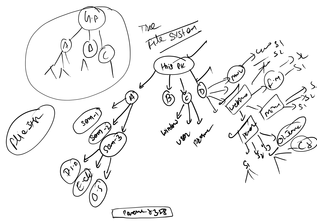
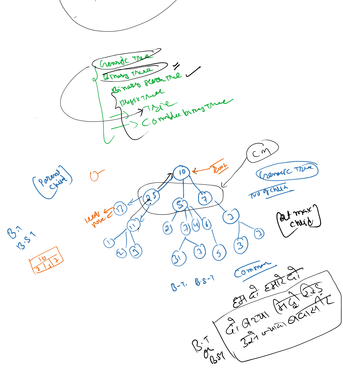
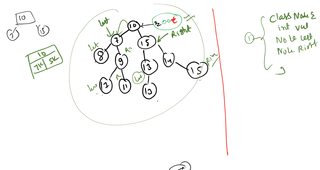
Lec-28

Trees
Binary Tree
Binary Search Tree
AVL Tree
Family Tree

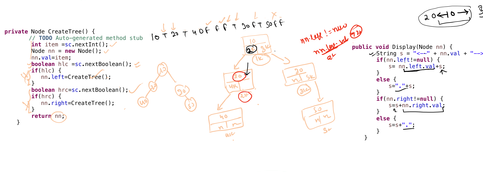

The File System
Root Dir
Pictures
Videos
Documents
Music

Balanced
Unbalanced
Complete Tree
Strictly Binary Tree
Almost Complete Tree
Degenerate Tree
Skewed Tree
Complete Binary Tree


Balanced Tree
Leaf Node
Root
Complete Tree
Not Complete
Not Balanced
B.T. B.S.T
Common
Either it is Empty
Or it has both left & right subtree
Both subtrees are balanced

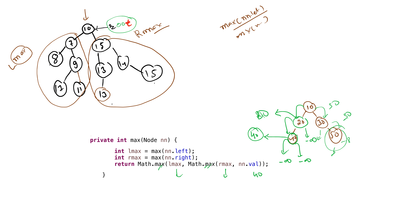

Left
Right
Leaf Node
Node with two Null Nodes

Print
Root Left Right


public Node CreateTree() {
// TODO Auto-generated method stub
int item =sc.nextInt();
Node nn = new Node();
nn.val=item;
boolean hlc =sc.nextBoolean();
if(hlc) {
nn.left=CreateTree();
}
boolean hrc=sc.nextBoolean();
if(hrc) {
nn.right=CreateTree();
}
return nn;
}
10 T 20 T 40 F F T 30 F T 50 F F
public void Display(Node nn) {
String s = "<--" + nn.val + "-->";
if(nn.left!=null) {
sm nn.left.val+";
}
else {
s+"_";
}
if(nn.right!=null) {
sms+nn.right.val;
}
else {
sms+"_";
}
System.out.println(s);
}

Display(nn.left);
Display(nn.right);
}

Root
Left
Right
max(nn.val)
nn.val
private int max(Node nn) {
int lmax = max(nn.left);
int rmax = max(nn.right);
return Math.max(lmax, Math.max(rmax, nn.val));
}


private int ht(Node node) {
if(node==null) {
return 0;
}
int lht=ht(node.left);
int rht=ht(node.right);
return Math.max(lh, rht)+1;
}
Pre-order
Post-order
In-order
Level-order
Root Left Right
Left Right Root
LN Rn Rt
Inorder
Preorder
Postorder
Levelorder
1: Queue
2: Node
3: Print
while(!q.isEmpty()) {
Node rn=q.poll();
System.out.print(rn.val+" ");
if(rn.left!=null) {
q.add(rn.left);
}
if(rn.right!=null) {
q.add(rn.right);
}
}
System.out.println();
}