PRESENTATION ON

Tree

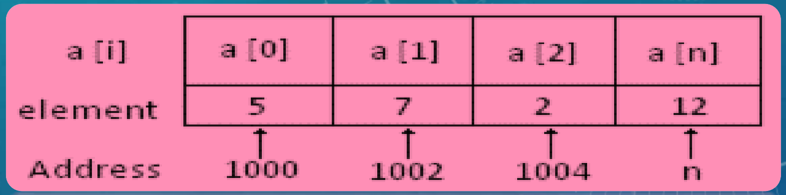
in

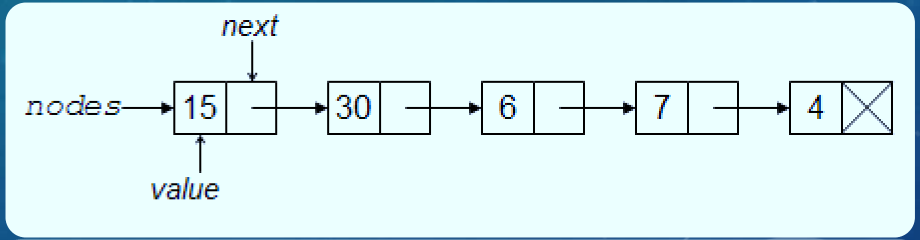
Data Structure

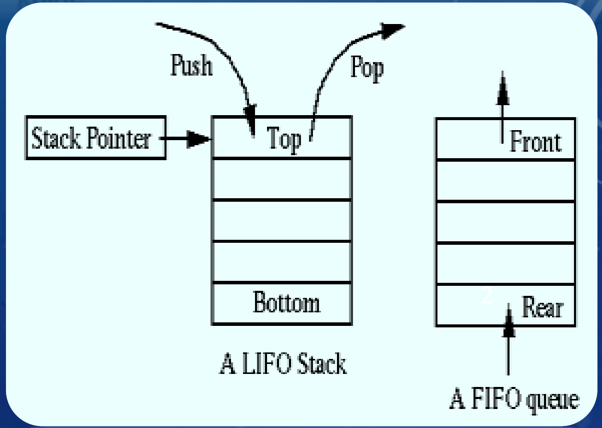
DATA STRUCTURE

Store and organize data in computer.

- ❑ **Linear Data Structure**
  - Arrays
  - Linked List
  - Stacks
  - Queues







LOGIC OF TREE

- Used to represent hierarchica data.

Shuvo(CEO)

Shawon(CTO) Tarun(Presedent)

Bithi Moni Sila Rashel Raj

Dola Shakila Riaj Shakib

- Used to represent hierarchical data.

Shuvo(CEO)

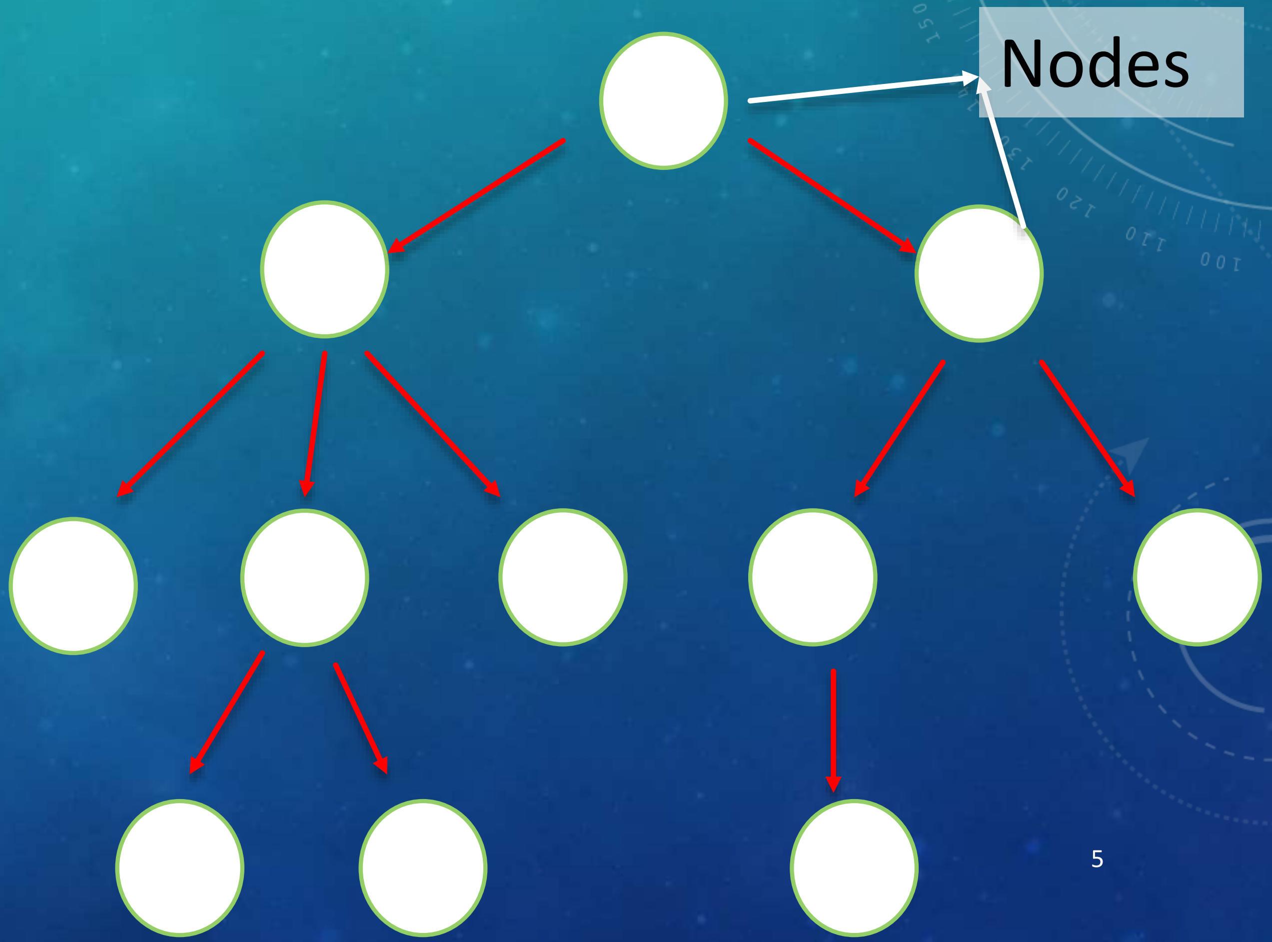
Shawon(CTO)   Tarun(Presedent)

