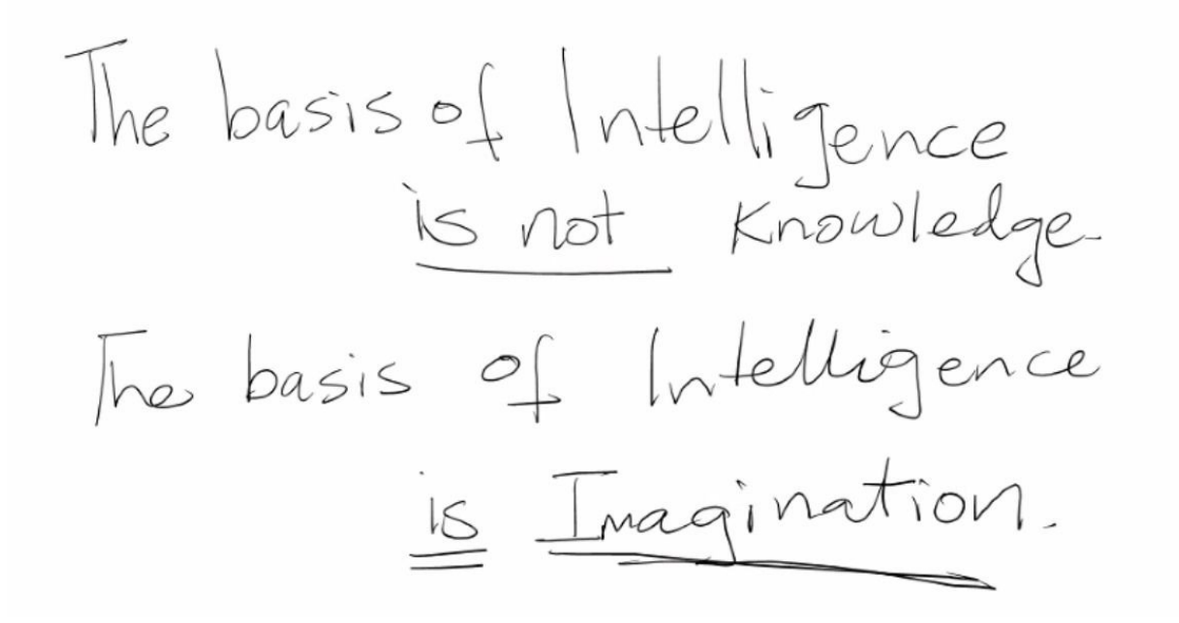
Trees Basics



– Late Prof. Ravi Kothari

Music: "Touch of the Sun" by AR Rahman from "127 Hours"

Agenda:

1. Insertion in linked list
2. ~~Trees Intro~~
3. ~~Naming convention~~
4. ~~Tree Traversal~~
5. Basic tree problems

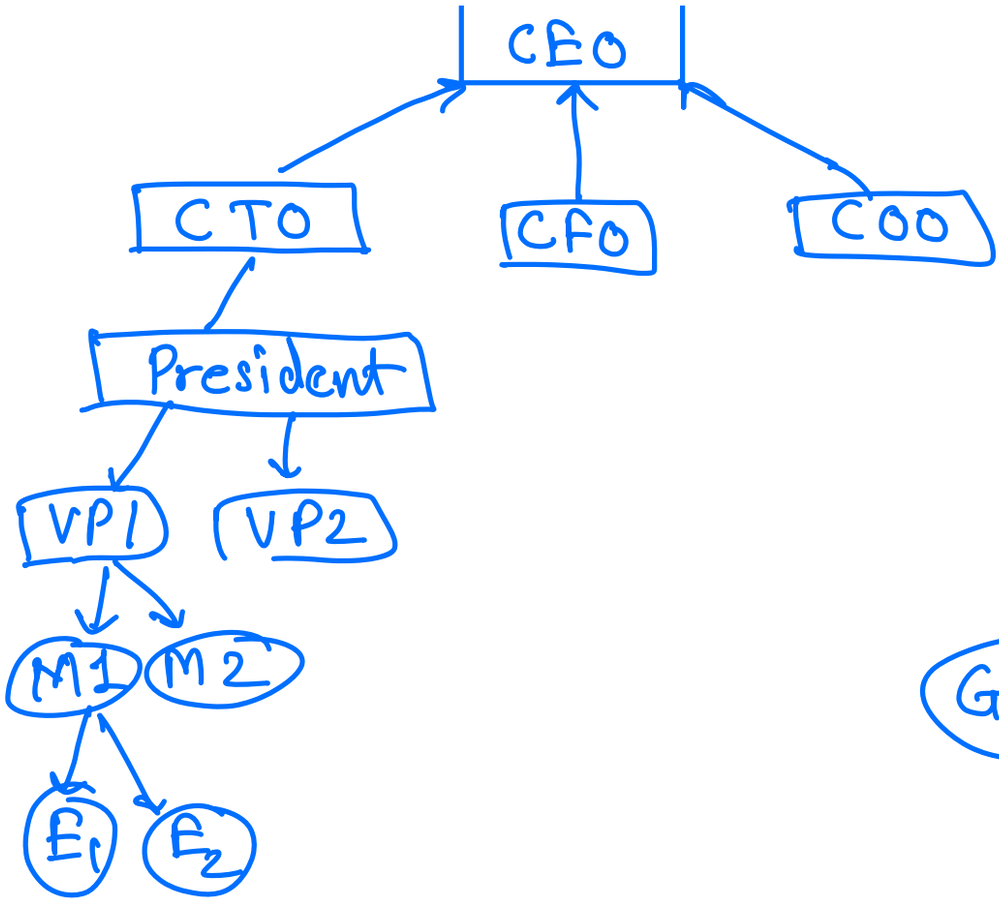
Linear Data structures:

