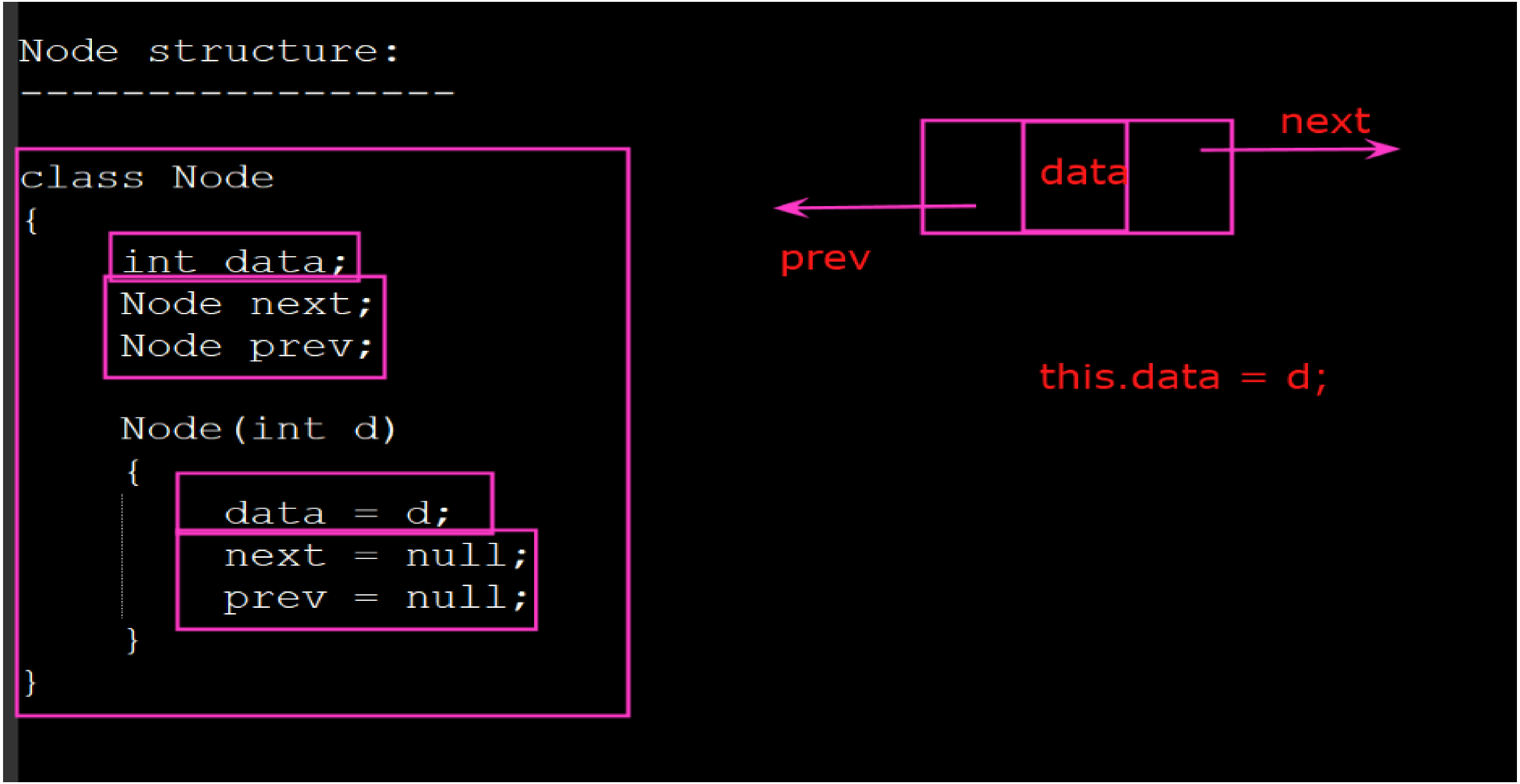
Algorithms & Data Structure
Day 6: Tree

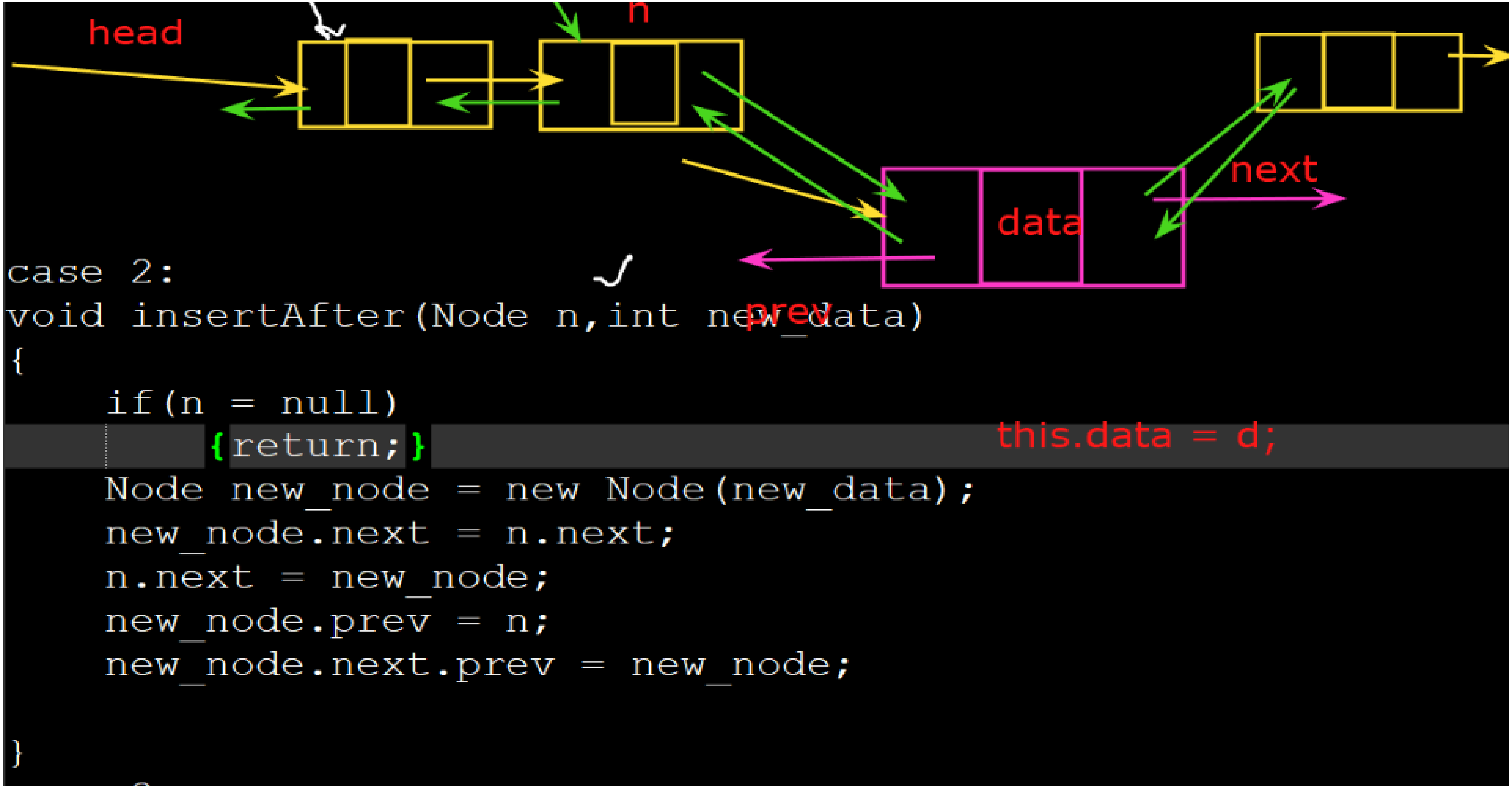
Kiran Waghmare

Imagination is more important than knowledge
- Albert Einstein



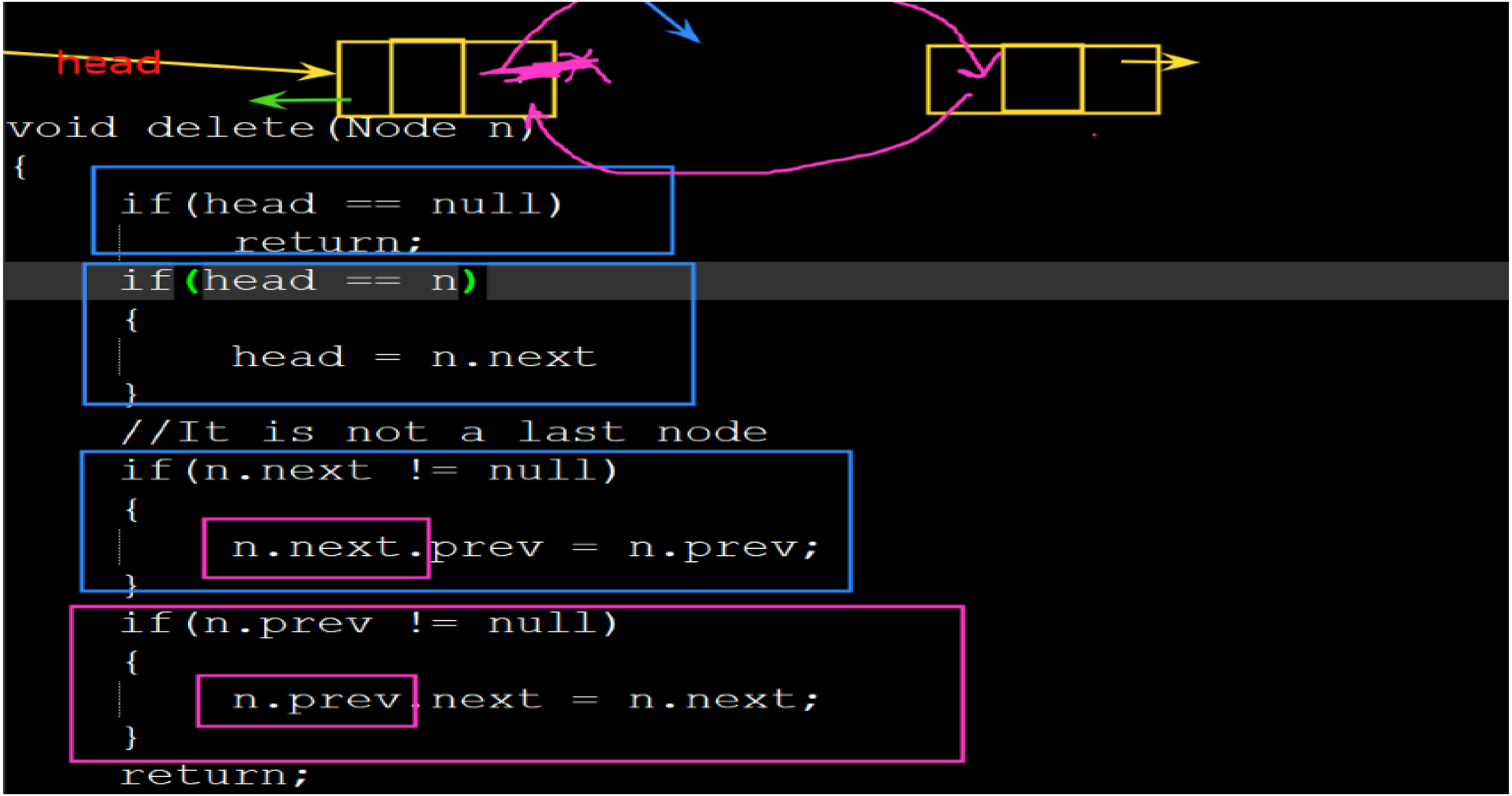
```
Node structure:
-------------------

class Node
{
      int data;
      Node next;
      Node prev;

      Node(int d)
      {
          data = d;
          next = null;
          prev = null;
      }
}
```

this.data = d;