Bithi   Moni   Sila   Rashel   Raj

Dola   Shakila   Riaj   Shakib

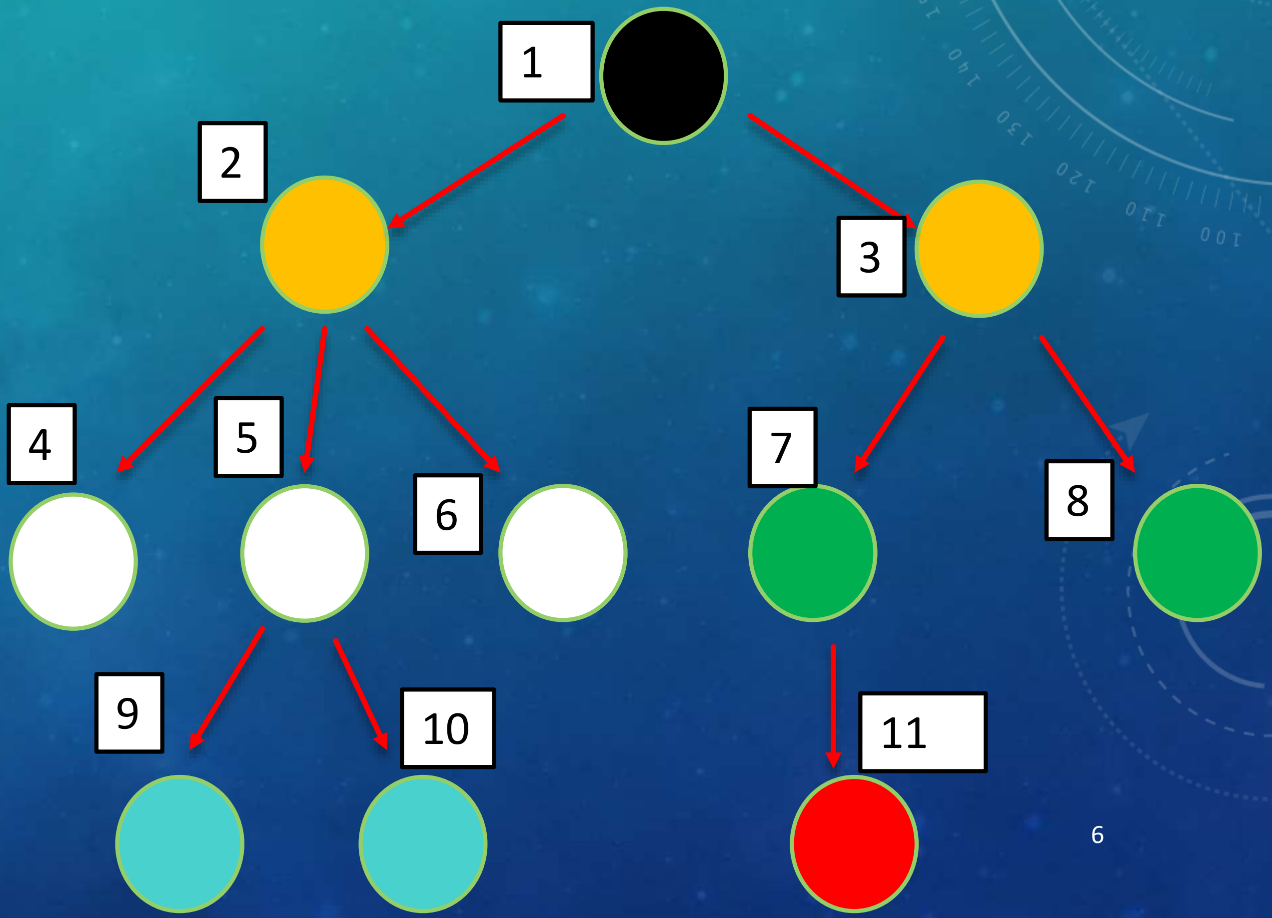
TREE

- A Collection of entities called Nodes.
- Tree is a **Non-Linear Data Structure.**
- It’s a **hierarchica Structure**.



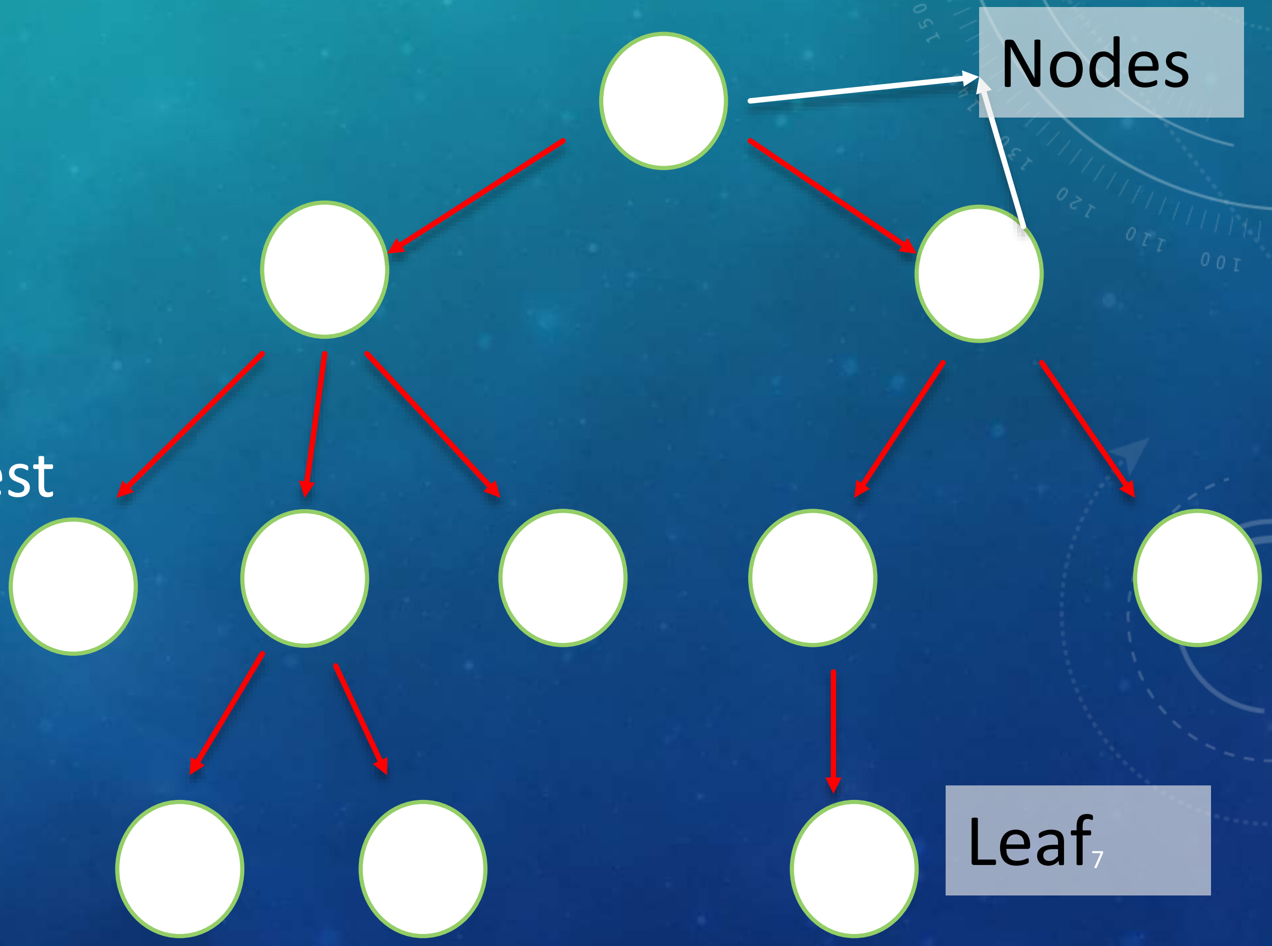
RELATION OF TREE

- **Root-**The top most Node.
- **Children**
- **Parents**
- **Siblings**- Have same parents.
- **Leaf**- Has no Child.



