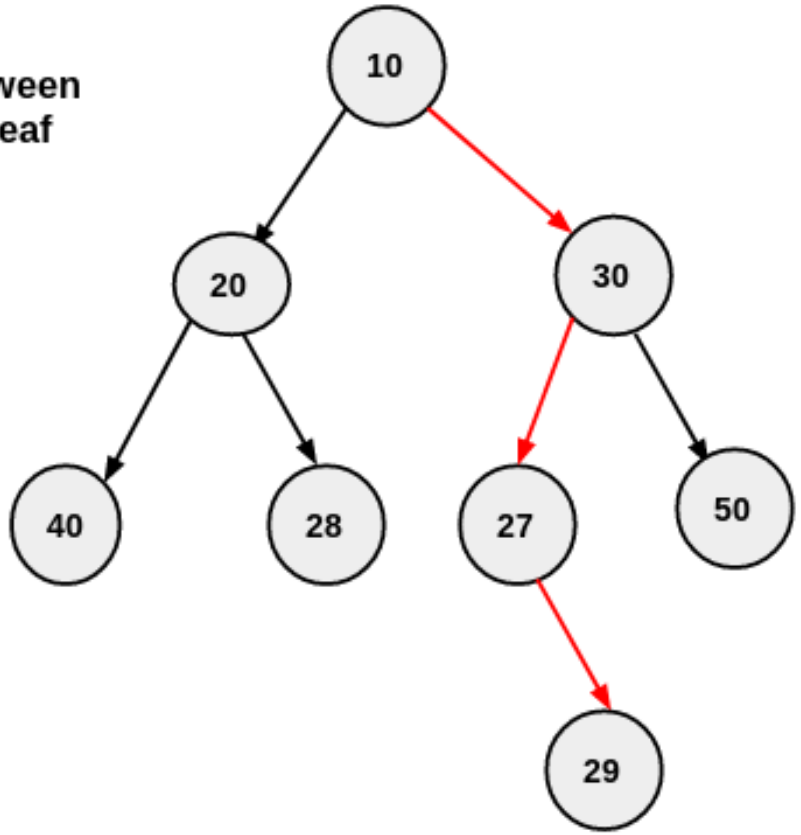
Terminology

- **Root node**: is the topmost node in a tree
- **External node** (also known as an **outer node**, **leaf node**, or **terminal node**) is any node that does not have child nodes.
- **Distance**: the number of edges along the shortest path between two nodes.
- **Level**: the level of a node is the number of edges along the unique path between it and the root node.
  - Level of the root node is 0
- **Degree of node**: degree of a node is its number of children.
  - A leaf has necessarily degree zero
- **Degree of tree**: degree of a tree is the maximum degree of a node in the tree.