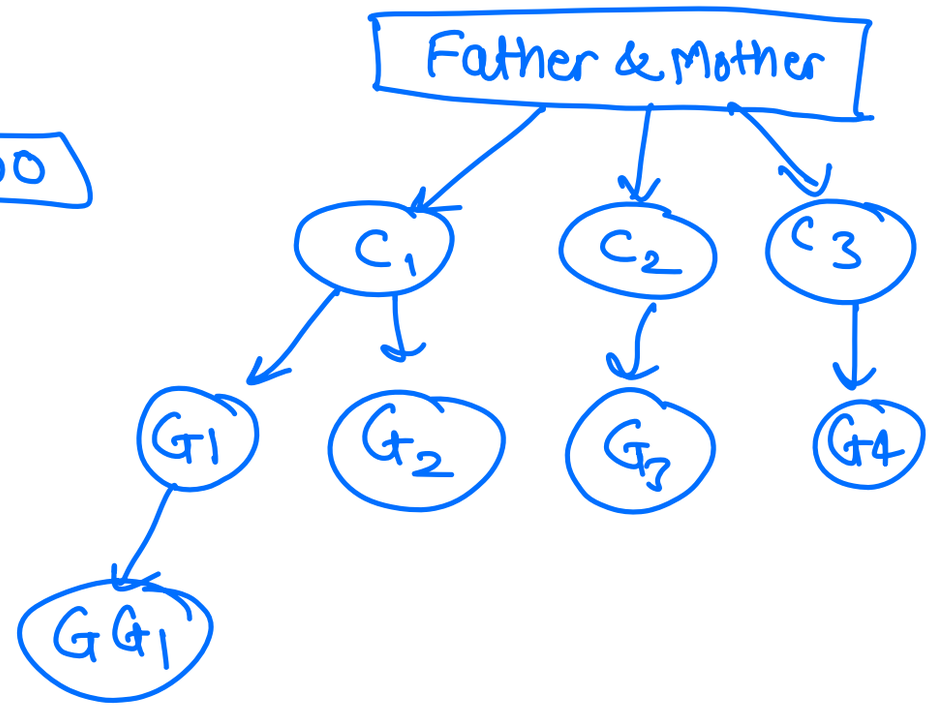
arrays | linked list | stacks, Qs

are useful for storing unrelated data

hirarchial Data

eg Company organization Family.