EDGES, DEPTH, HEIGHT

- **Edges:** If a tree have N nodes
It have N-1 edges.
- **Depth of x:** Length of path from
Root to x.
- **Hight of x:** No. Of edges in longest
Path from x to a leaf

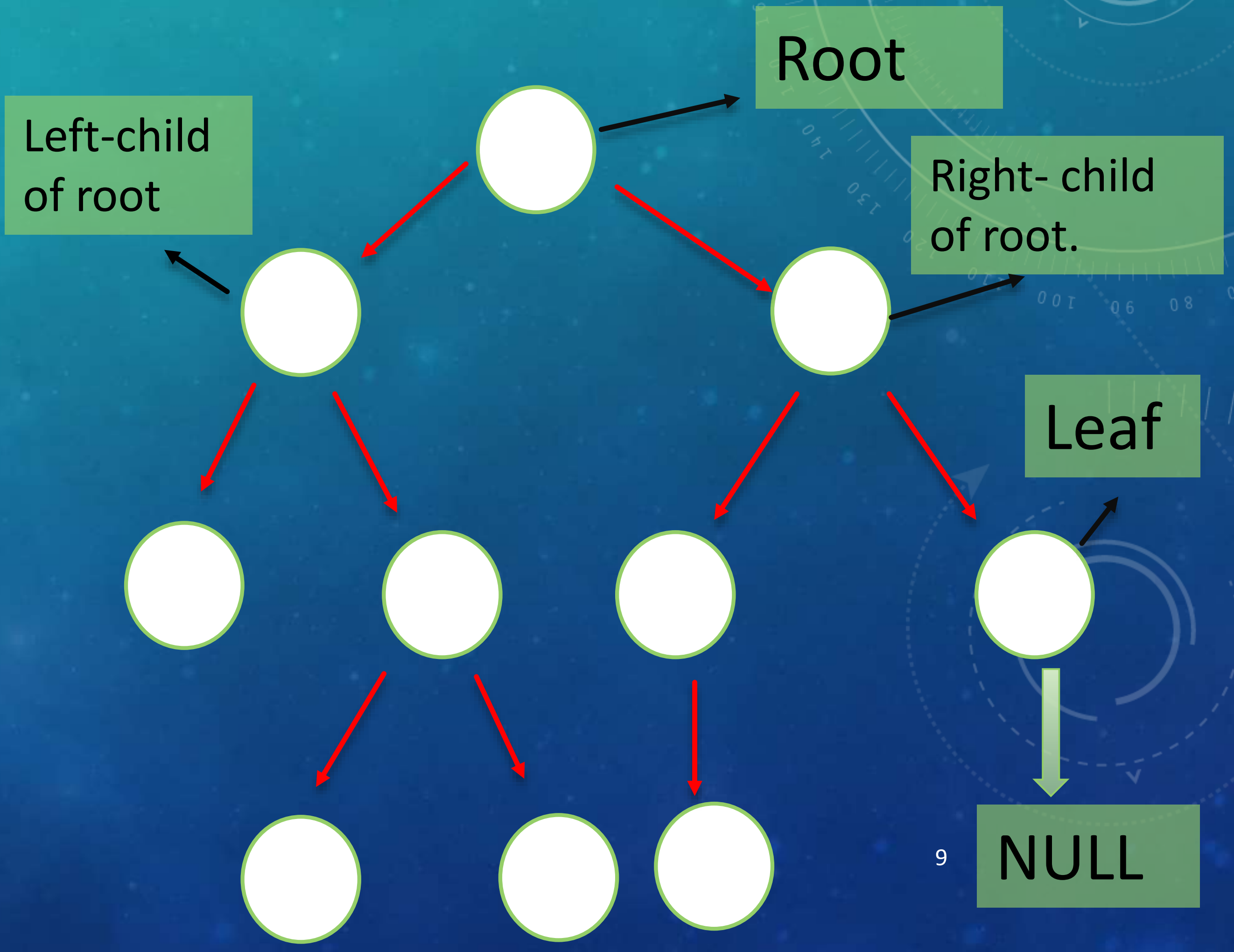


SOME APPLICATION OF TREE IN COMPUTER SCIENCE

1. Storing naturally hierarchicl data- File system.
2. Organige data for quick search, insertion, deletion- Binary search tree.
3. Dictionary
4. Network Routing Algorithm.

BINARY TREE

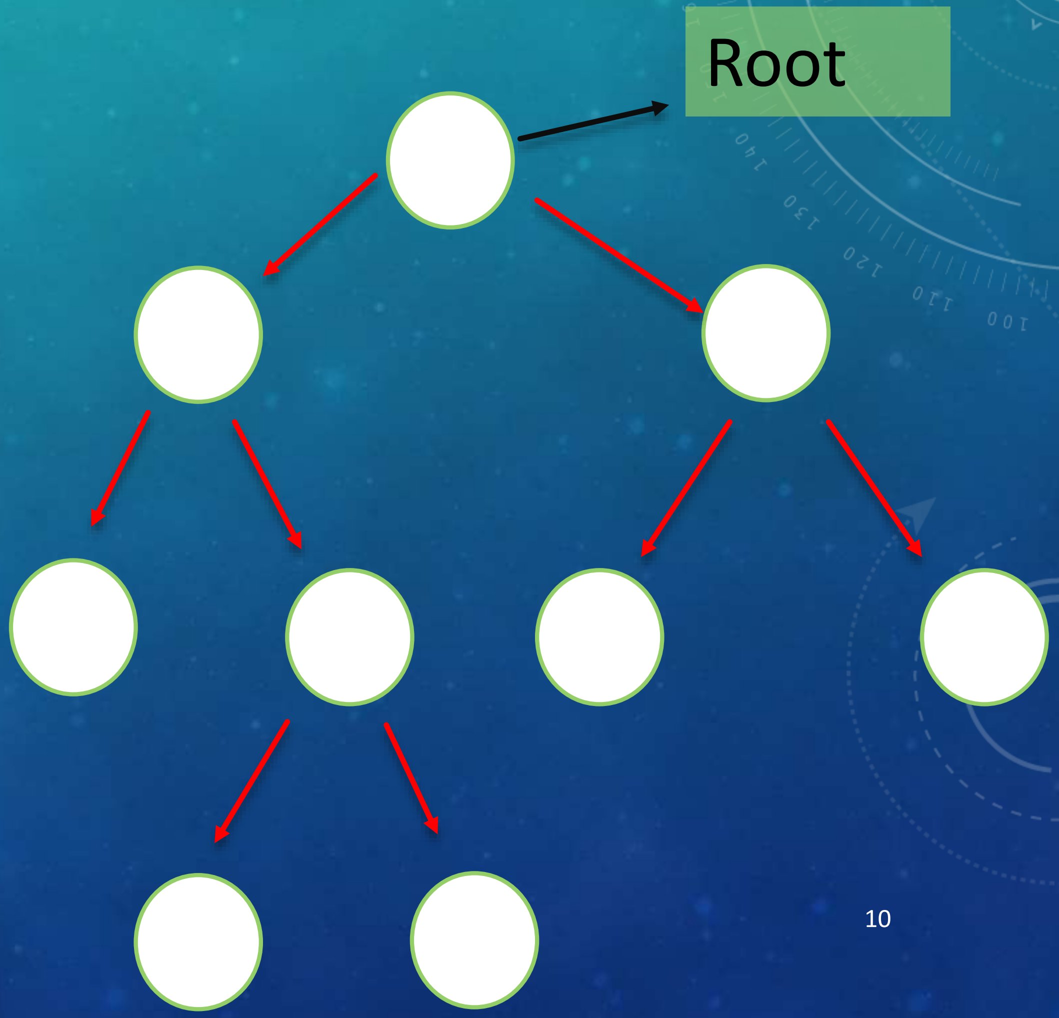
- Each node can have at most 2 childern.
- A node have only left and right child or
- Only left child or
- Only right child.
- A leaf node has no left or right child.
- A leaf node has only NULL.