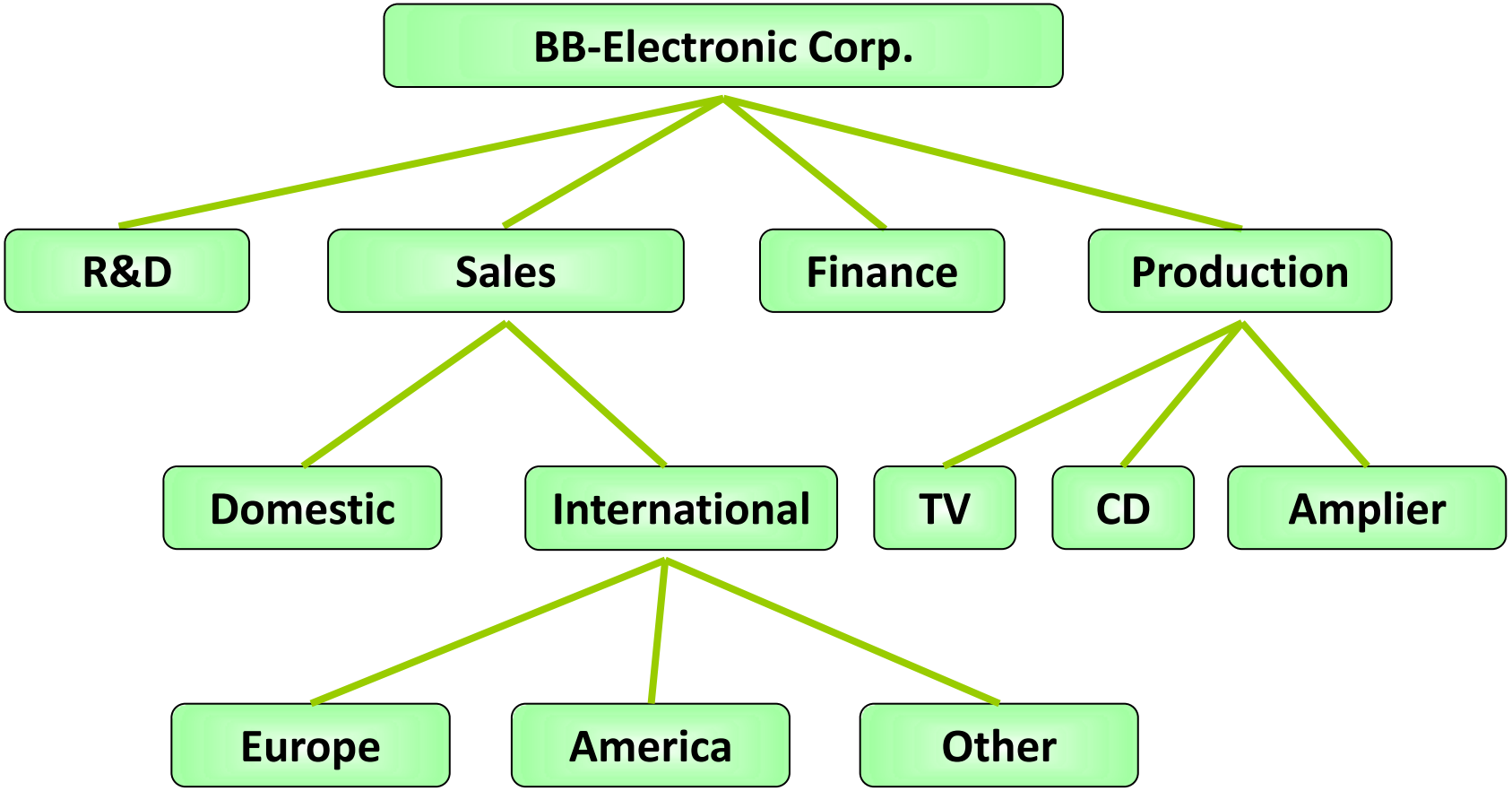
Terminology

- **Height of tree**: the number of edges between the tree's root and its furthest leaf

Number of edges between root and it's furthest leaf node =3.
Hence,
Height of tree = 3.