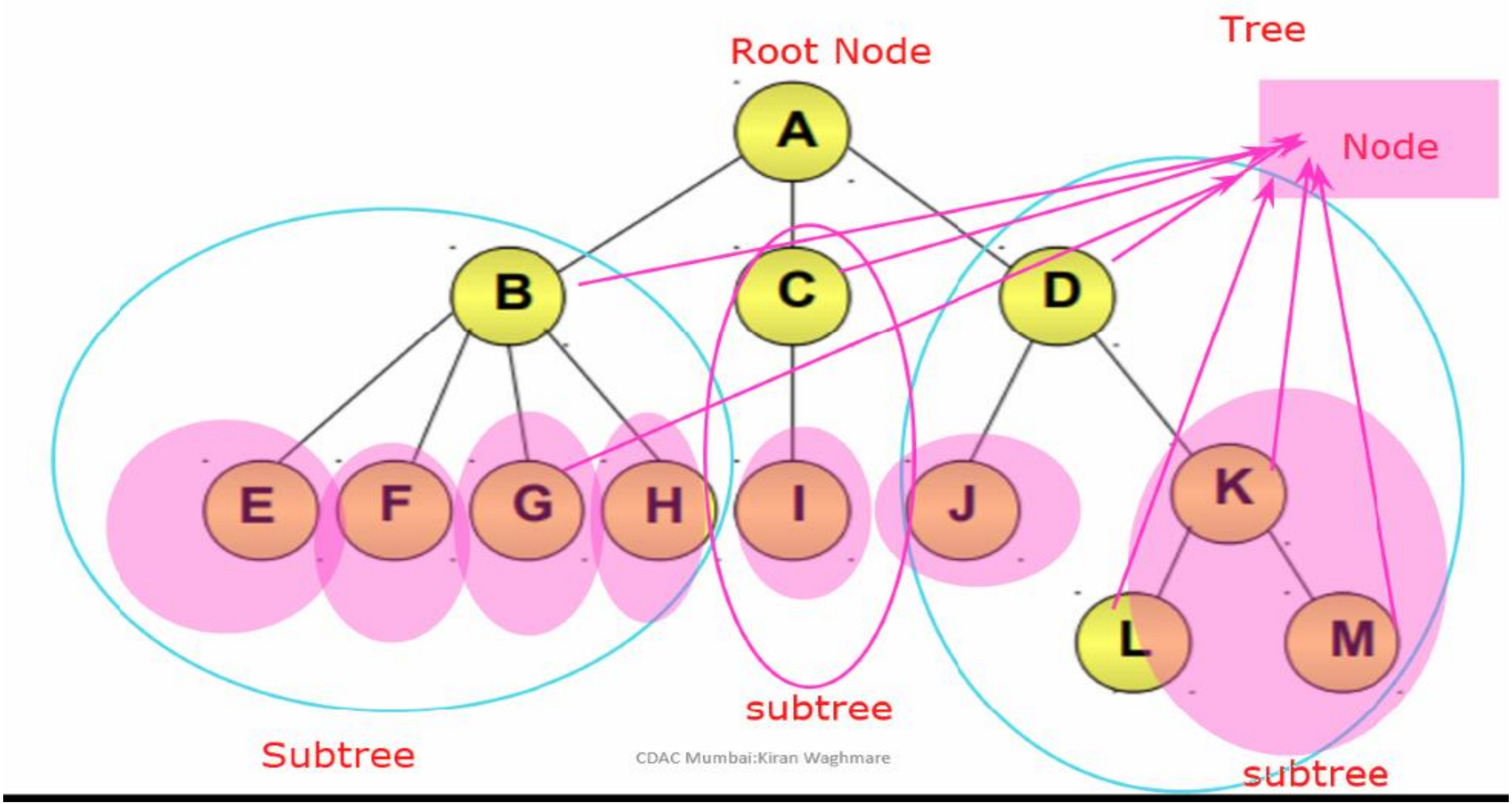


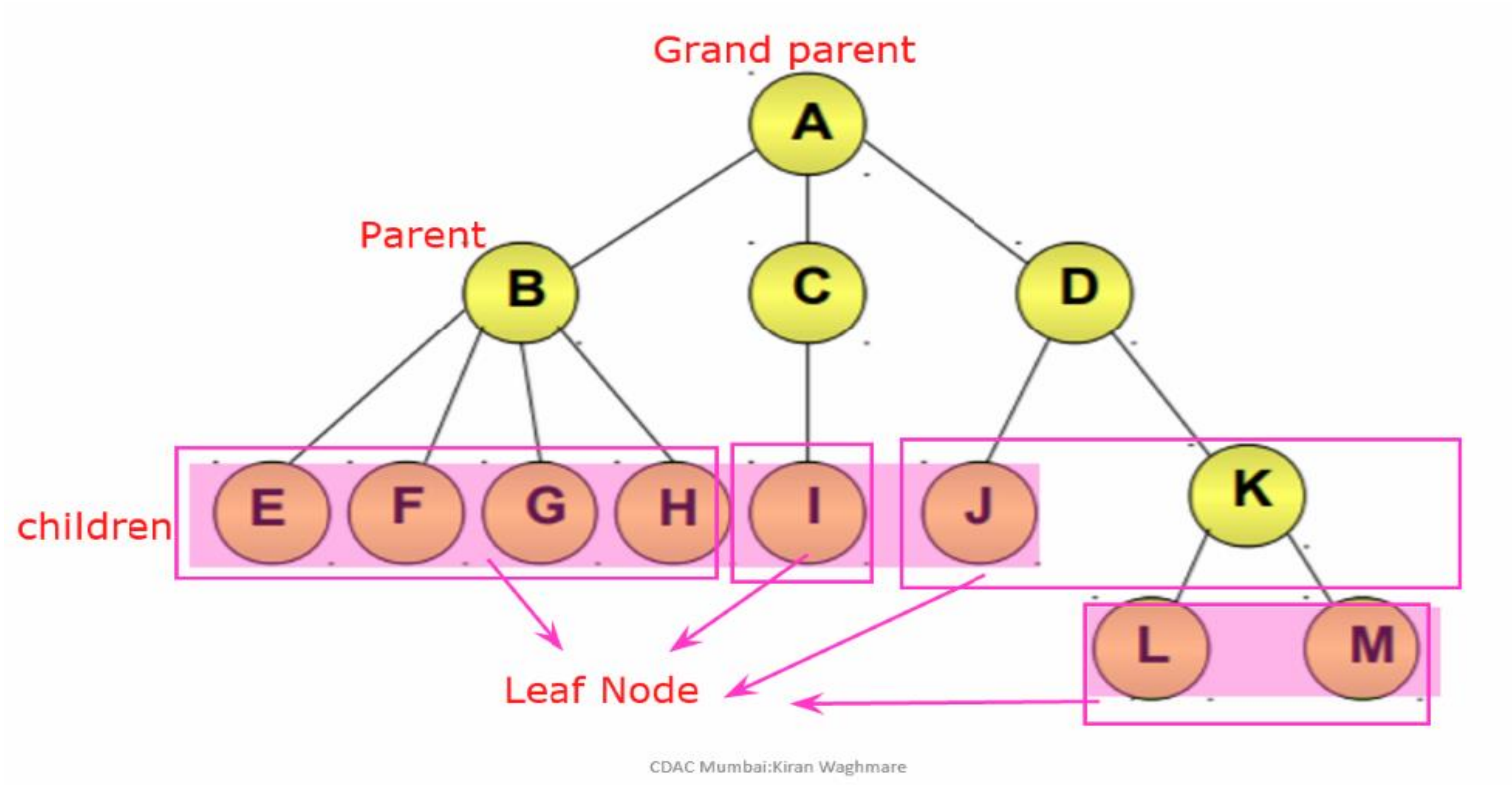
```
case 2:
void insertAfter(Node n,int new_data)
{
    if(n = null)
        {return;}
    Node new_node = new Node(new_data);
    new_node.next = n.next;
    n.next = new_node;
    new_node.prev = n;
    new_node.next.prev = new_node;

}
```

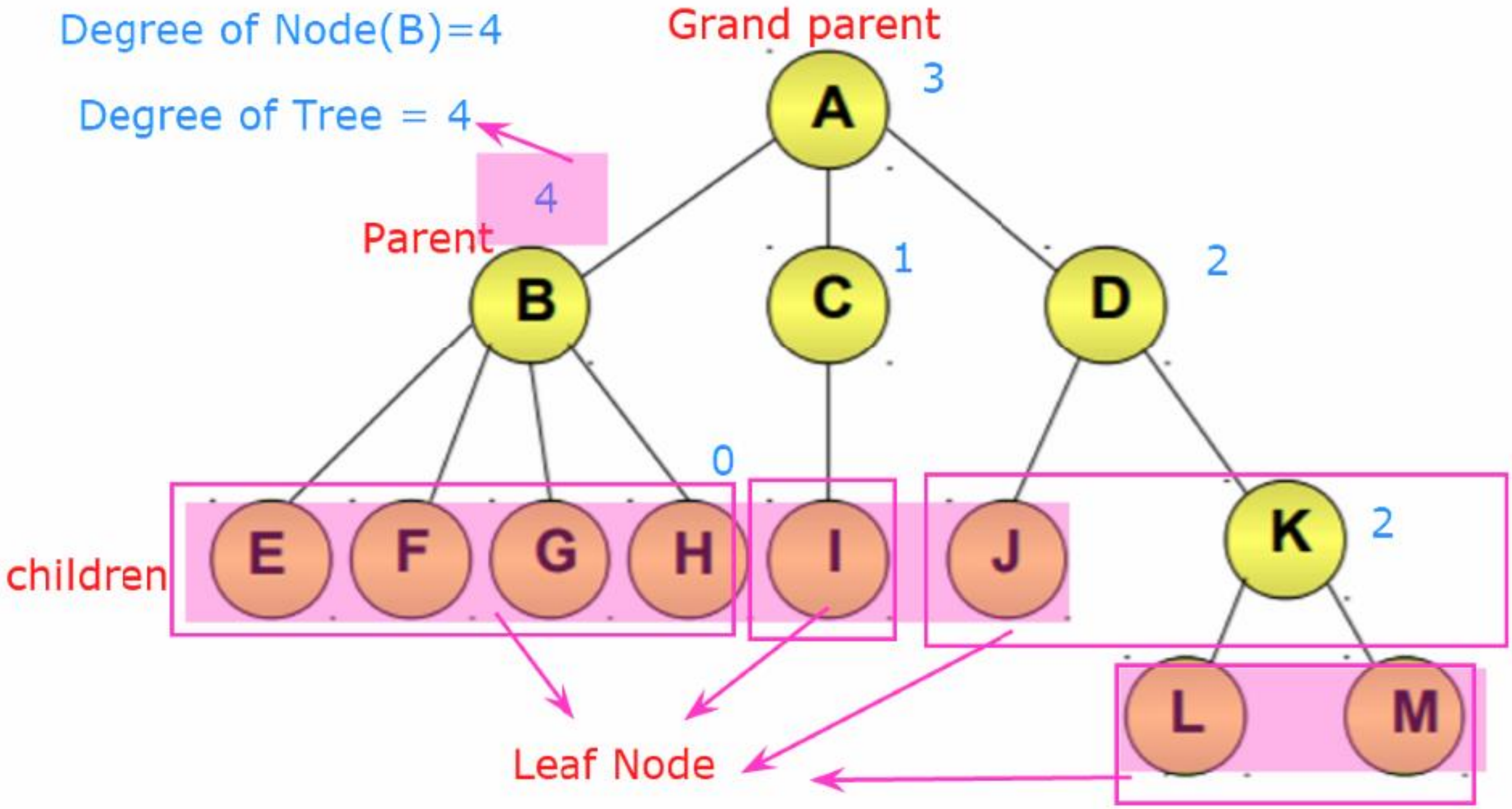
this.data = d;



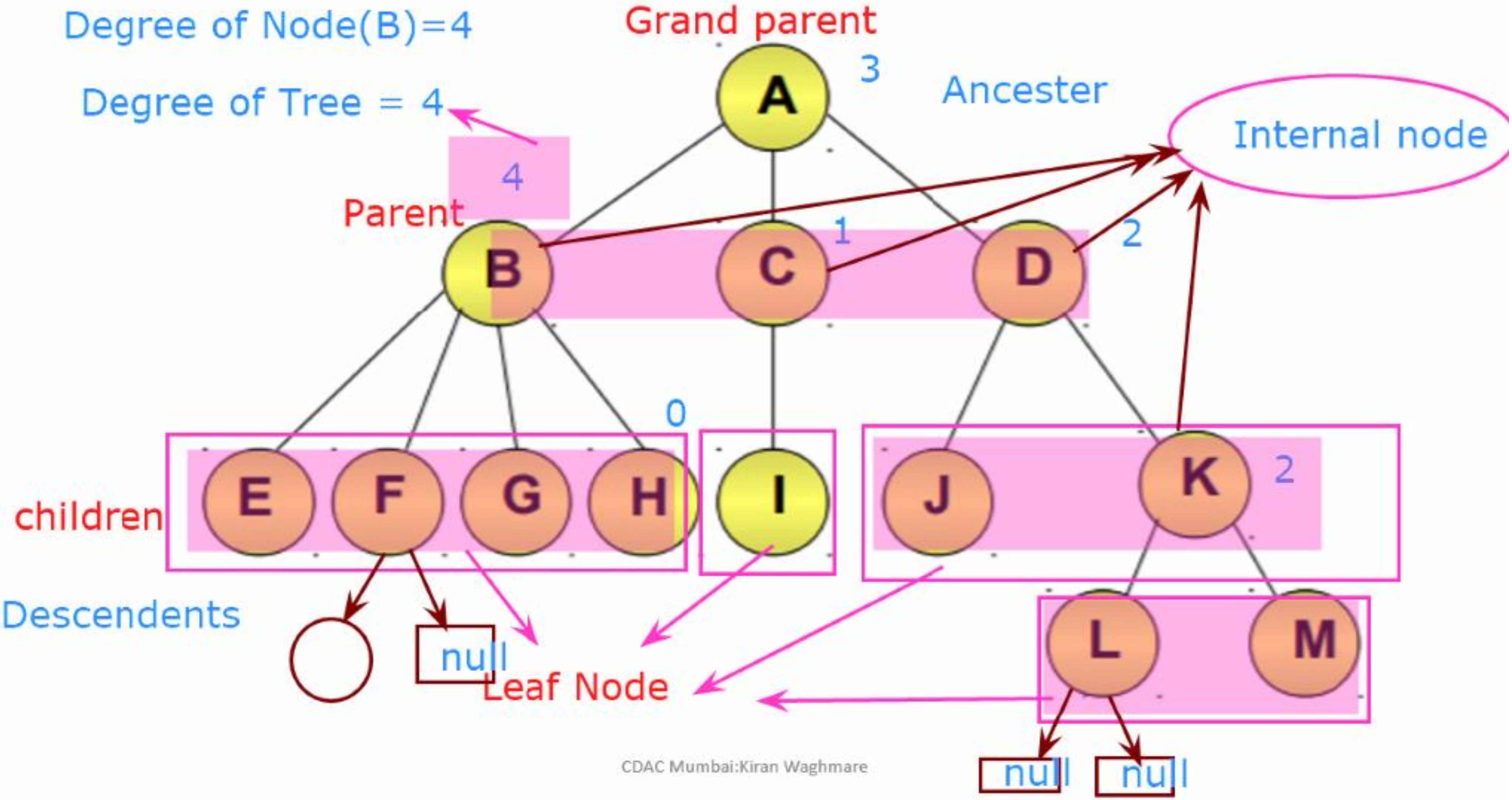
```
void delete(Node n)
{
    if(head == null)
        return;
    if(head == n)
    {
        head = n.next
    }
    //It is not a last node
    if(n.next != null)
    {
        n.next.prev = n.prev;
    }
    if(n.prev != null)
    {
        n.prev.next = n.next;
    }
    return;
```