STRICT/PROPER BINARY TREE

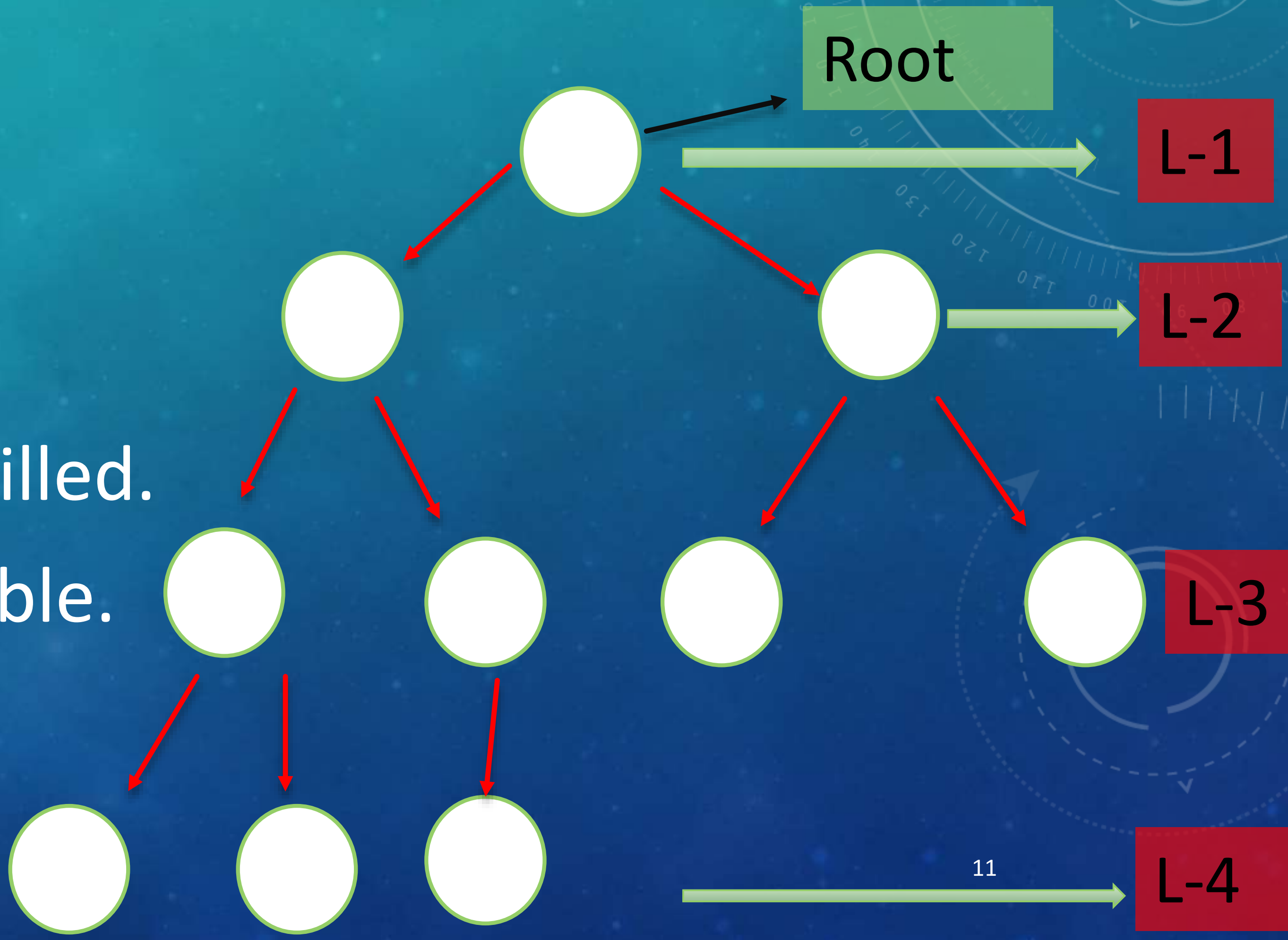
Each node can have either

2 or 0 child



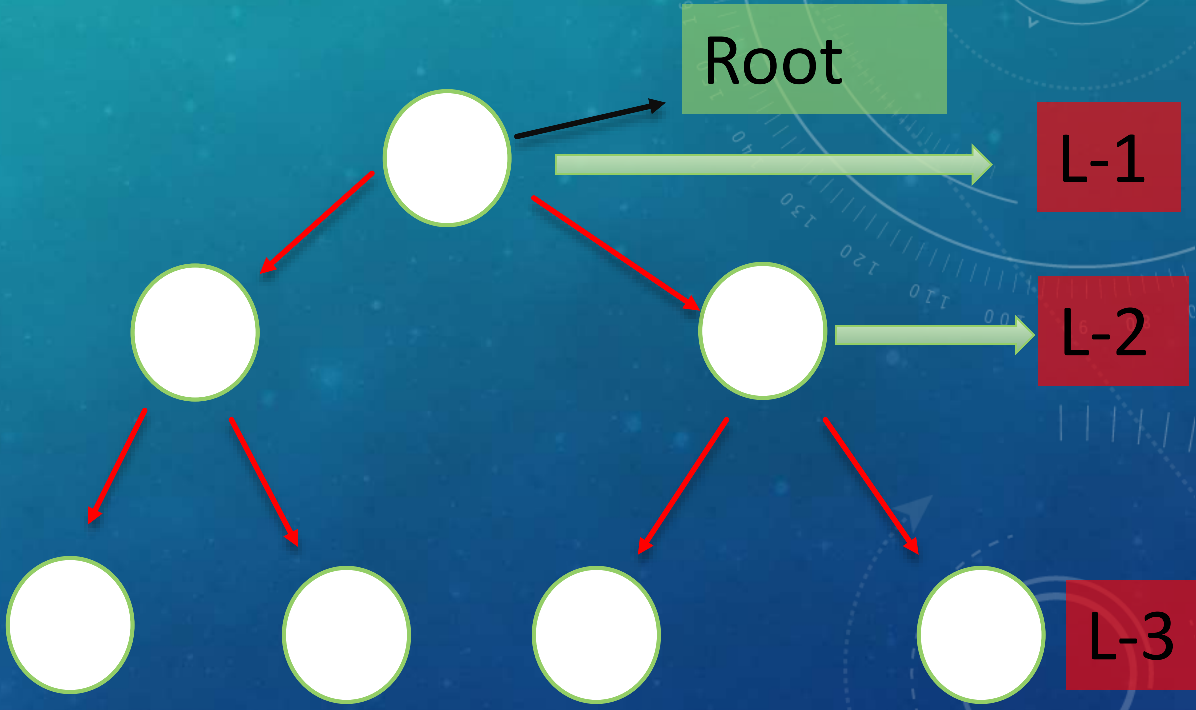
COMPLETE BINARY TREE

- The last level is completely filled.
- All nodes are as left as possible.