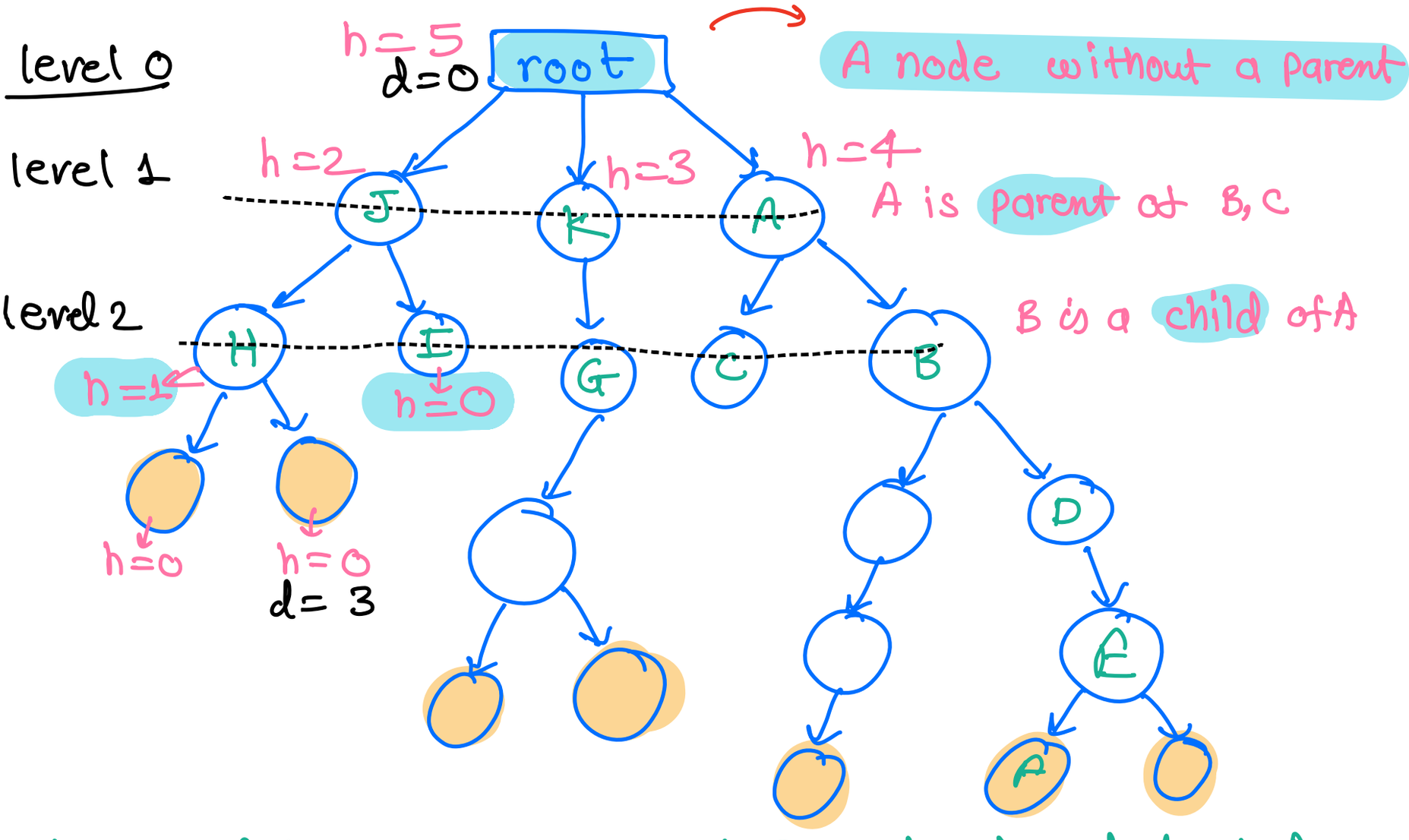


⇒ Tree data structure to store these kinds of data.

tree

eg: folders in our computers.

5. Tree data structure:



A → D A is the ancestor of D, D is decedant of A.
A → E A is the ancestor of E

B → C are siblings

H, G, C, B → nodes at the same level

F → leaf node : a node without children

Tree : ⊛ will have only one root
⊛ for every node we will have exactly one parent.

Height (node) : length of longest path from node to any of its descendent leaf node

Path : path is calculated based on no. of edges between the nodes.

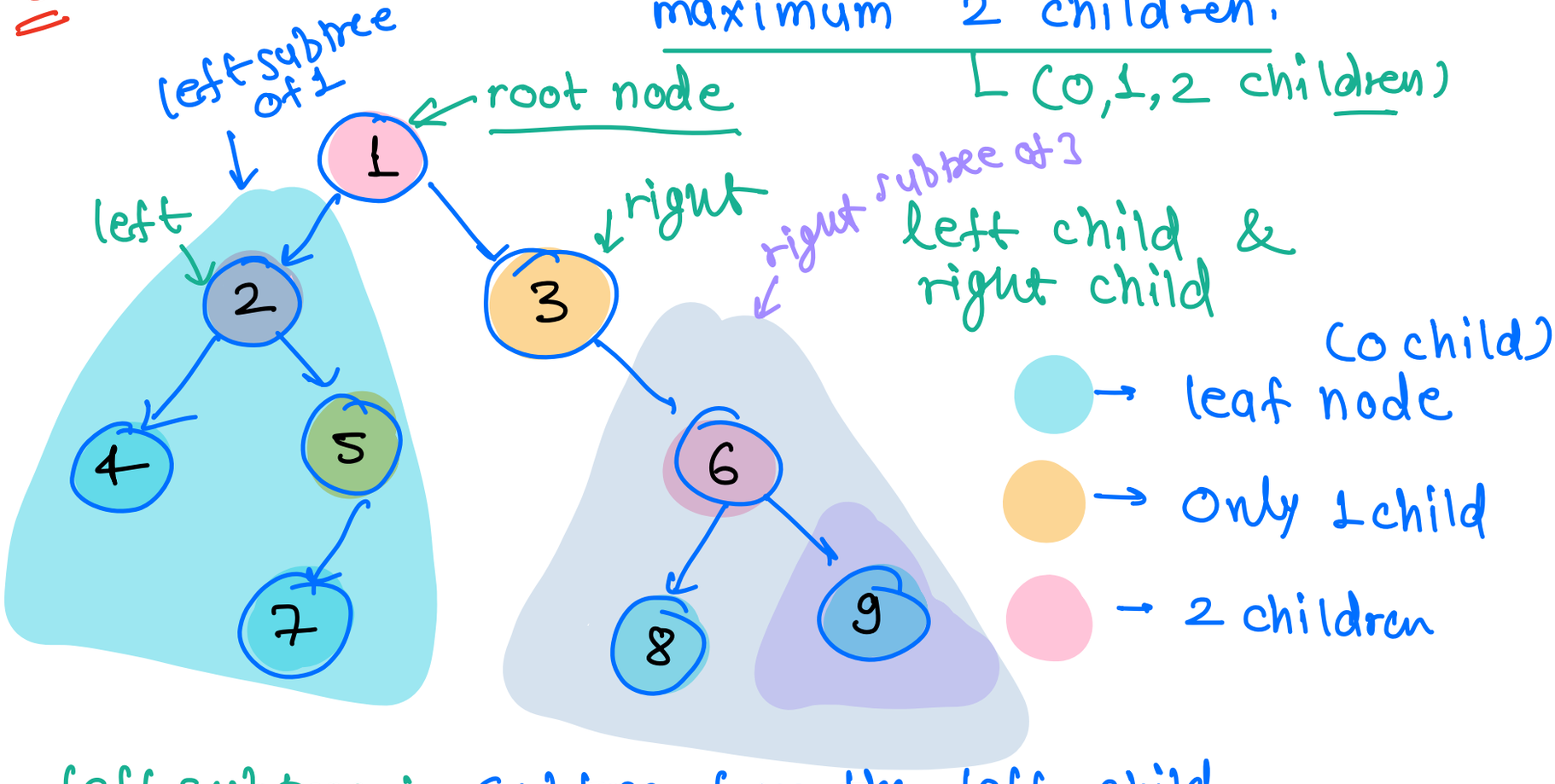
Depth (node) : length of the path from root to node