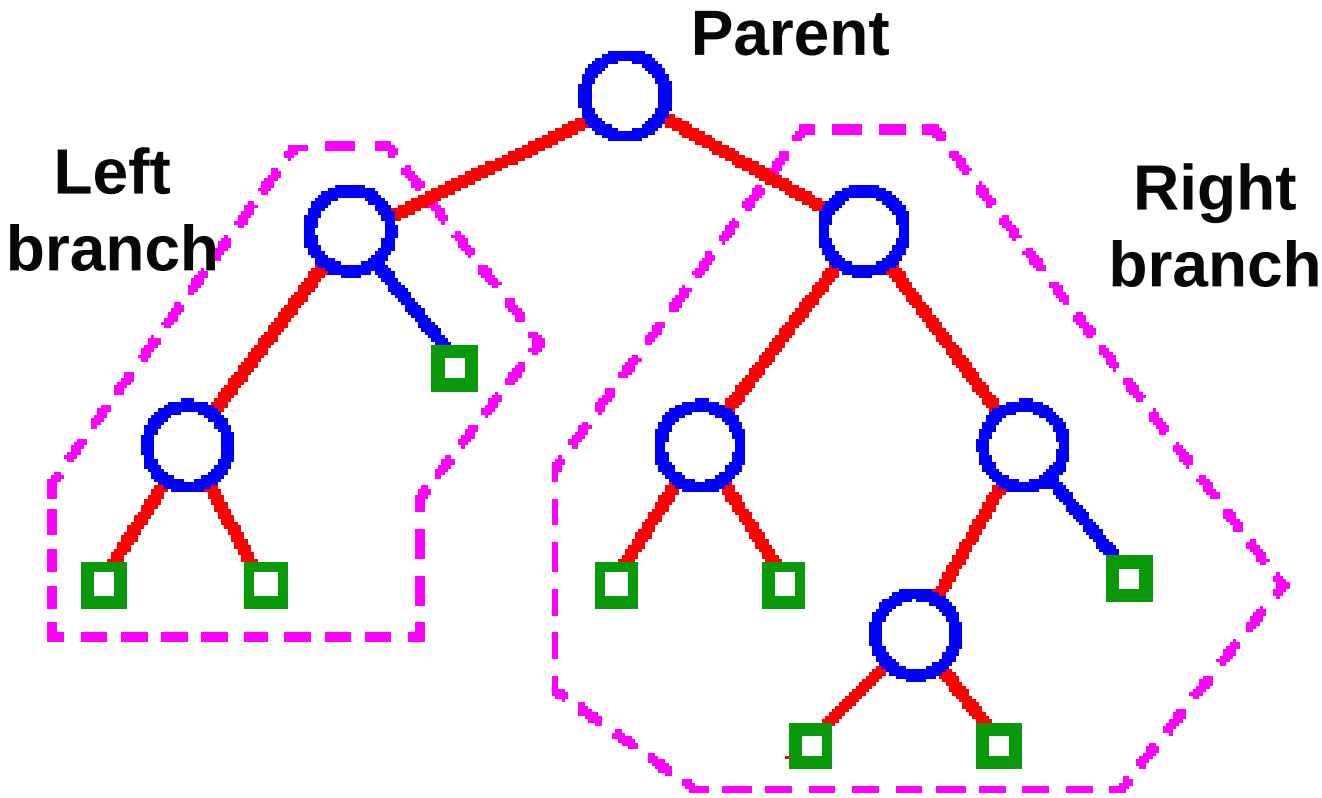


Example of tree



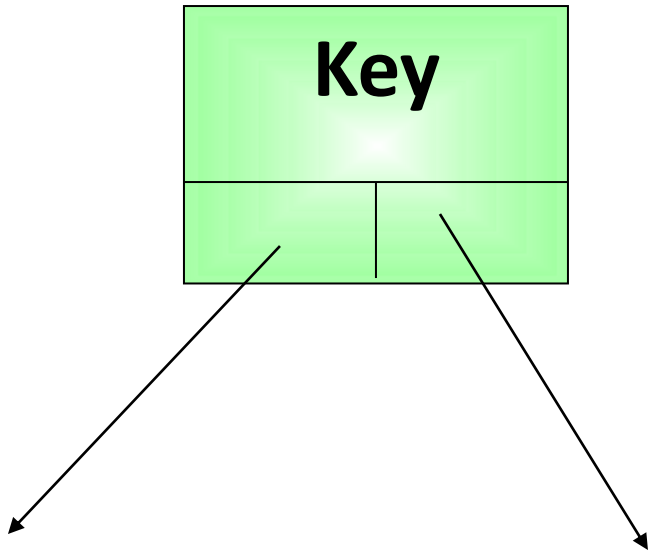
Binary Tree

- A **binary tree** is a tree data structure in which each node has at most two children, which are referred to as the *left child* and the *right child*