PARFECT BINARY TREE

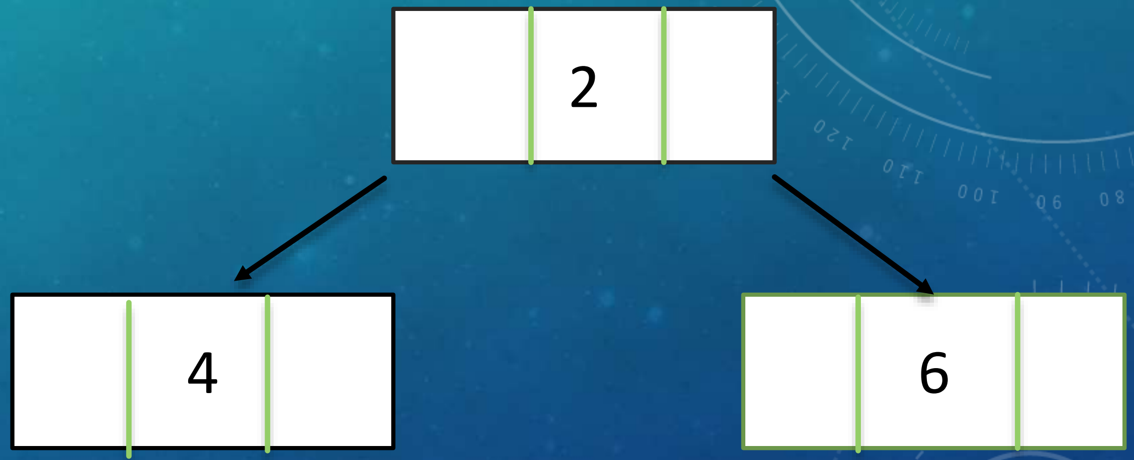
All the levels are comletely filled.



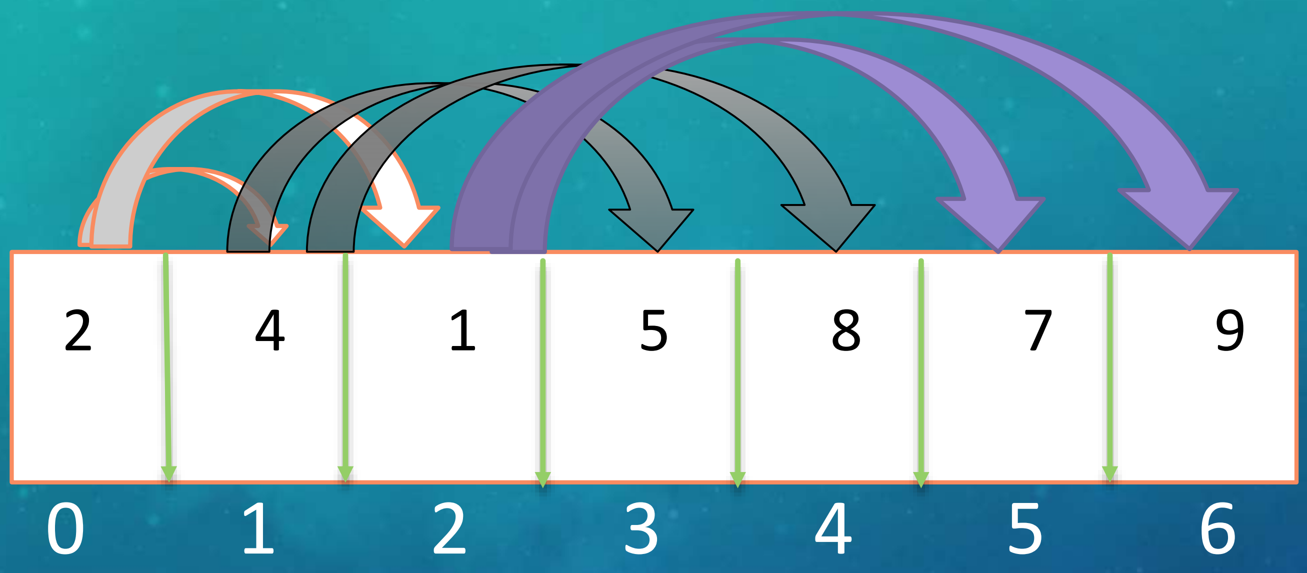
WE CAN IMPLEMENT BINARY TREE USING

A) Dynamically created nodes.

```
struct node
{
    int data;
    struct node* left;
    struct node* right
}
```



OR



B) Arrays: It only works “complete Binary tree”.

For node at index i;