Obsⁿ : If the node is at depth d, child's
① depth will be d+1
② depth (root) = 0

depth (node) ≠ height (node)

8. Binary tree: A tree where every node have maximum 2 children.

(0, 1, 2 children)

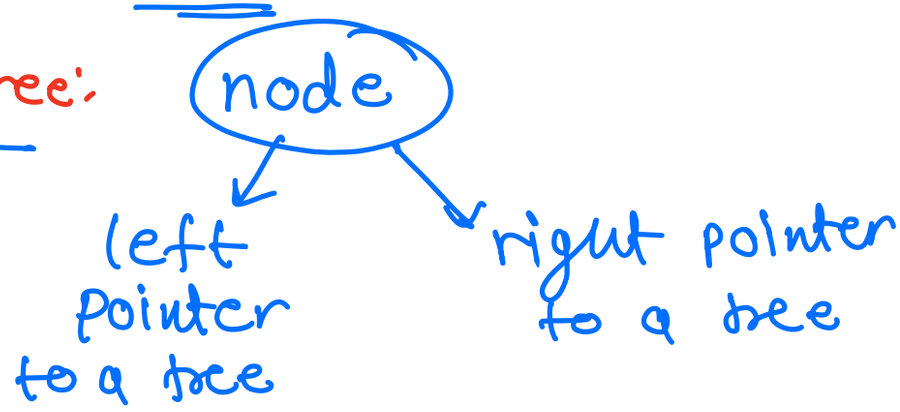


leftsubtree : subtree from the left child

right subtree : subtree from the right child

left subtree of node 3: NULL

Defining the Binary Tree:




```
Class  TreeNode {
        int  data;
        TreeNode left;
        TreeNode right;
        TreeNode (int num){
            this.data = num:
```

TreeNode mytree

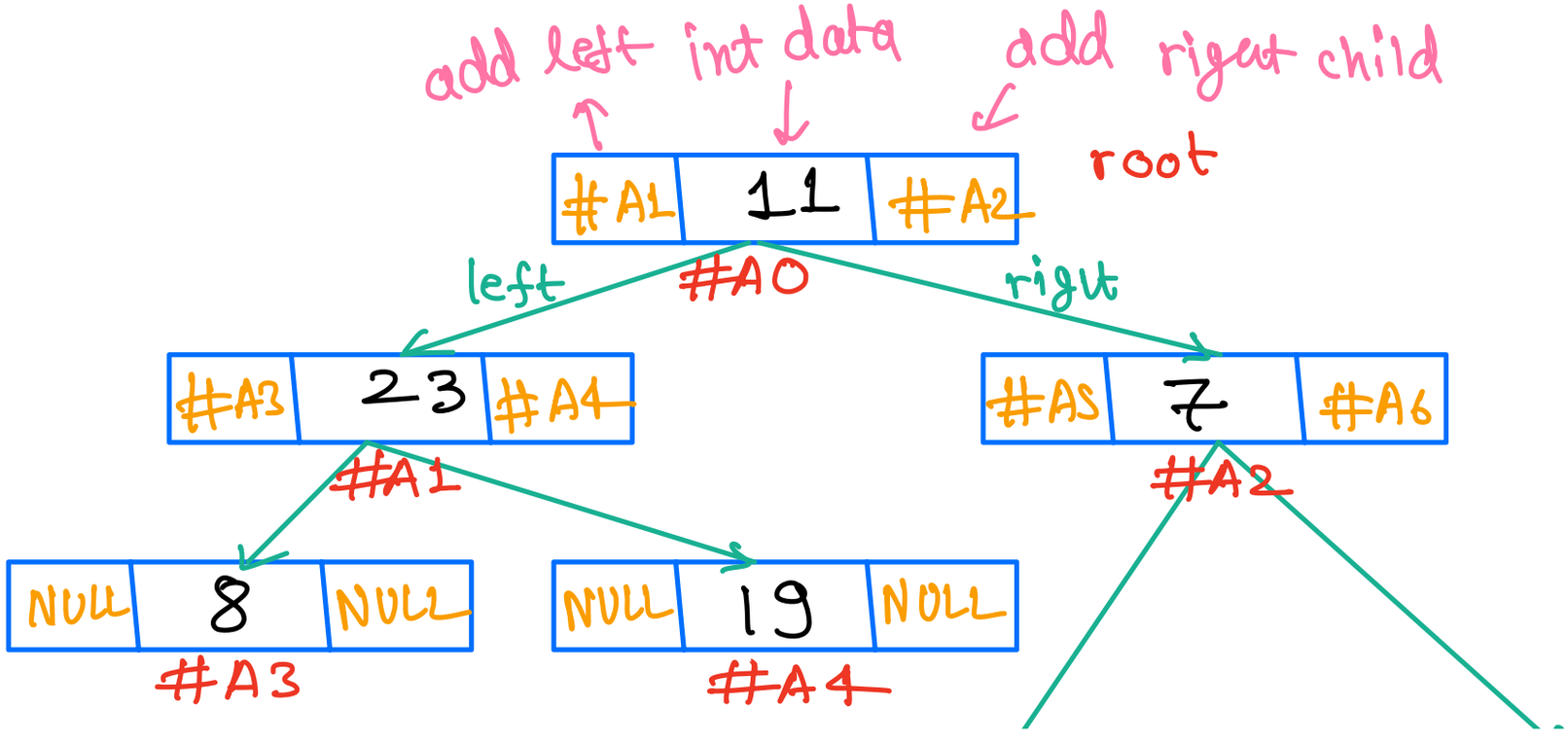
// left contain the address of left child