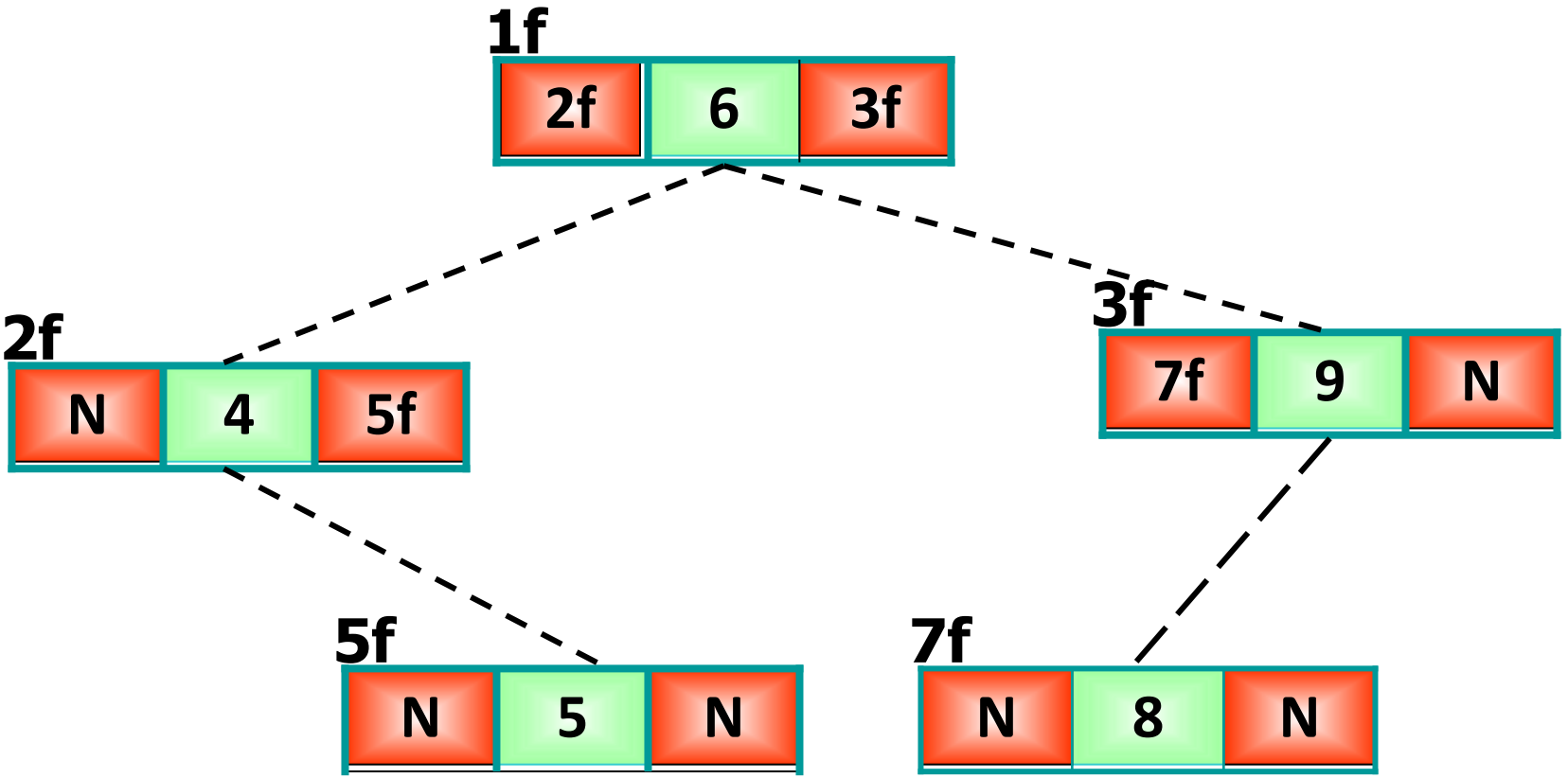


Data structure of node

```
struct TNode
{
   Data     Key;
   TNode *pLeft;
   TNode *pRight;
};
typedef TNode *TREE;
```