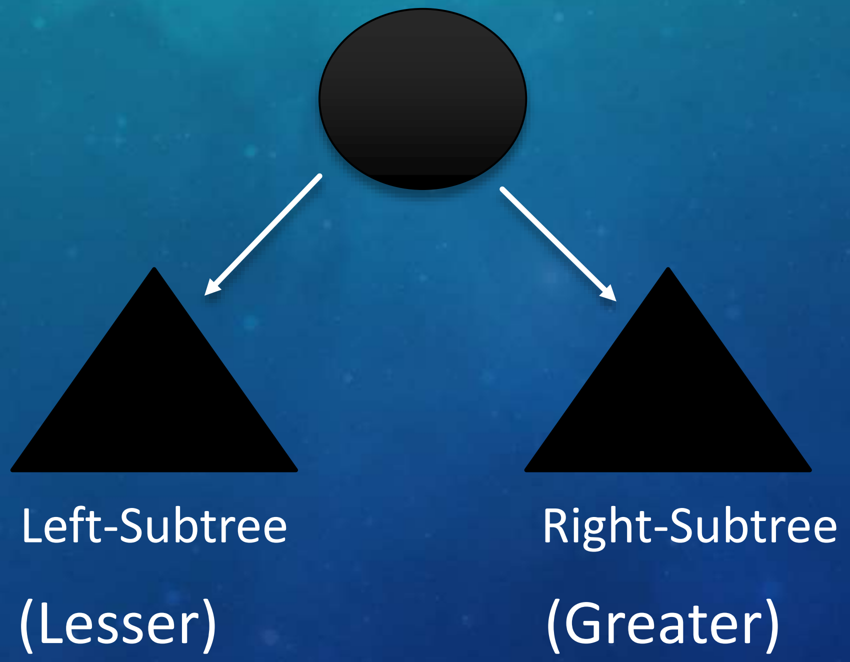
Left-child-index=2i+1

Right-child-index=2i+2

IMPLEMENT OF BINARY SEARCH TREE

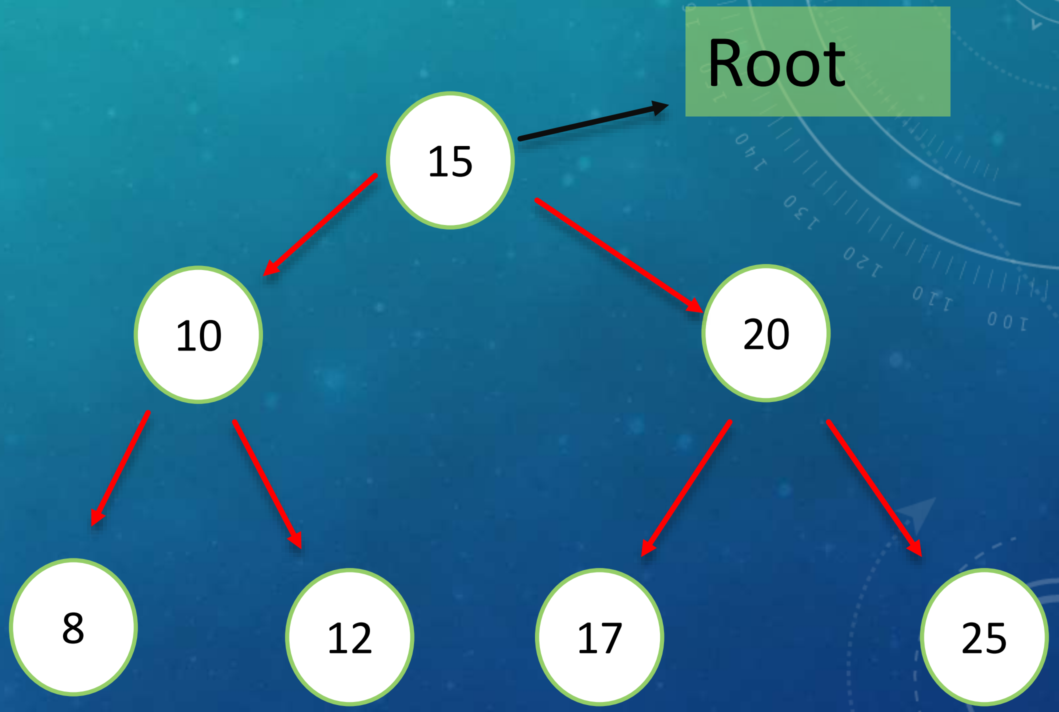
Value of all the nodes in left subtree is Lesser or Equal.

Value of all the nodes in right subtree is greater.