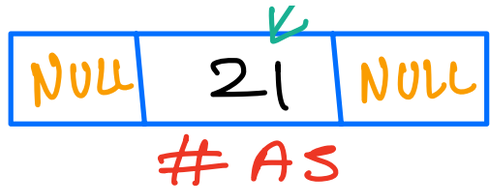
// right contain the address of right child

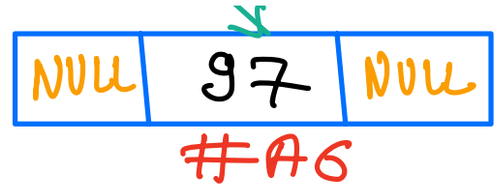
```
        this.left = null;
        this.right = null;
    }
}
```

Obsⁿ: Given the root we can visit each node of the tree

// for all the problems, tree is already constructed and we are given the root Node.

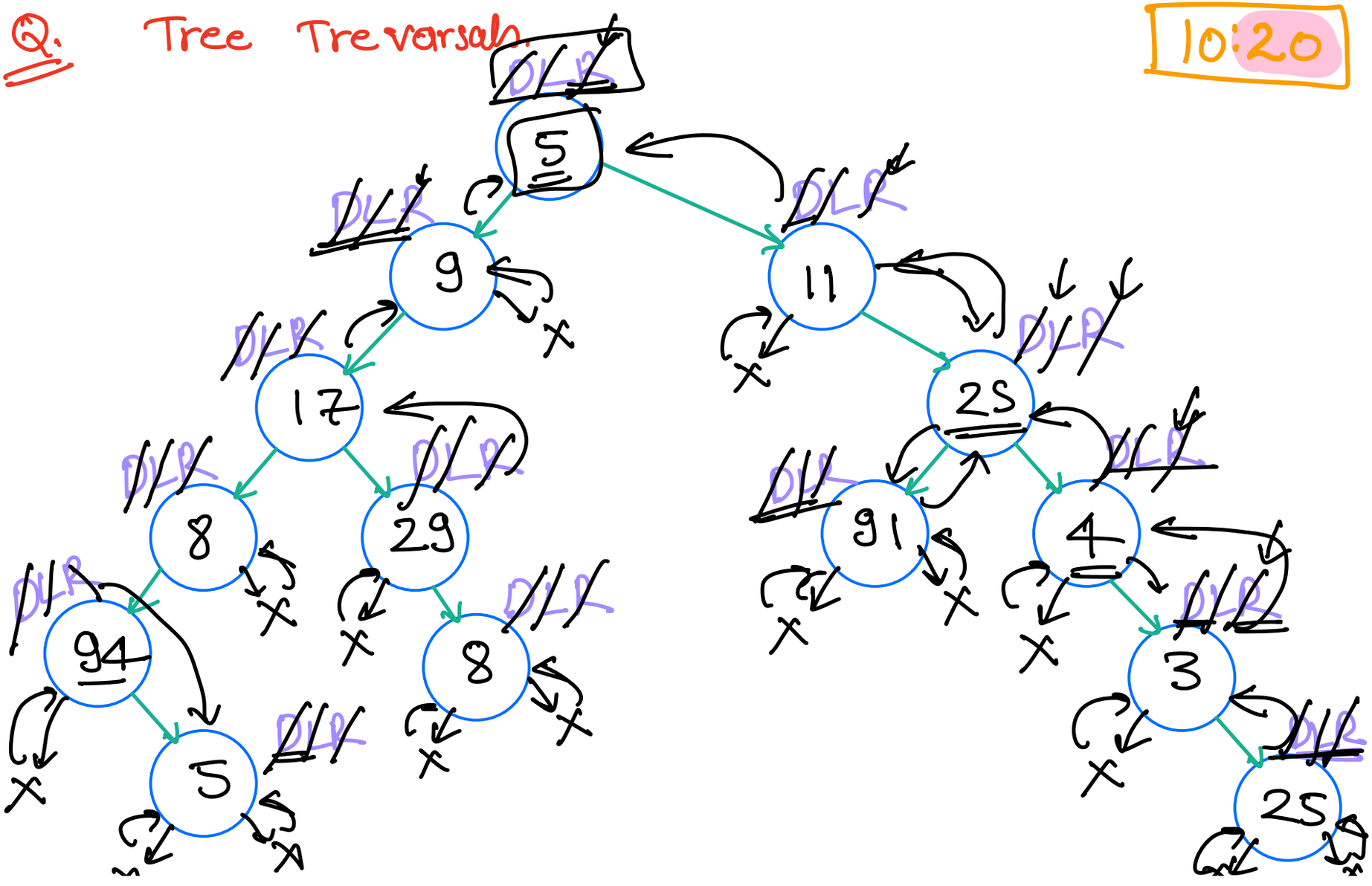






10:20

Q. Tree Traversals



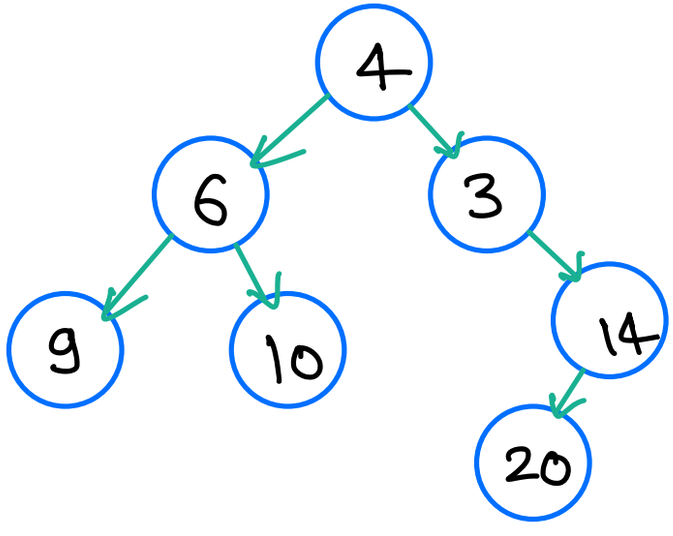
Output: 5, 9, 17, 8, 94, 5, 29, 8, 11, 25, 91, 4, 3, 25

① Preorder DLR → (Recursion)