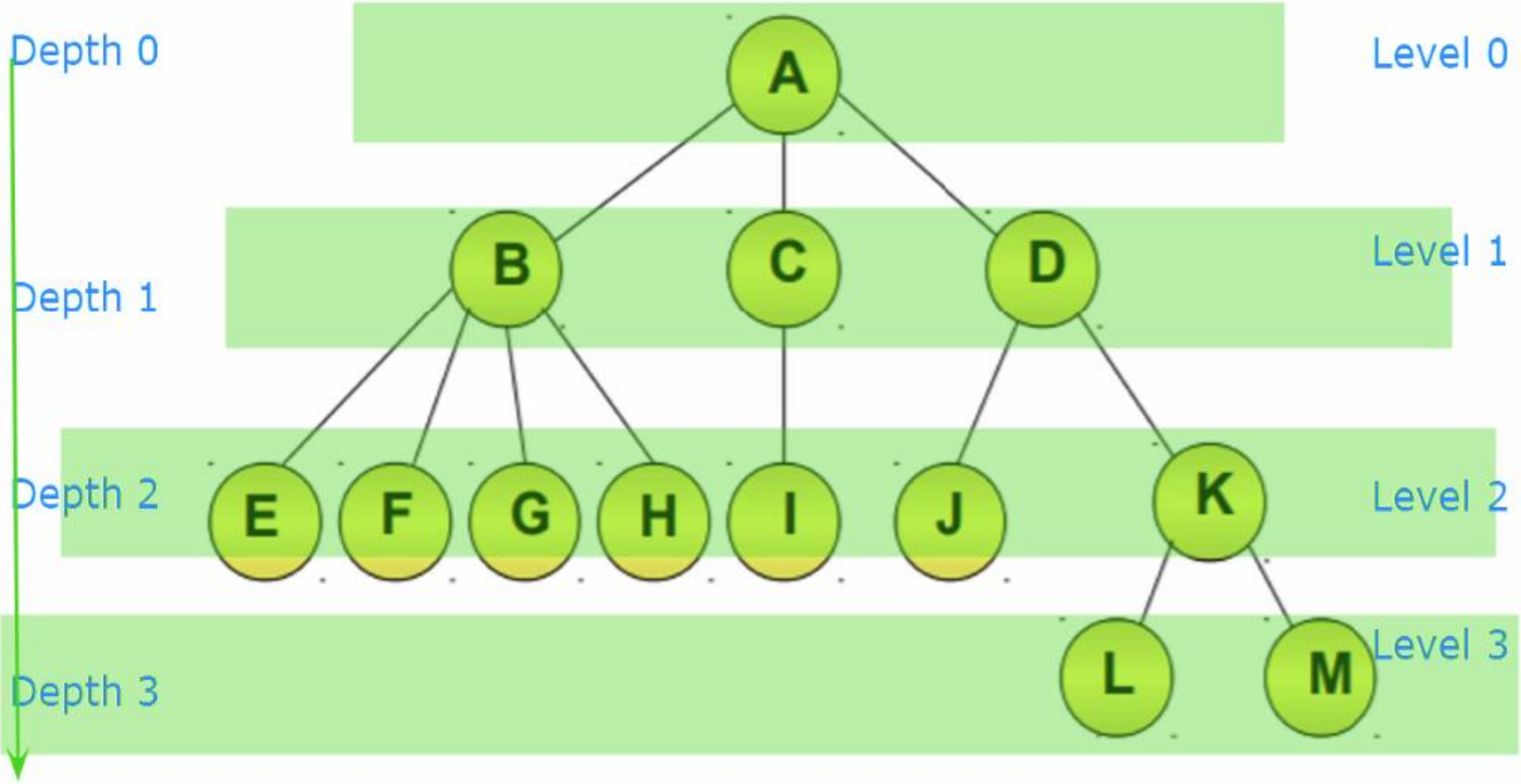

Tree
Root Node
Node
A
B
C
D
E
F
G
H
I
J
K
L
M
subtree
Subtree
subtree
CDAC Mumbai:Kiran Waghmare

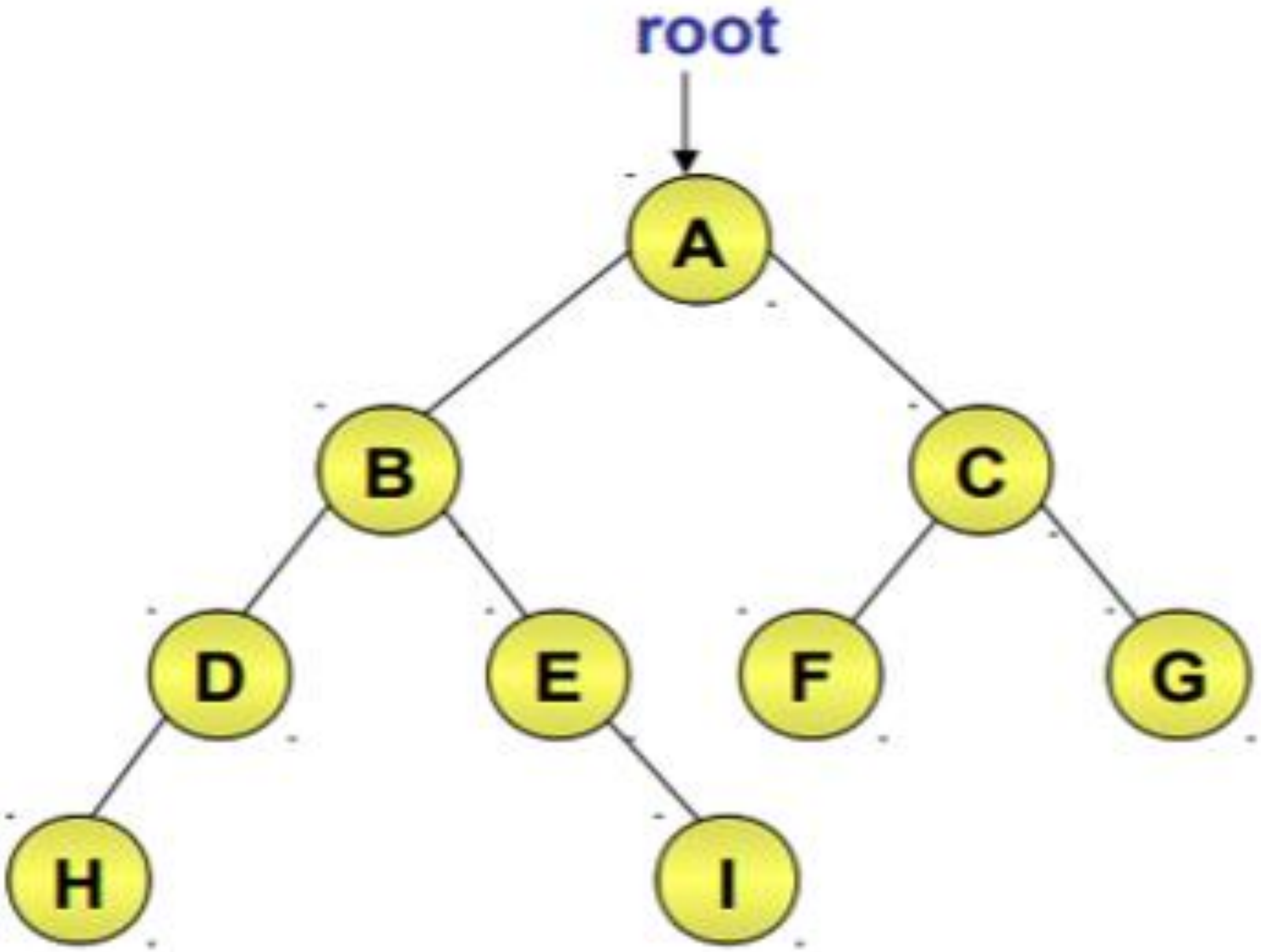

Grand parent
A
Parent
B
C
D
children
E
F
G
H
I
J
K
L
M
Leaf Node
CDAC Mumbai:Kiran Waghmare



CDAC Mumbai:Kiran Waghmare


Degree of Node(B)=4
Degree of Tree = 4
Grand parent
A
3
Ancester
Internal node
4
Parent
B
C
1
D
2
0
children
E
F
G
H
I
J
K
2
L
M
Descendents
null
Leaf Node
null
null


Depth 0
Depth 1
Depth 2
Depth 3
A
B
C
D
E
F
G
H
I
J
K
L
M
Level 0
Level 1
Level 2
Level 3

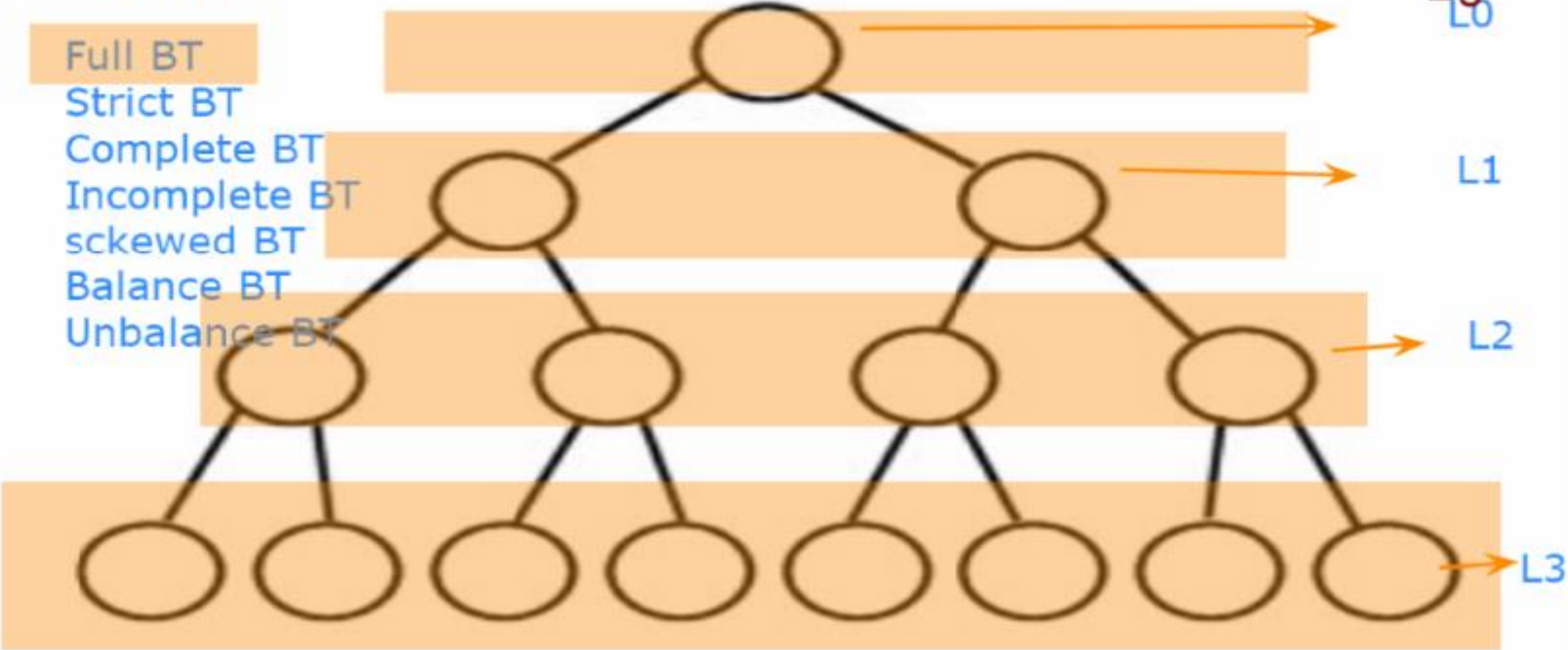

root
A
B
C
D
E
F
G
H
I

The max no. o f nodes on level i of a binary tree is 2^i, i>=0

(0,1,2)

2^i
=2^3
=8

Full BT
Strict BT
Complete BT
Incomplete BT
sckewed BT
Balance BT
Unbalance BT



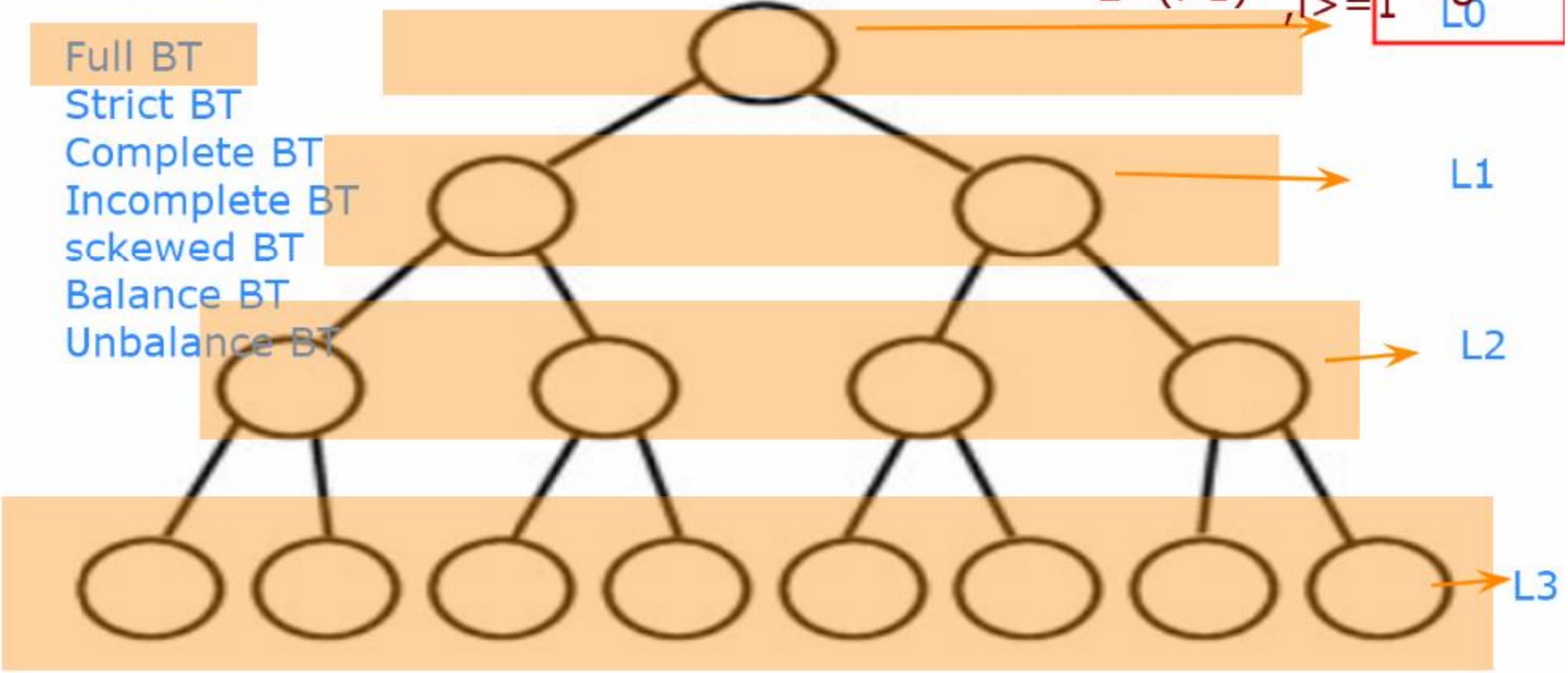
The max no. o f nodes on level i of a binary tree is 2^i, i>=~~0~~ 1