EXAMPLE

- 15>10-Left
- 15<20-Right
- 10>8-Left
- 10<12-Right
- 20>17-Left
- 20<25-Right



BINARY TREE TRAVERSAL

Tree traversal

- Breadth-first or

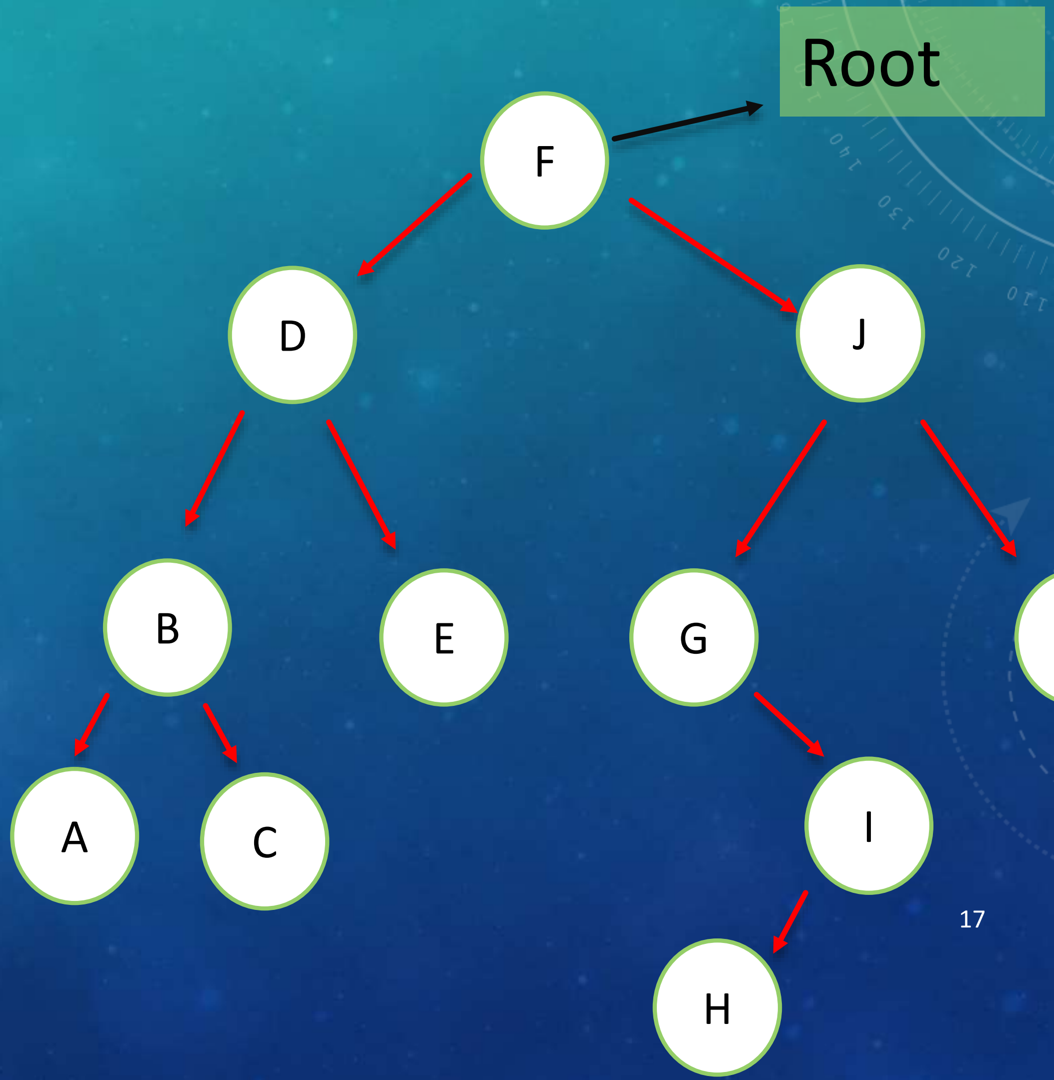
  Lever-order

  F,D,J,B,E,G,K,A,C,I,H

- Depth-first

  Preorder, Inorder &

  Postorder



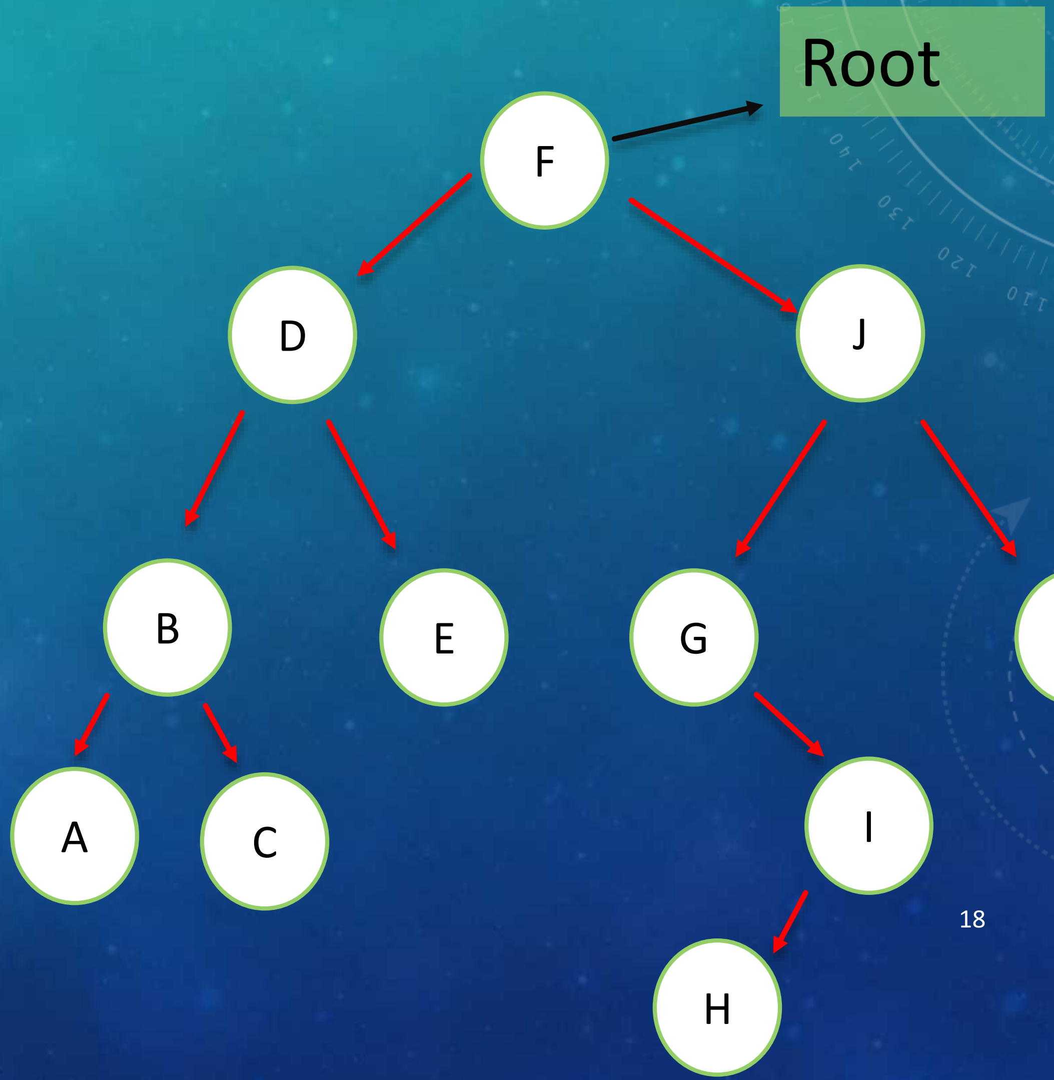
PREORDER(DLR)

Data Left Right

<root><left><right>

F,D,B,A,C,E,J,G,I,H,K

```
Void Postorder(struct bstnode* root)
{
    if(root==NULL)
    Postorder(root->right);
    Postordrt(root->right);
    printf(“%c”,root->data);
}
```



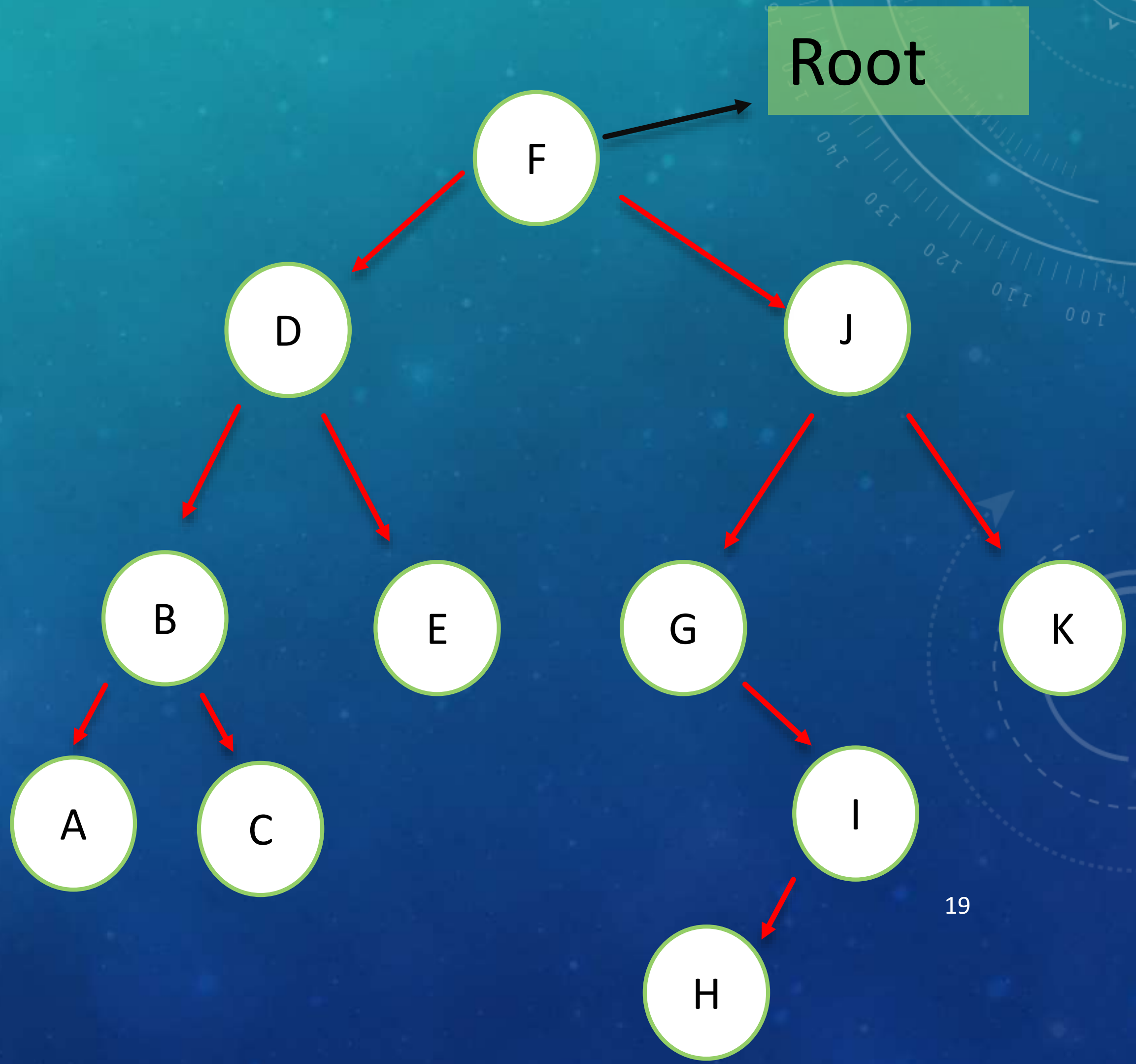
INORDER(LDR)

Left Data Right

<left><root><right>

A,B,C,D,E,F,G,H,I,J,K

```
Void Inorder(struct bstnode* root)
{
        if(root==NULL) return;
        Inorder(root->left);
        printf("%c",root->data);
        Inorder(root->right);
}
```