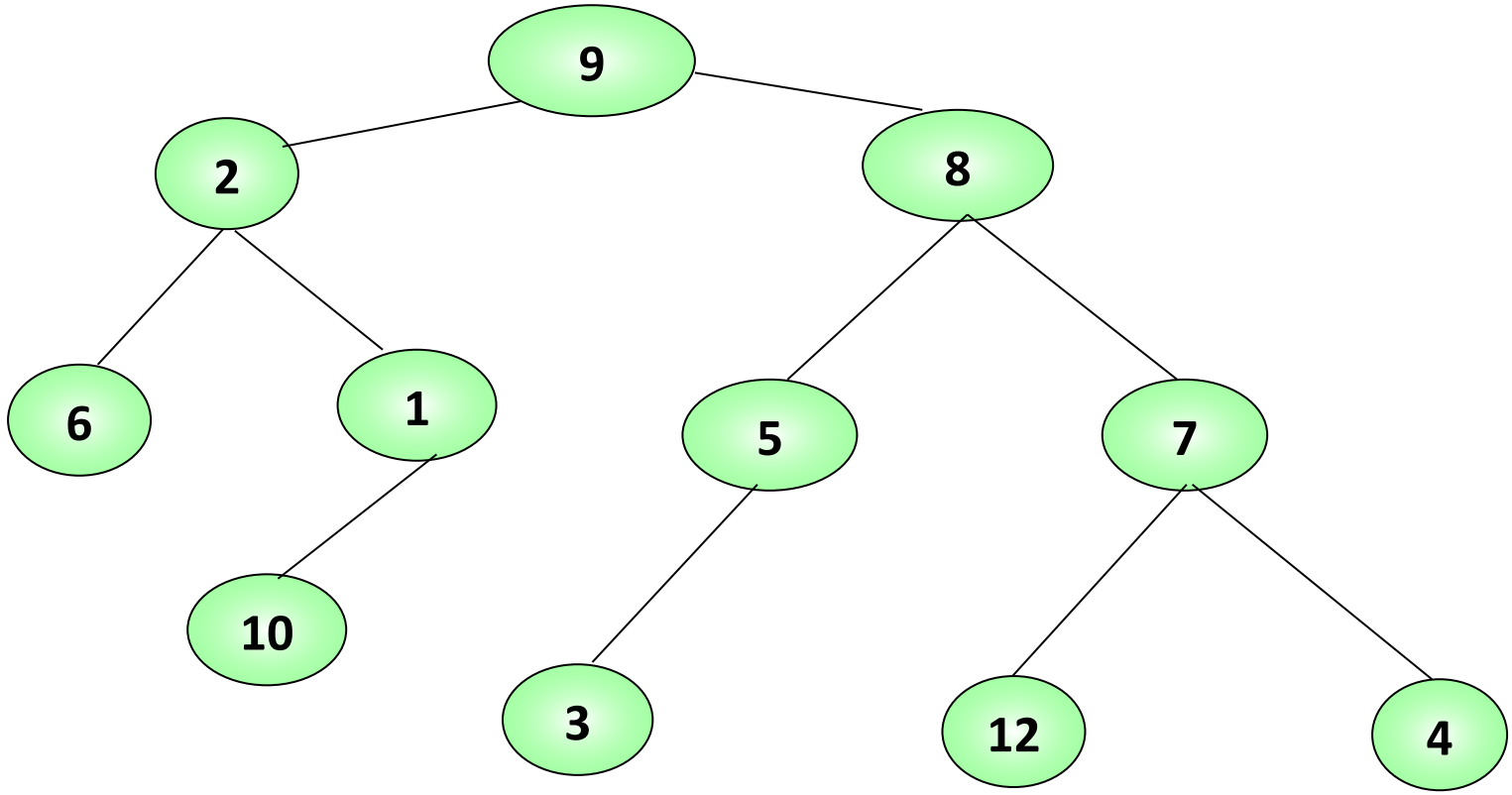
Binary Tree - Example



Traversal

- There are lots of types of tree traversal, such as:
  - Pre-order (NLR)
  - In-order (LNR)
  - Post-order (LRN)
- Complexity: O(log2(h))
  - Where, h is the height of the tree

Traversal - Example



- NLR: 9, 2, 6, 1, 10, 8, 5, 3, 7, 12, 4.
- LNR: 6, 2, 10, 1, 9, 3, 5, 8, 12, 7, 4.
- Find results of the tree traversal according to the following orders: LRN, NRL, RLN, LNR?