(0,1,2)

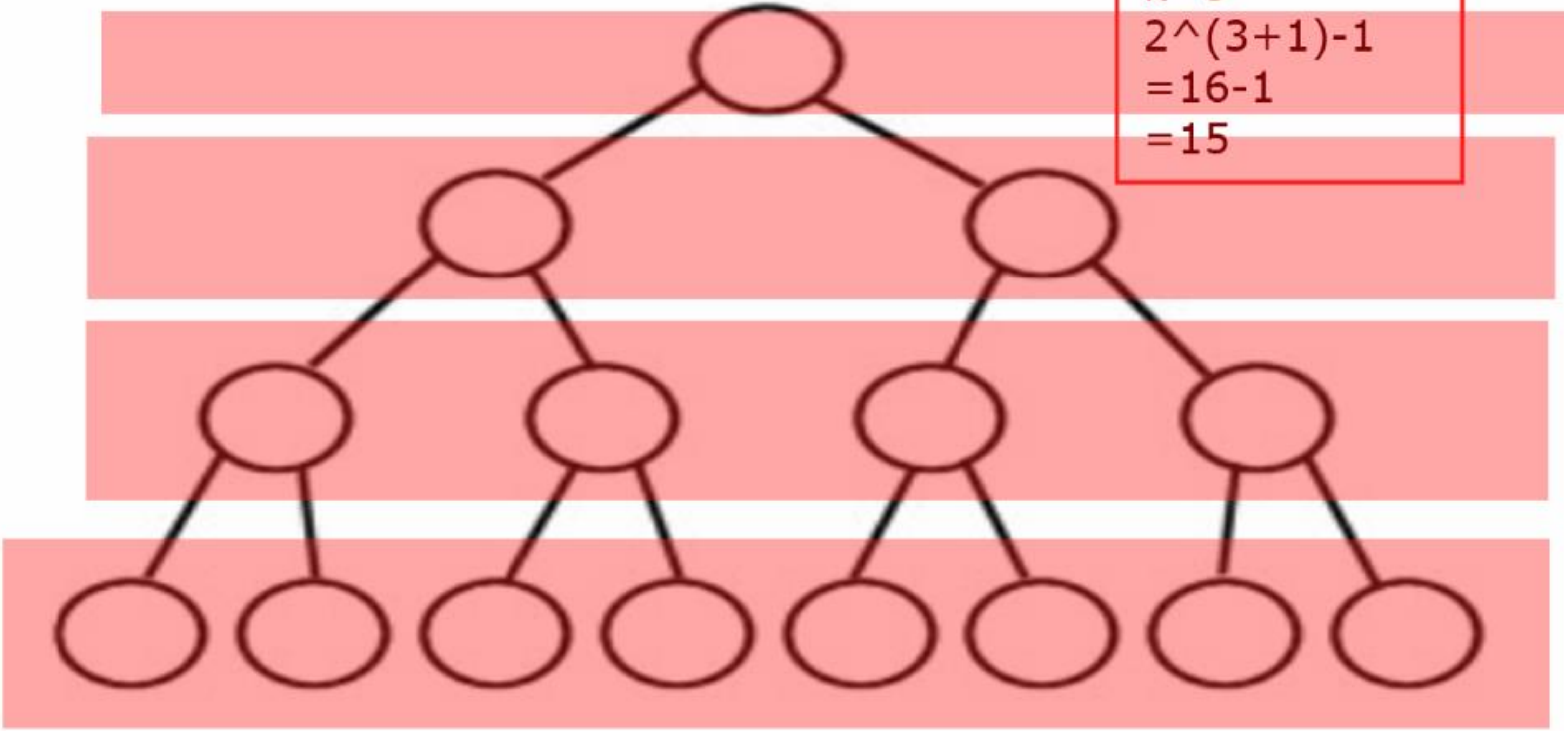
2^(i-1) ,i>=1

2^i
=2^3
=8

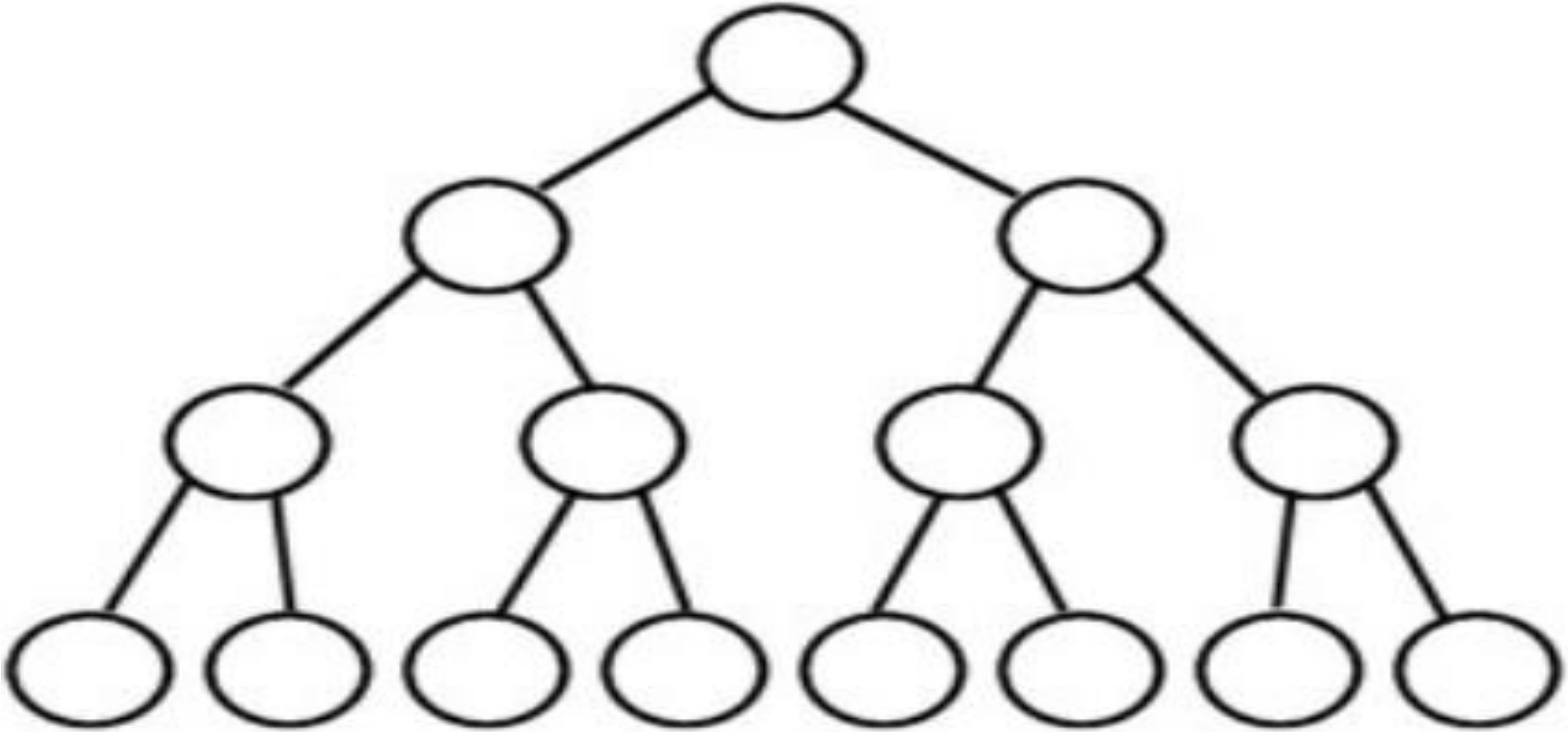
Full BT
Strict BT
Complete BT
Incomplete BT
sckewed BT
Balance BT
Unbalance BT



The max no nodes in a BT of depth k =>(2^(k+1))-1, k>=~~0~~ 1

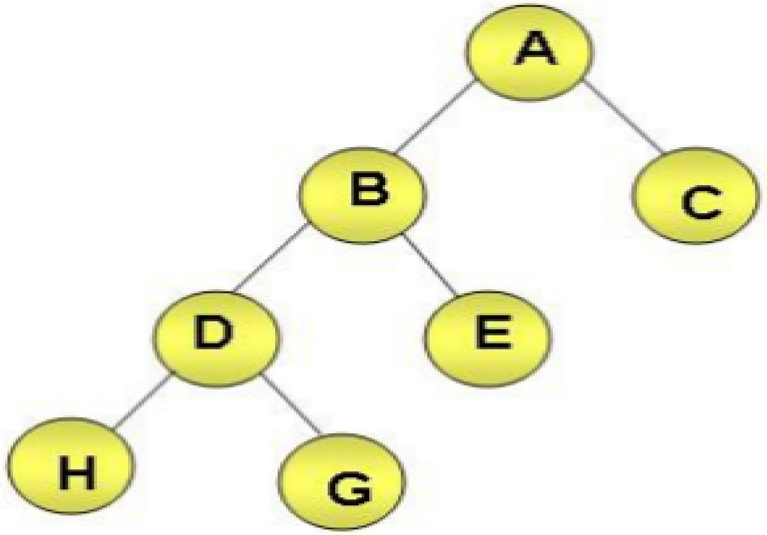


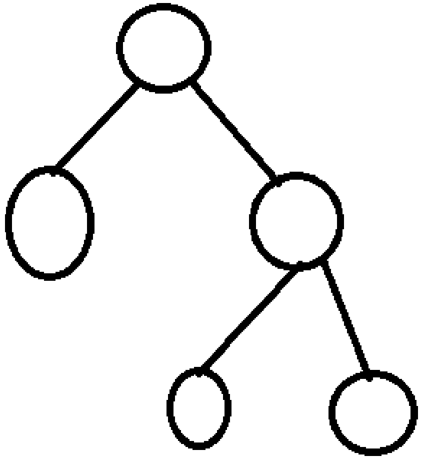
k=3
2^(3+1)-1
=16-1
=15



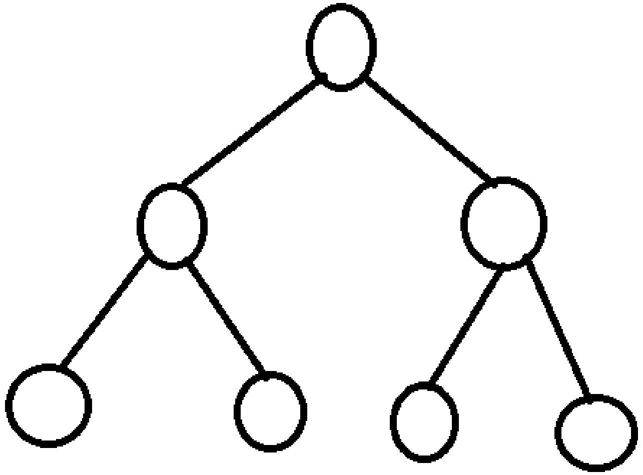
Defining Binary Trees

- Binary tree is a specific type of tree in which each node can have at most two children namely left child and right child.
- There are various types of binary trees:
  - Strictly binary tree
  - Full binary tree
  - Complete binary tree
- Strictly binary tree:
  - A binary tree in which every node, except for the leaf nodes, has non-empty left and right children.