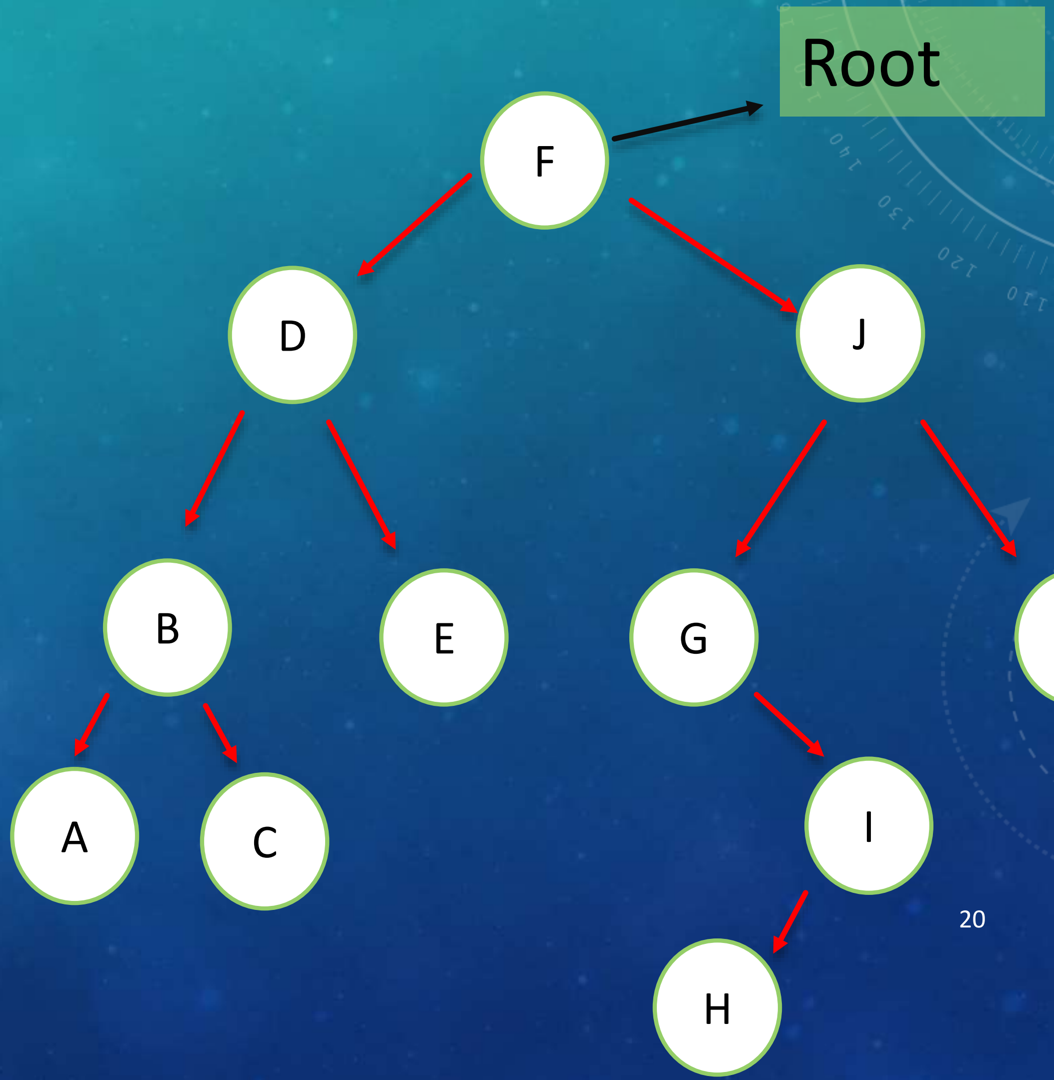
POSTORDER(LRD)

Left Right Data

<left><right><root>

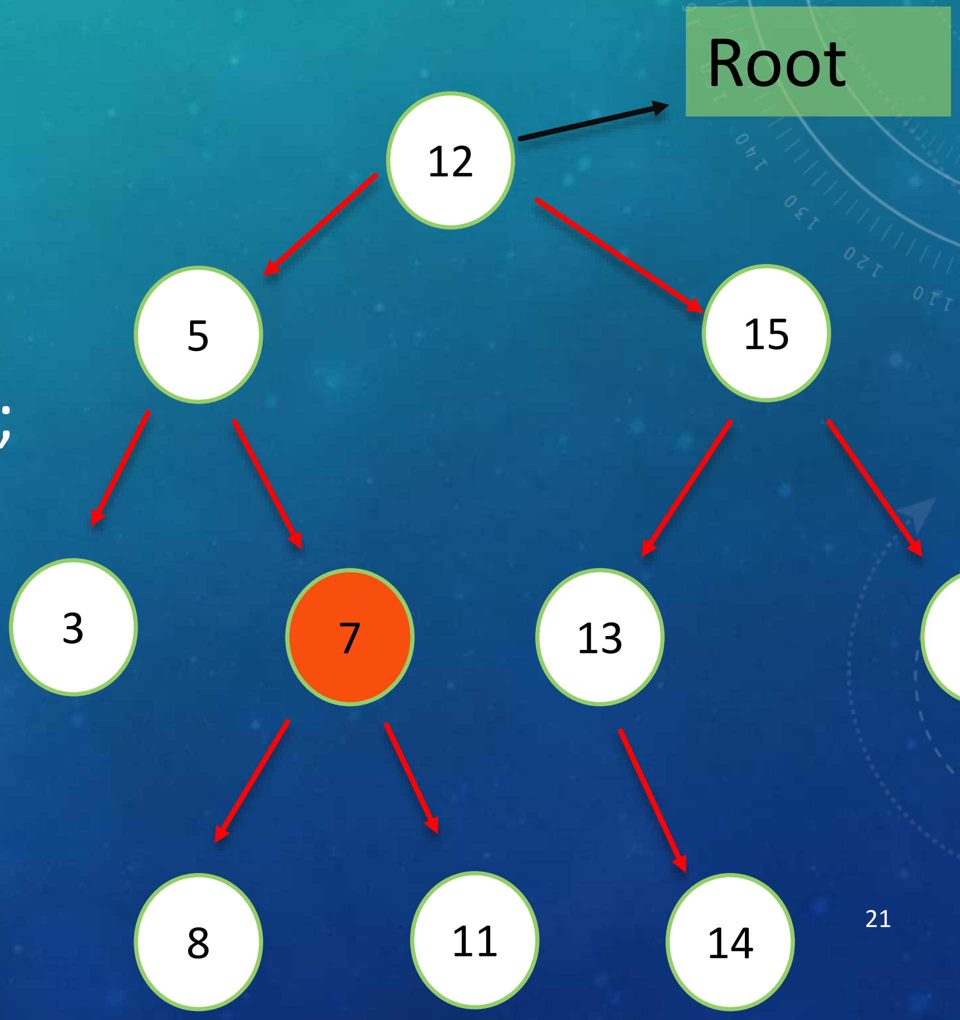
A,C,B,E,D,H,I,G,K,J,F,A,B

```
Void Postorder(struct bstnode* root)
{
    if(root==NULL)
    Postorder(root->right);
    Postordrt(root->right);
    printf("%c",root->data);
}
```



SEARCH AN ELEMENT IN BST

```
bool Search( bstnode* root, data type)
{
     if (root==NULL) return false;
     else if(root->data == data) return true;
     else if(root->data <= data)
     return Search(root->left, data);
     else
          return Search(root->right, data);
}
```



Question ?