Step 1: Process the current node (Print the data)

Step 2: go to left subtree and print the left subtree in preorder

Step 3: go to right subtree and print it in preorder



- Preorder DLR : 4 6 9 10 3 14 20
- inorder LDR : 9 6 10 4 3 20 14
- postorder LRD : 9 10 6 20 14 3 4

Code:

Q. Given root node of a binary tree, print elements in preorder. DLR

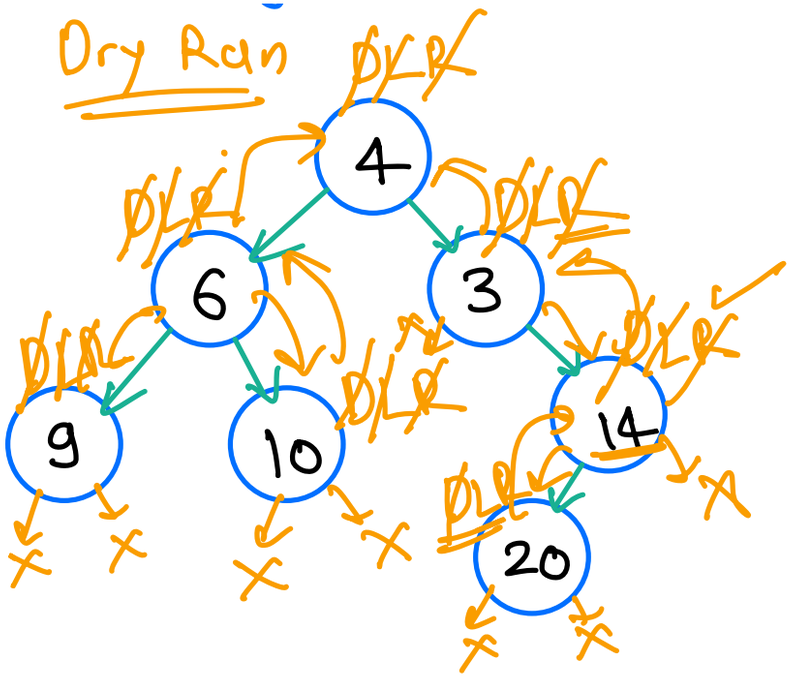
```
void preorder (TreeNode root){
    1. if (root == null) return;
    2. print (root.data);
    3. preorder (root.left);
    4. preorder (root.right);

    return;
}
```