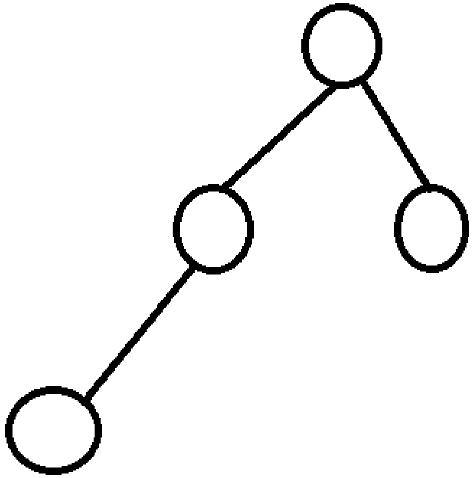




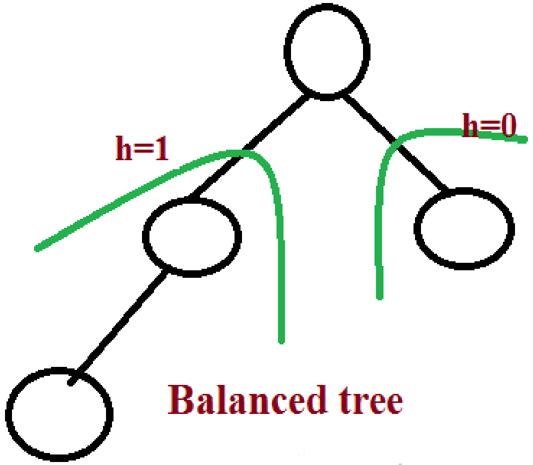
Binary Tree



Fully (perfect) Binary Tree

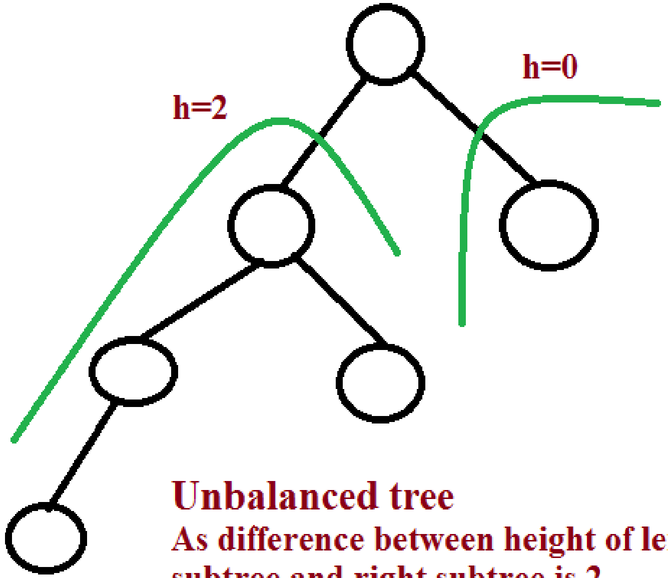


Complete Binary Tree



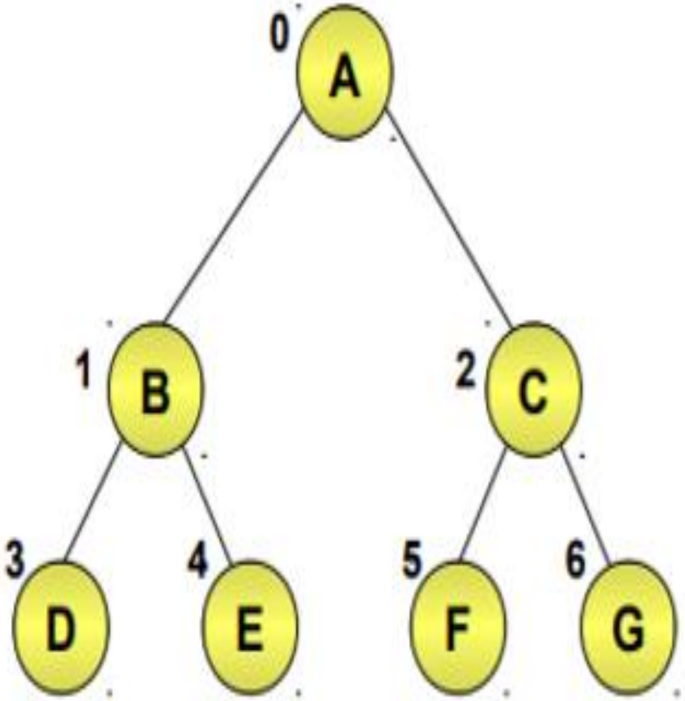
Balanced tree

As difference beween height of left subtree and right subtree is 1

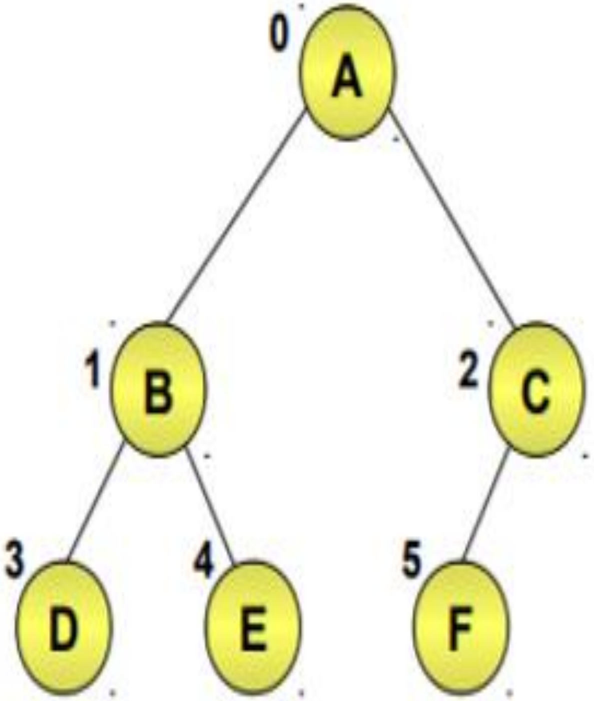


Unbalanced tree

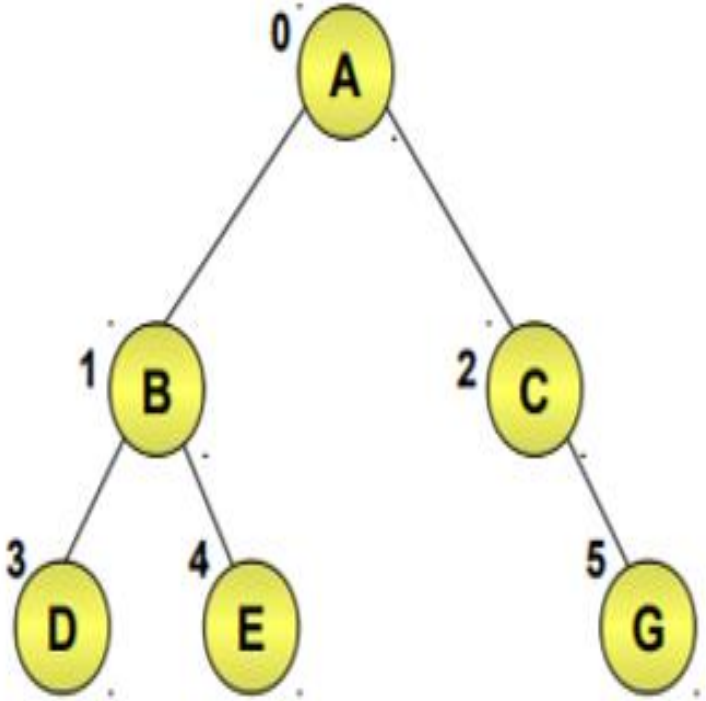
As difference between height of left subtree and right subtree is 2