Pre-order traversal

```
void   NLR(TREE Root)
{
   if (Root != NULL)
   {
        <Process Root>;
        NLR(Root->pLeft);
        NLR(Root->pRight);
   }
}
```

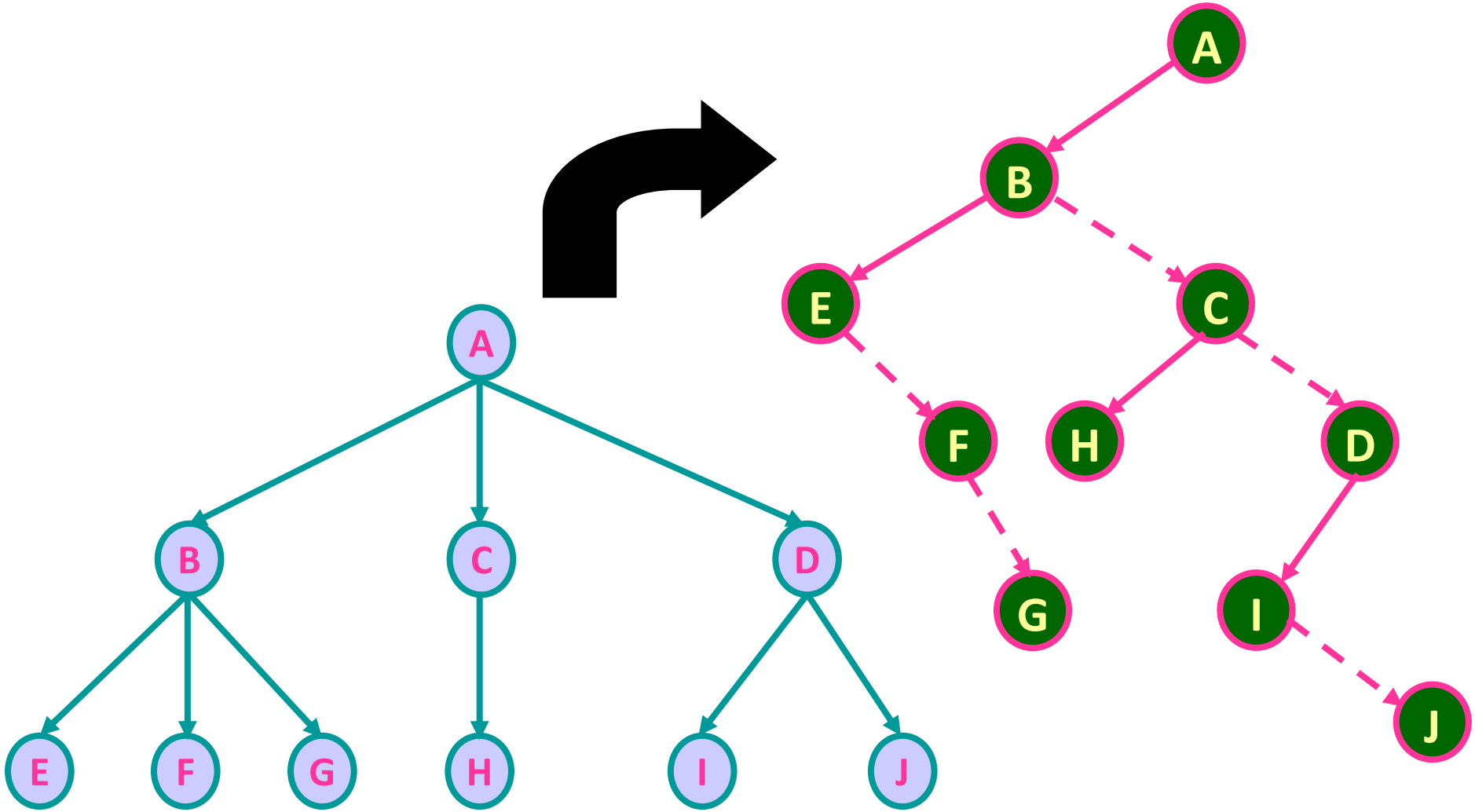

In-order traversal

```
void LNR(TREE Root)
{
   if (Root != NULL)
   {
        LNR(Root->pLeft);
        <Process Root>;
        LNR(Root->pRight);
   }
}
```


Post-order traversal

```
void    LRN(TREE Root)
{
  if (Root != NULL)
  {
     LRN(Root->pLeft);
     LRN(Root->pRight);
     <Process Root>;
  }
}
```


Covert General Tree to Binary Tree



Create a Tree

```
void CreateTree(TREE &T)
{
    int x;
    do
    {
        cin >> x;
        if(x<=0)
            break;
        insertNode(T,x);
    }while(1);
}
```