4 6 9 10 3 14 20

T.C. = O(N)
S.C. = O(H)
↑
H is the height of the tree

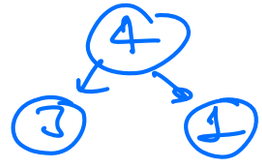
Create function call copies

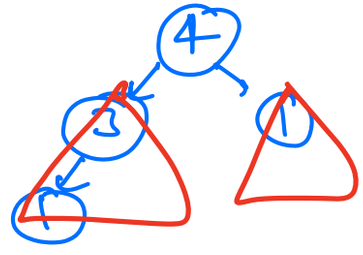
Dry Run



T_0 root = NULL ⇒ return

T_1  ⇒ 4

T_2  ⇒ 431

T_3 

H.W: write the code for inorder & postorder using recursion

Q. Using recursion: given root node

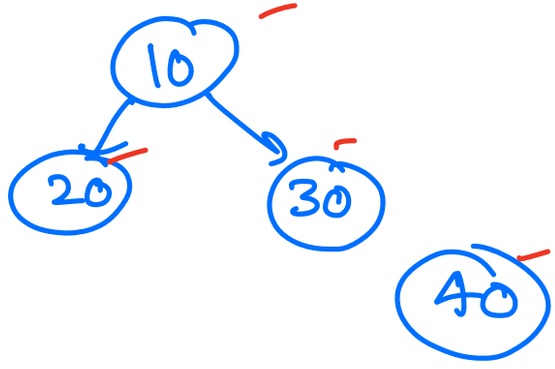
1. Size of the tree
2. Sum of all elements of the tree
3. Height of the tree

9:10:30

10:30 - midnight discussion

9. ① Size of the tree

ex:



ans = 4

idea: ① if root == null return 0;

② How to break big problem into smaller problem of exact same type.

```
int treeSize(Node root){

    if(root == NULL) return 0;
    int leftSubTreeSize = treeSize(root.left);

    int rightSubTreeSize = treeSize(root.right);

    return leftSubTreeSize + rightSubTreeSize+1;
}
```

§. ② sum of all elements of the tree, given root node.

idea: ① by recursion get the sum of left subtree
right subtree

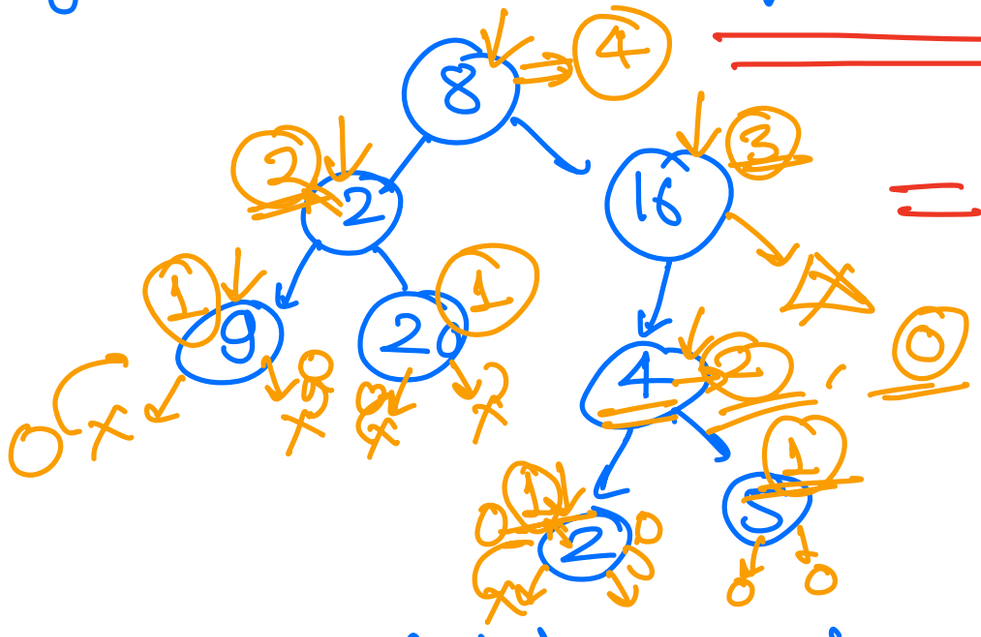
② sum = leftSubtreeSum + rightsubtreeSum
+ root.data

```
int treeSum (Node root){
    if (root == NULL) return 0;
    int leftSubTreeSum = treeSum (root.left);
    int rightSubTreeSum = treeSum (root.right);
    return leftSubTreeSum + rightSubTreeSum
           + root.data;
}
```

5. ③ Height of the tree

Height of the tree is equal to height of the root.