Full Binary Tree



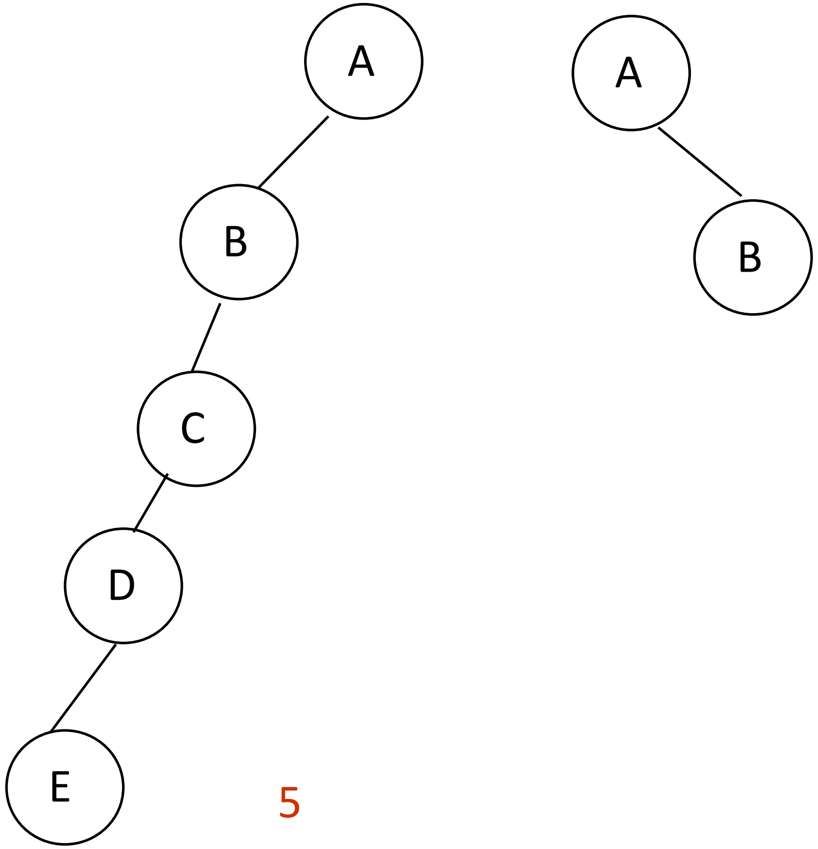
Complete Binary Tree



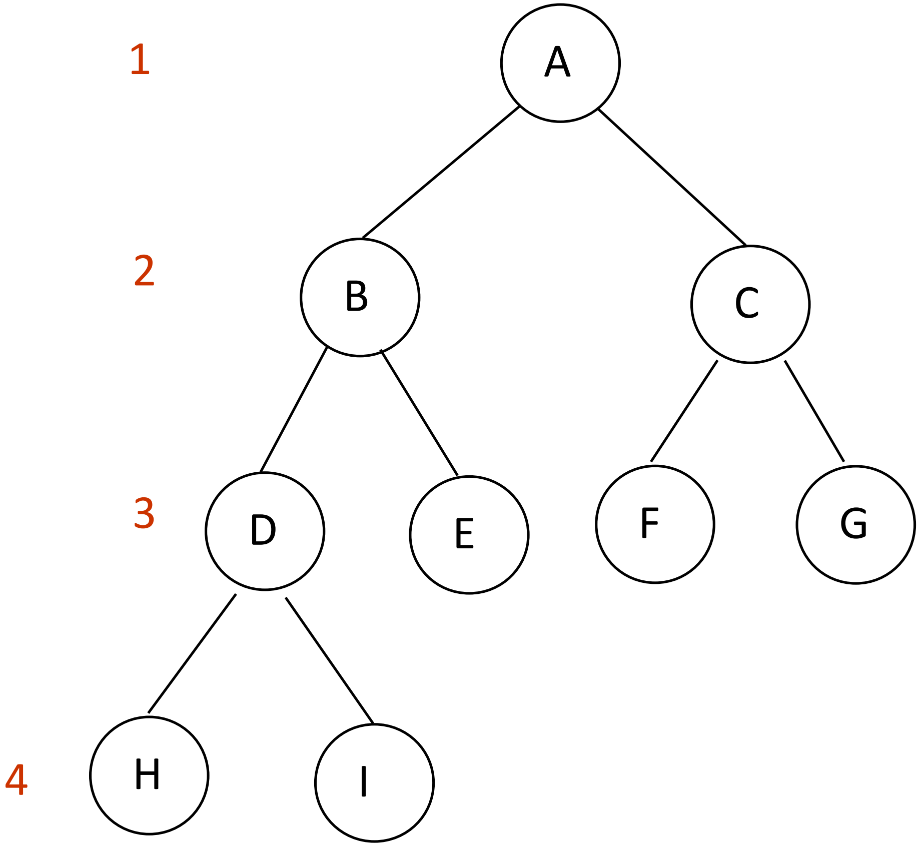
Incomplete Binary Tree

Examples of the Binary Tree

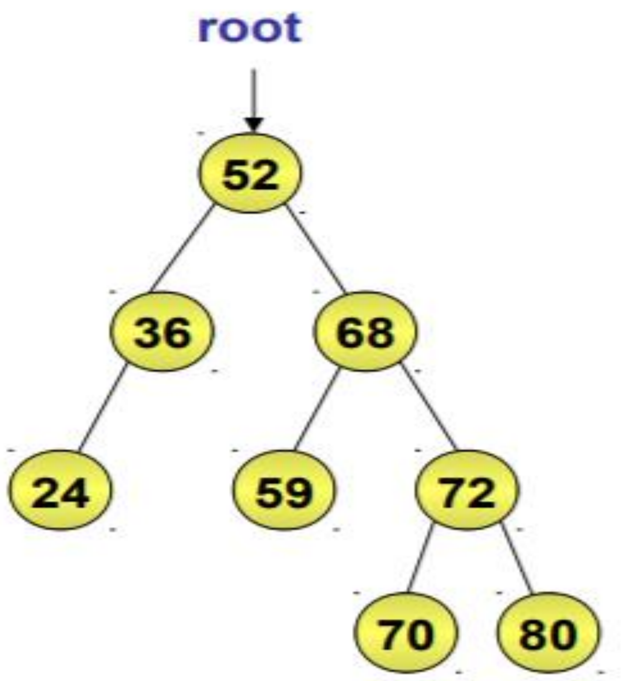
Skewed Binary Tree



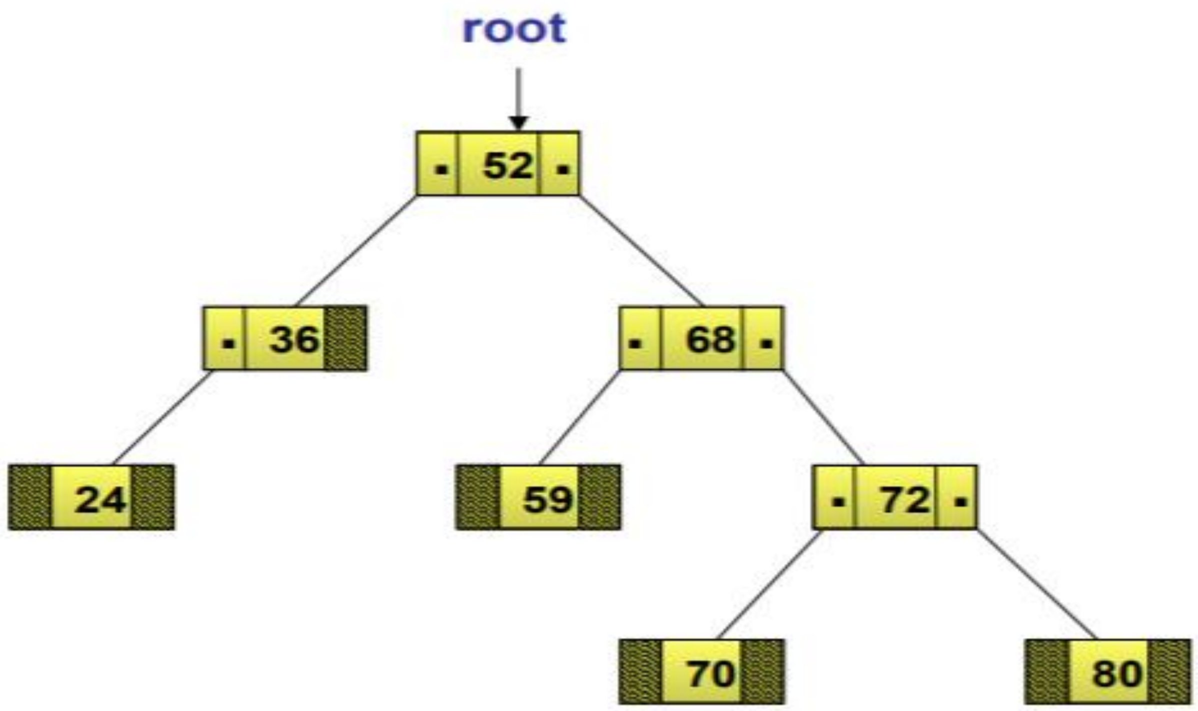
Complete Binary Tree



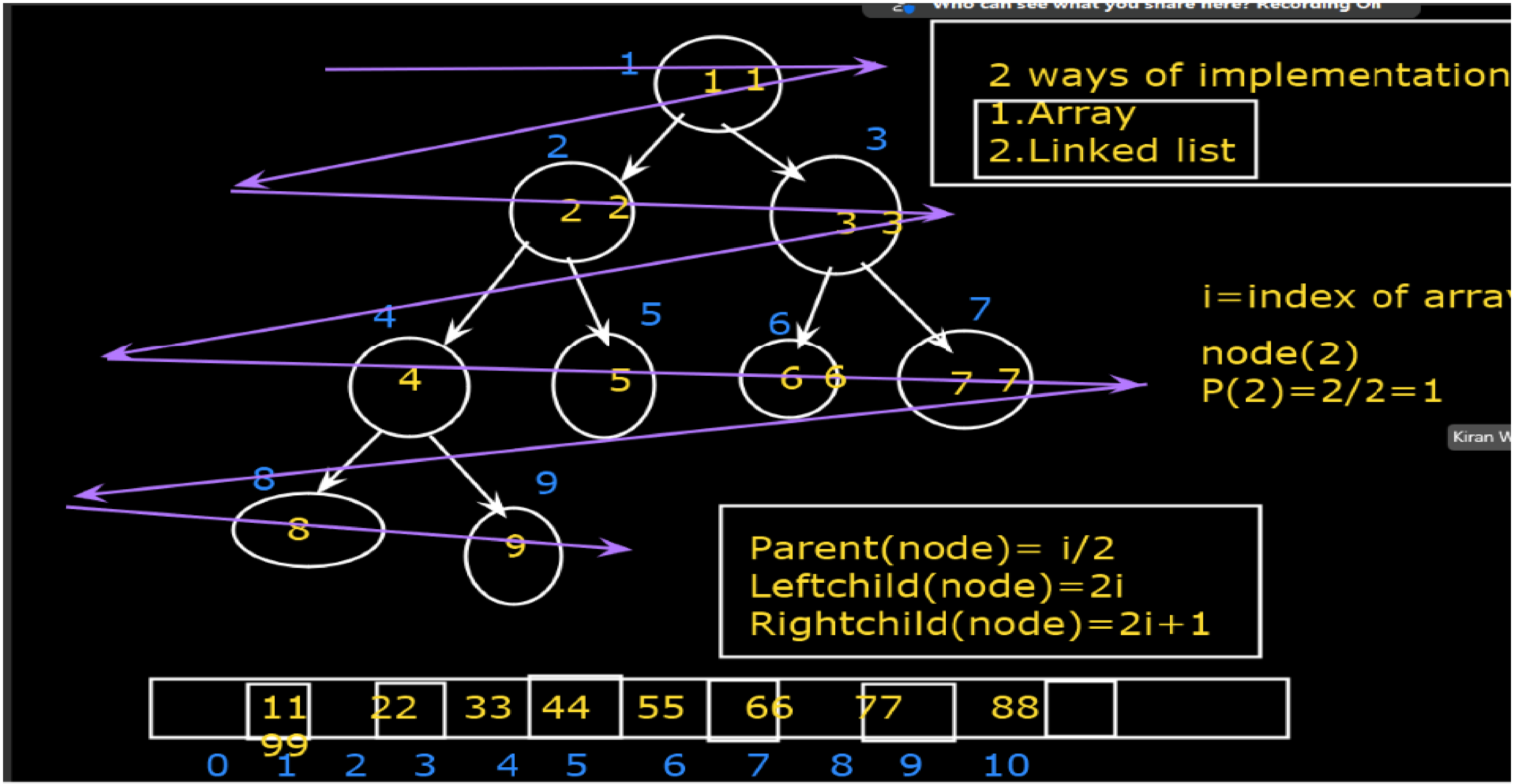
Representing a Binary Tree (Contd.)



Binary Tree



Linked Representation



2 ways of implementation

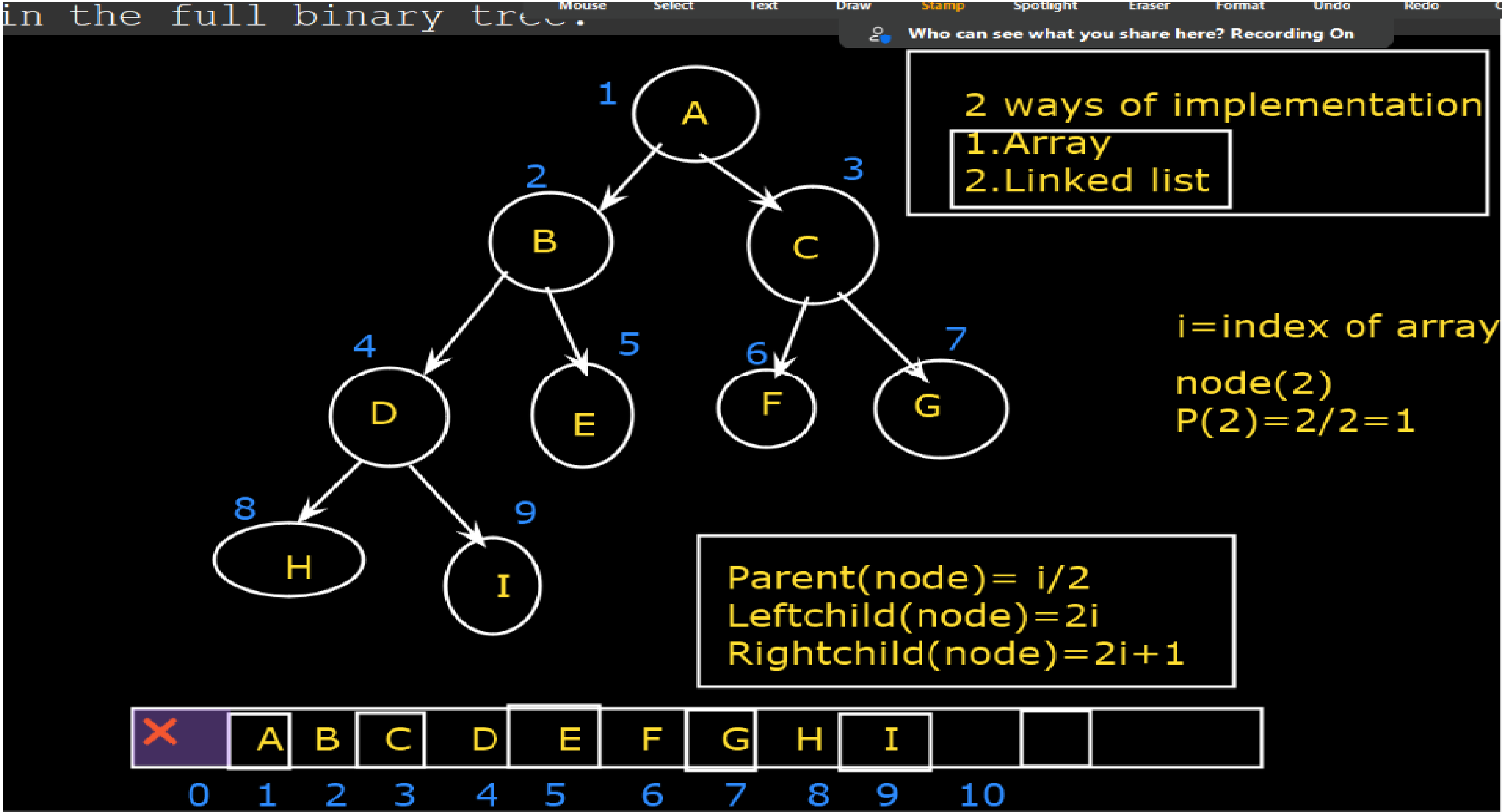
1.Array
2.Linked list

i=index of array

node(2)
P(2)=2/2=1

Parent(node)= i/2
Leftchild(node)=2i
Rightchild(node)=2i+1

11 22 33 44 55 66 77 88
99
0 1 2 3 4 5 6 7 8 9 10



in the full binary tree.

Who can see what you share here? Recording On

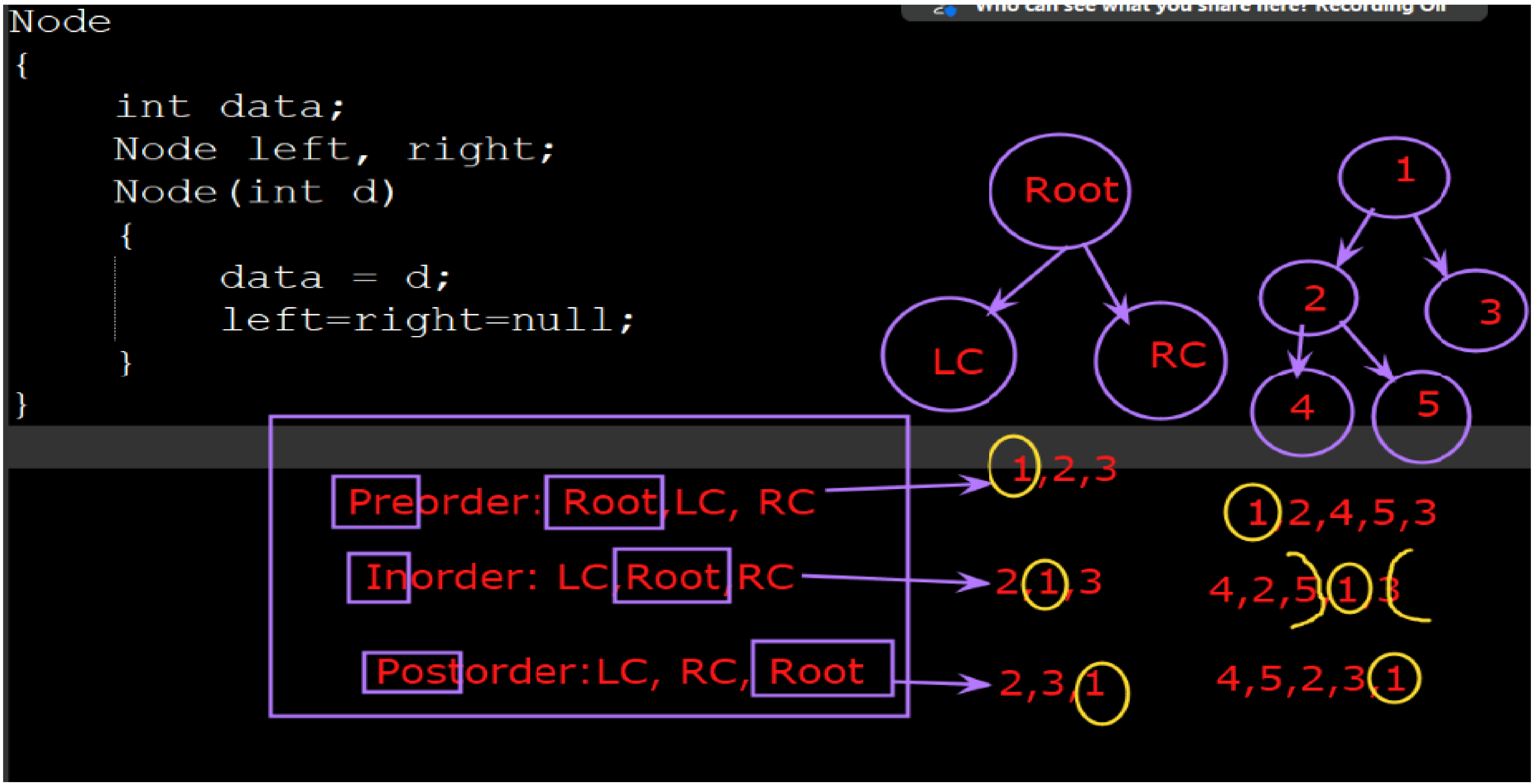
2 ways of implementation
1.Array
2.Linked list

i=index of array

node(2)
P(2)=2/2=1

Parent(node)= i/2
Leftchild(node)=2i
Rightchild(node)=2i+1

✗	A	B	C	D	E	F	G	H	I	
0	1	2	3	4	5	6	7	8	9	10



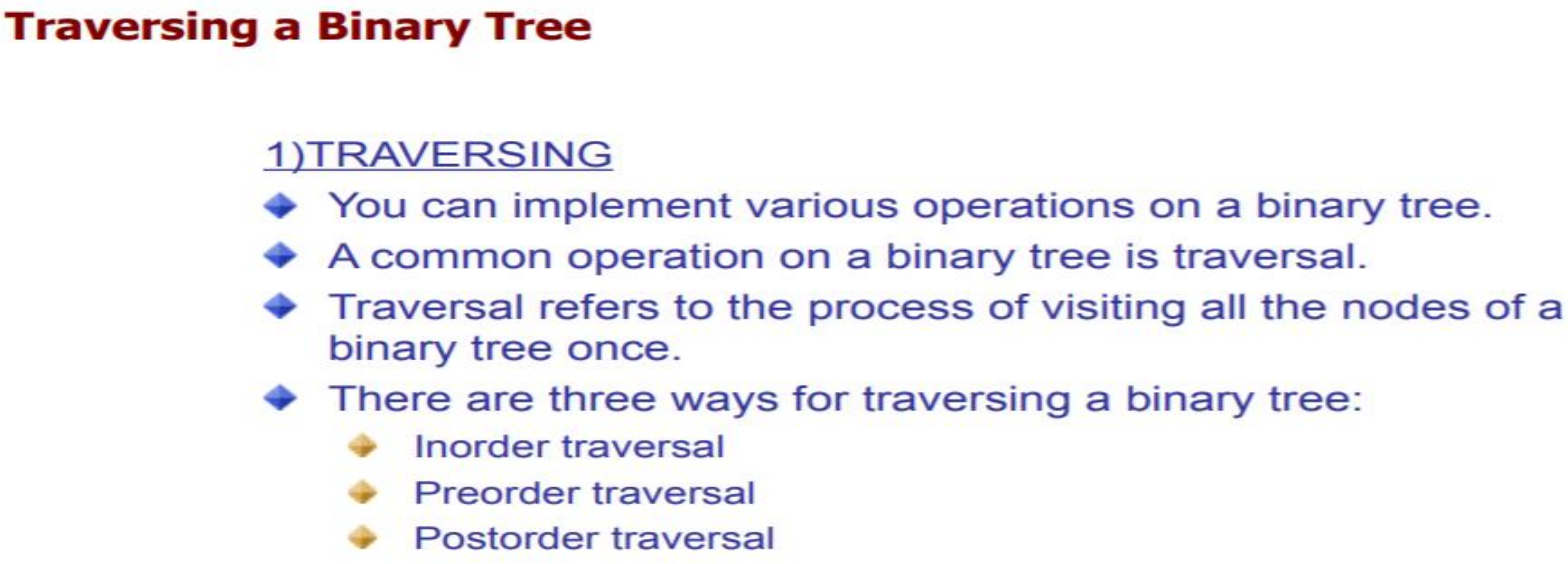
```
Node
{
    int data;
    Node left, right;
    Node(int d)
    {
        data = d;
        left=right=null;
    }
}
```