ex



= 3

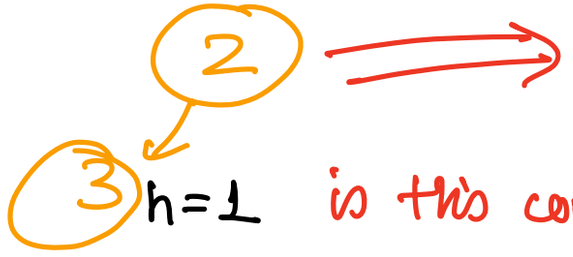
4

⇓

It height is based on nodes instead of edges

```
int treeHeight(Node root){
1.    if (root == NULL) return 0;          // the root is leaf than return 0
2.    int leftSubTreeHeight = treeHeight(root.left);
3.    int rightSubTreeHeight = treeHeight(root.right);
4.    return max(leftSubTreeHeight, rightSubTreeHeight) + 1
```

}



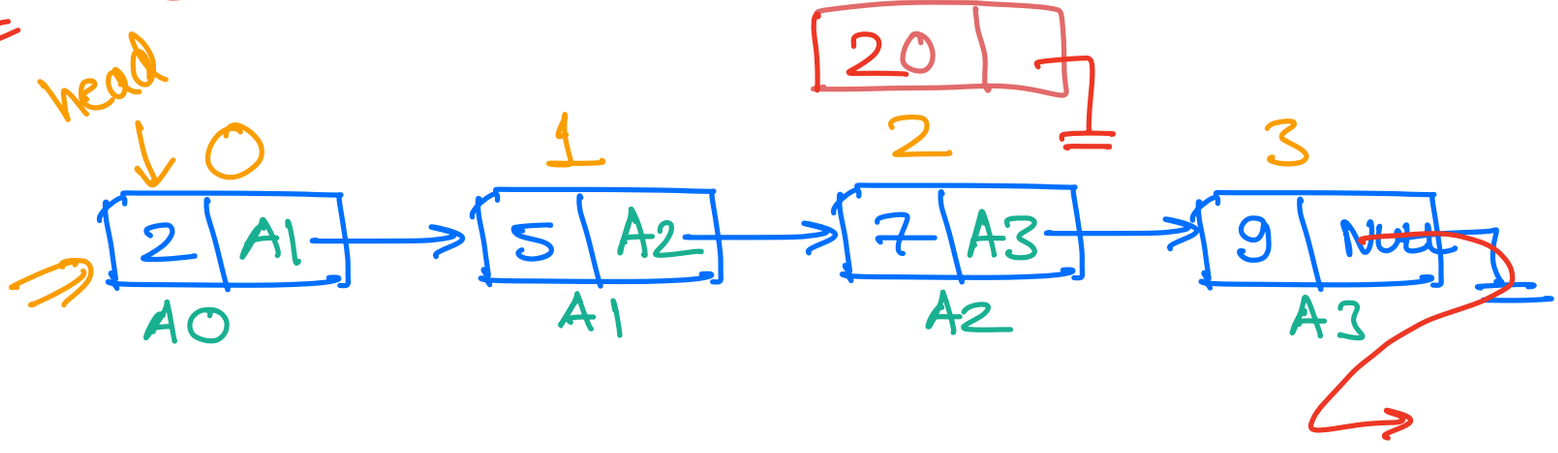
how many nodes = 2
edges = 1

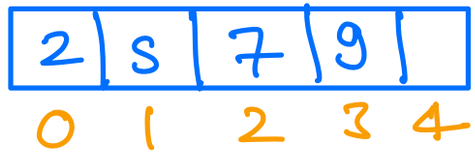
is this correct?

Note: in your assignments, height is defined as number of nodes instead of number of edges.

⇒ difference of 1 would come

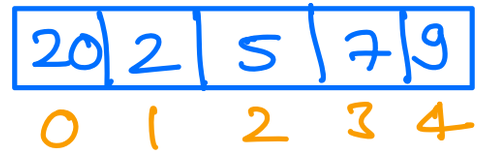
Insertion in linked list





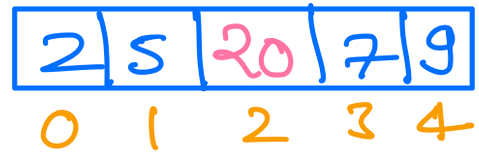
insert 20: where

i=0



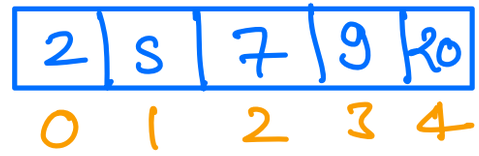
Start

i=2



middle

j=4



end

```
Node insert ( Node head, int index, int num){
    Node new_node = new Node (num);
    if (index == 0){
        new_node.next = head;
        head = new_node;
        return head;
```

```
}
Node it = head;
for (int i=0; i< index; i++){
        if (it.next == null) break;
        it = it.next;
}
Node next_node = it.next
it.next = new_node

new_node.next = next_node;

return;
}
```