Preorder: Root,LC, RC → 1,2,3   1,2,4,5,3

Inorder: LC,Root,RC → 2,1,3   4,2,5,1,3

Postorder:LC, RC, Root → 2,3,1   4,5,2,3,1

OPERATIONS ON TREES

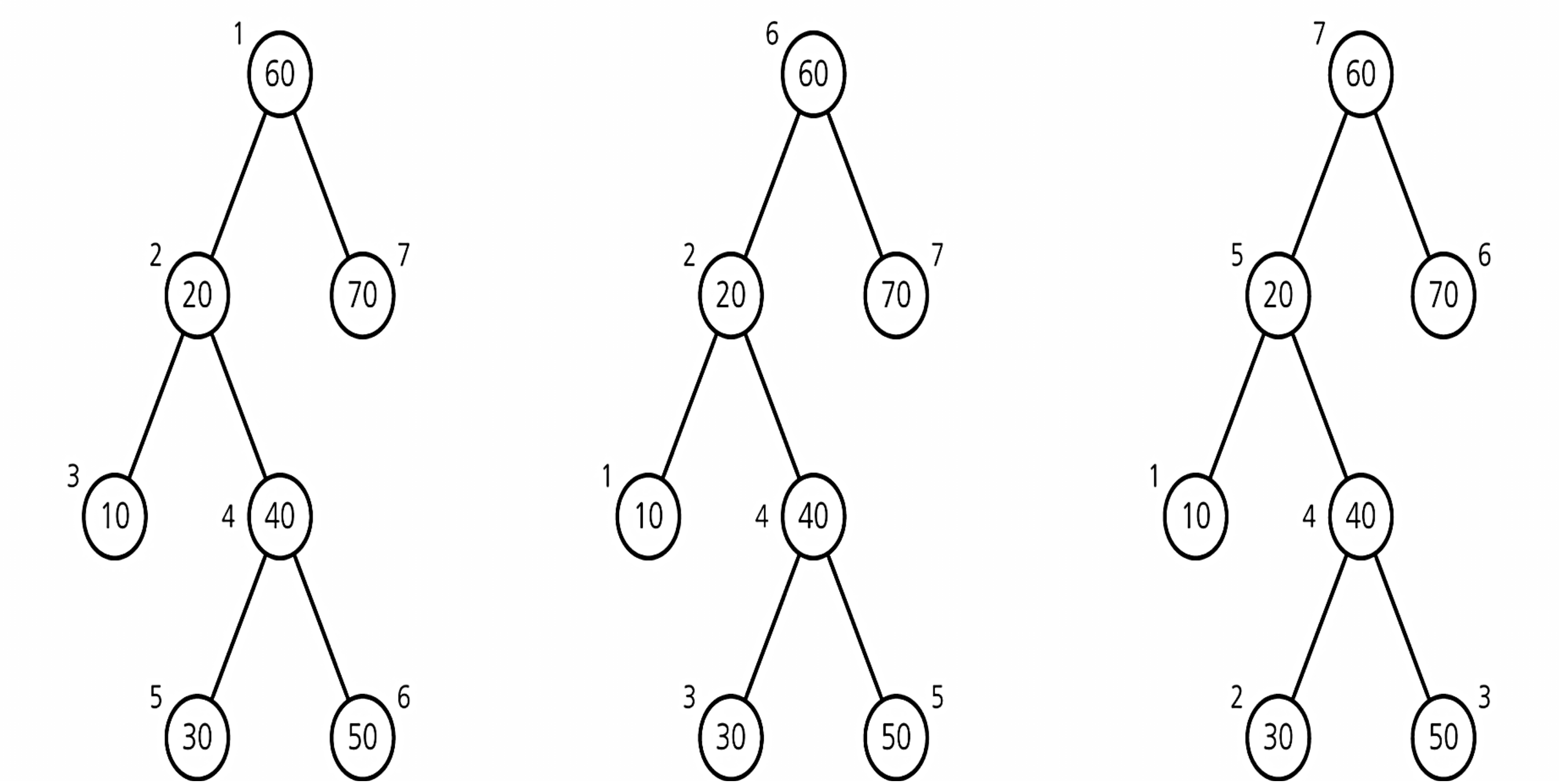


Traversing a Binary Tree

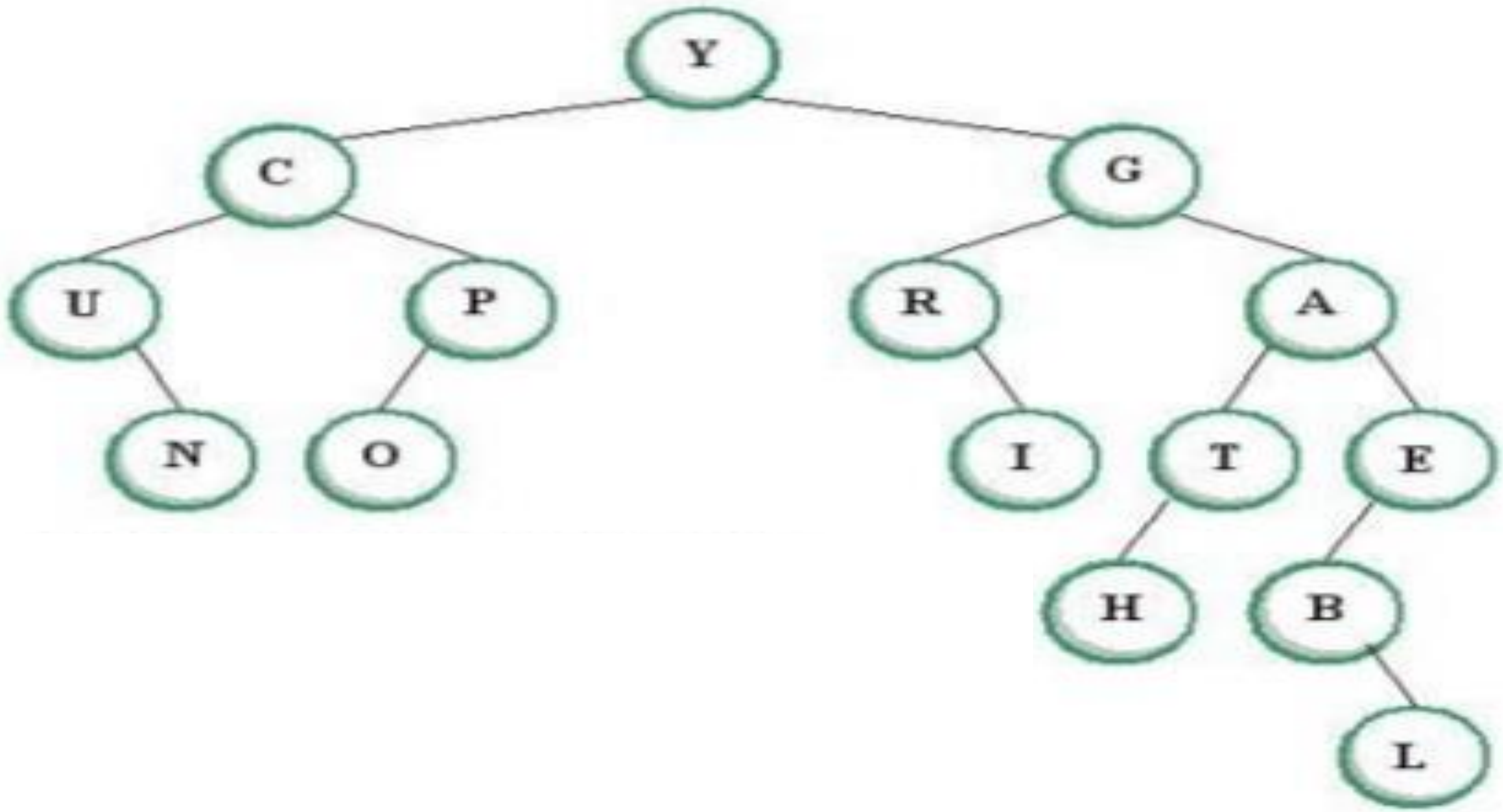
1)TRAVERSING

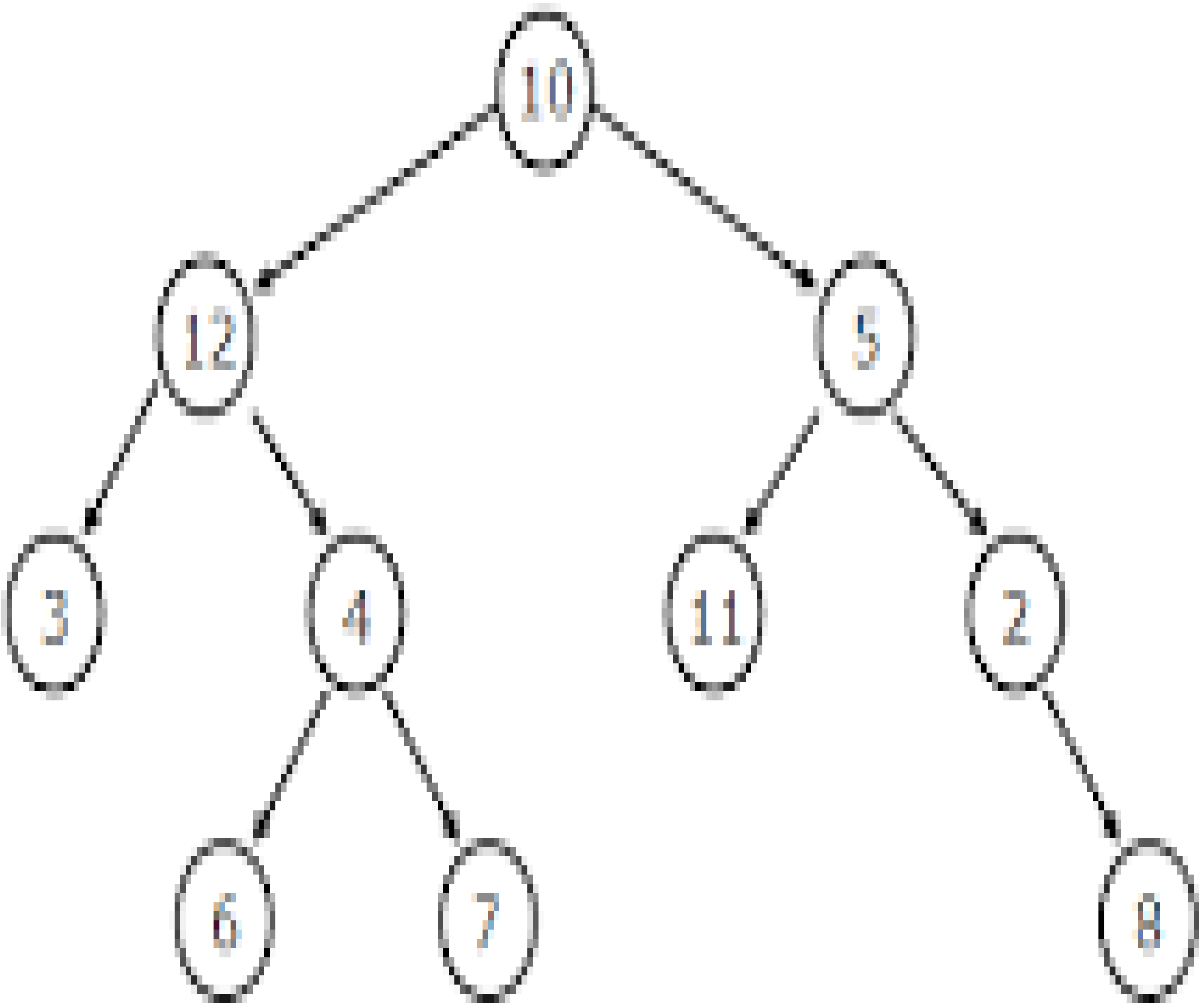
- You can implement various operations on a binary tree.
- A common operation on a binary tree is traversal.
- Traversal refers to the process of visiting all the nodes of a binary tree once.
- There are three ways for traversing a binary tree:
  - Inorder traversal
  - Preorder traversal
  - Postorder traversal

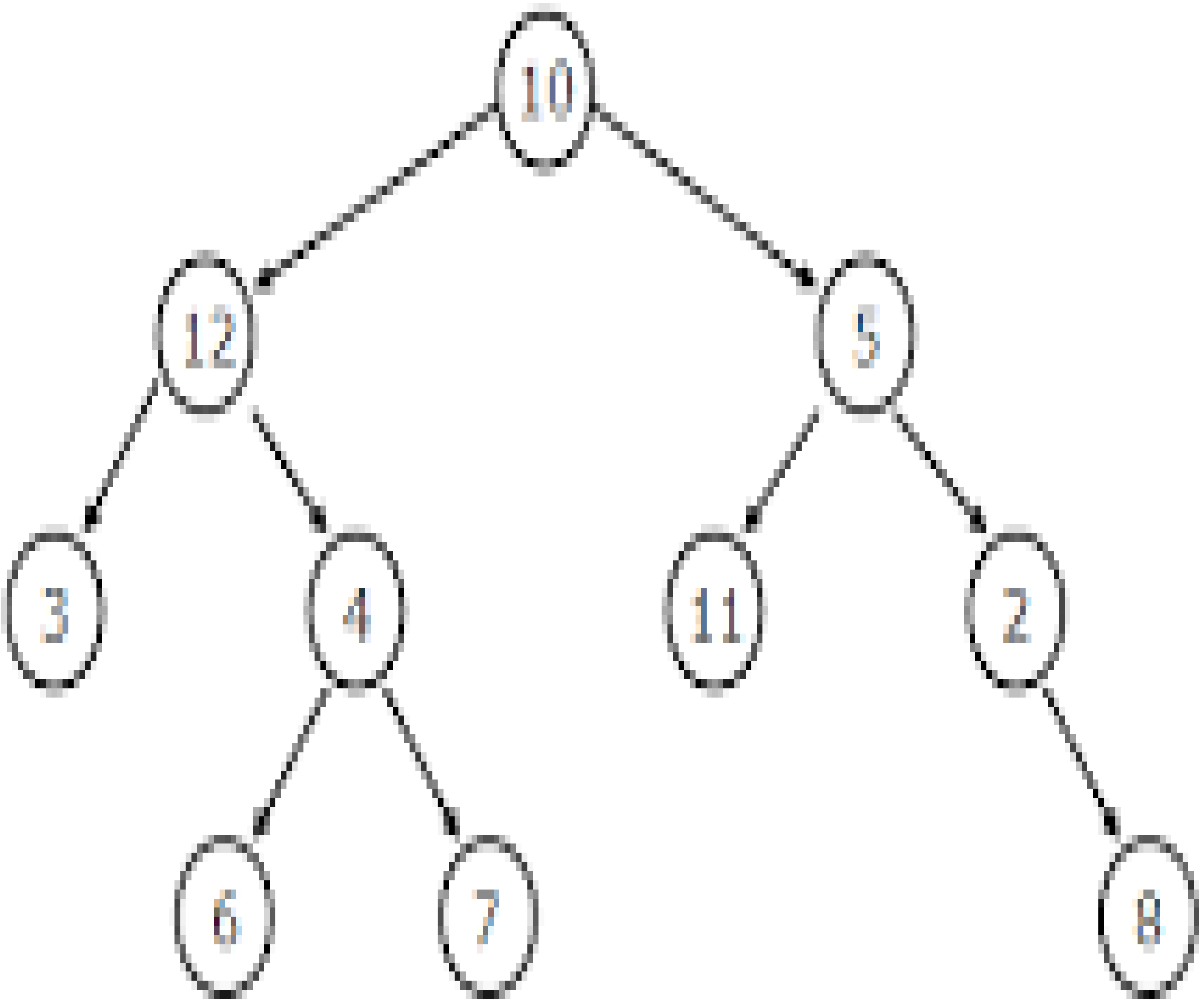
Binary Tree Traversals

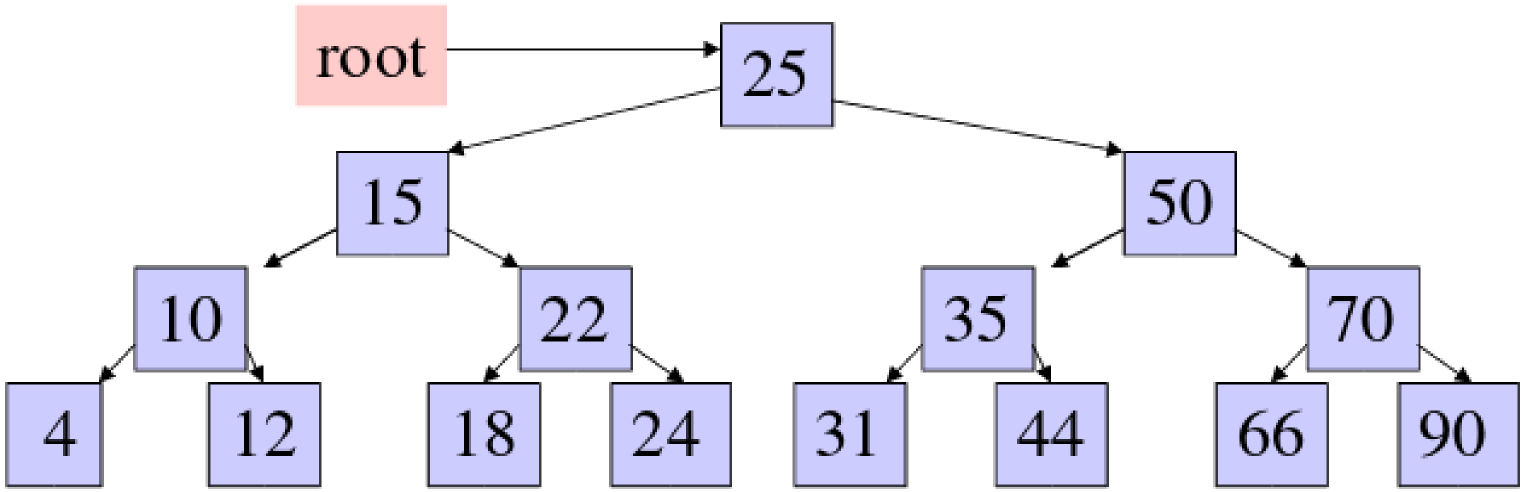


Traversal Examples



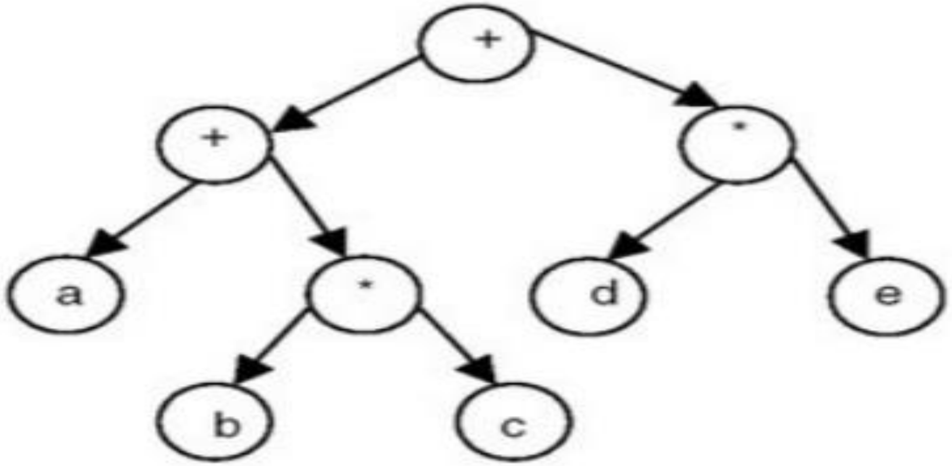

10
12
5
3
4
11
2
6
7
8


10
12
5
3
4
11
2
6
7
8


root
25
15
50
10
22
35
70
4
12
18
24
31
44
66
90

Expression Binary Tree Traversal

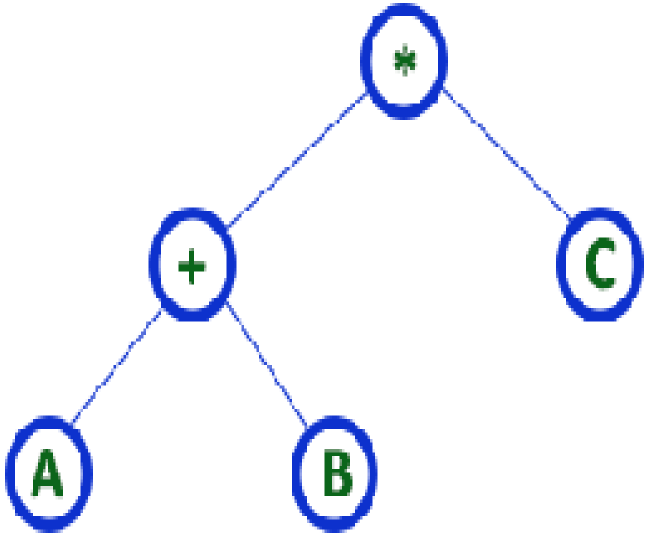
If an expression is represented as a binary tree, the inorder traversal of the tree gives us an infix expression, whereas the postorder traversal gives us a postfix expression as shown in Figure.



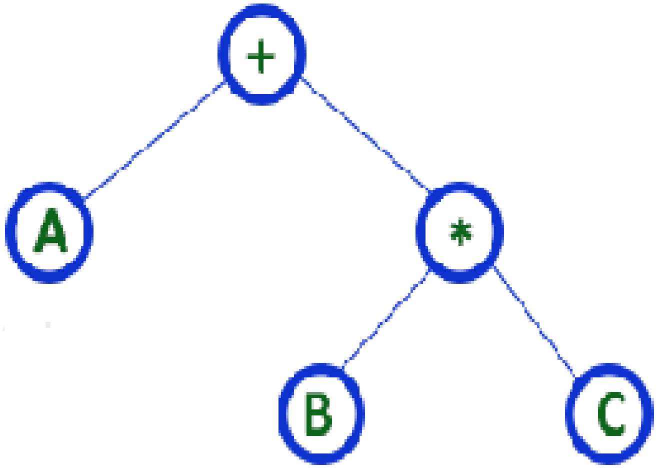
Inorder : a + b * c + d * e

postorder : abc*+de*+

Strictly binary tree data structure is used to represent mathematical expressions.

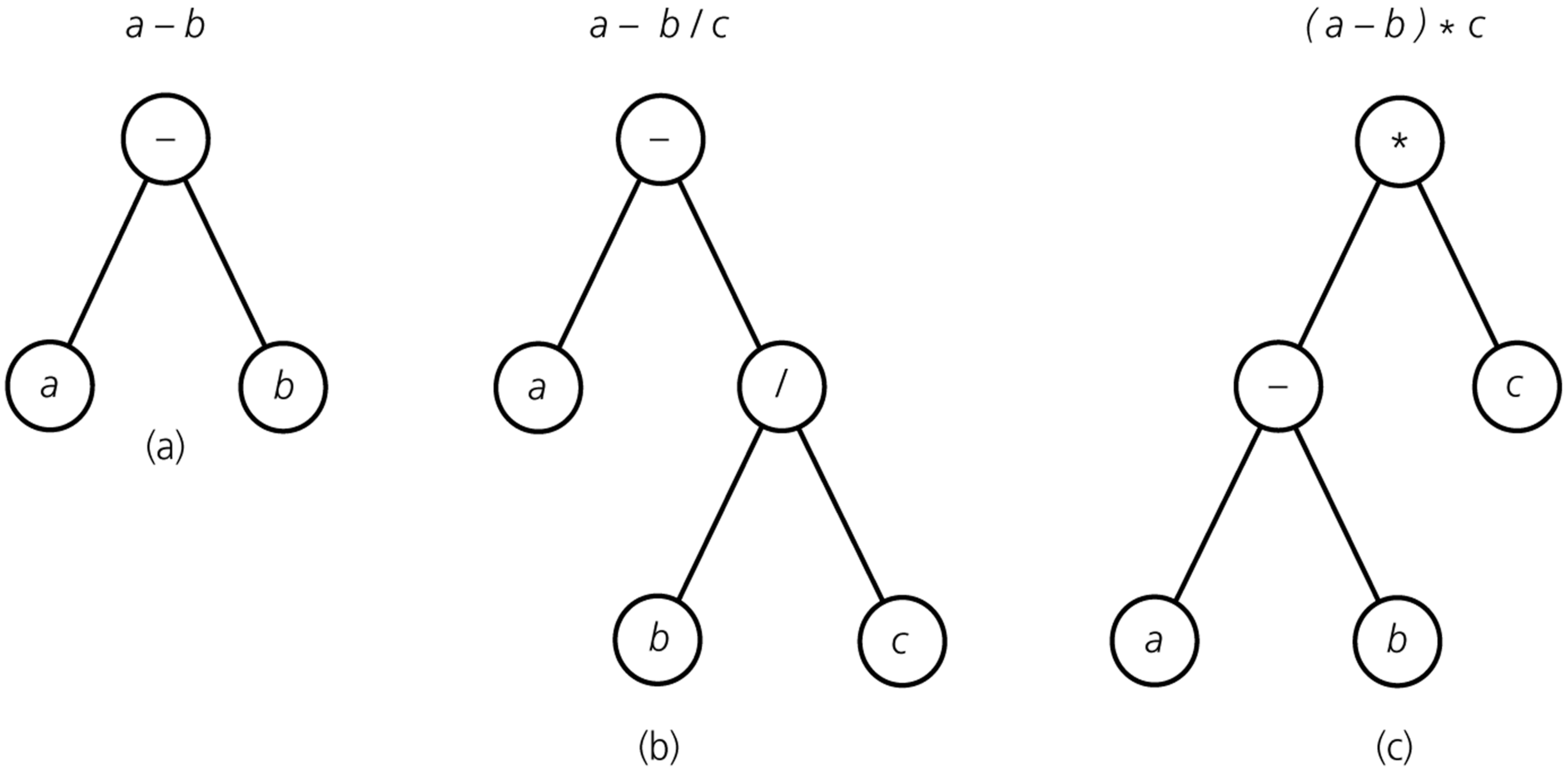


(A+B)*C



A+B*C

Binary Tree – Representing Algebraic Expressions



$a - b$

(a)

$a - b / c$

(b)

$(a - b) * c$

(c)

Thanks