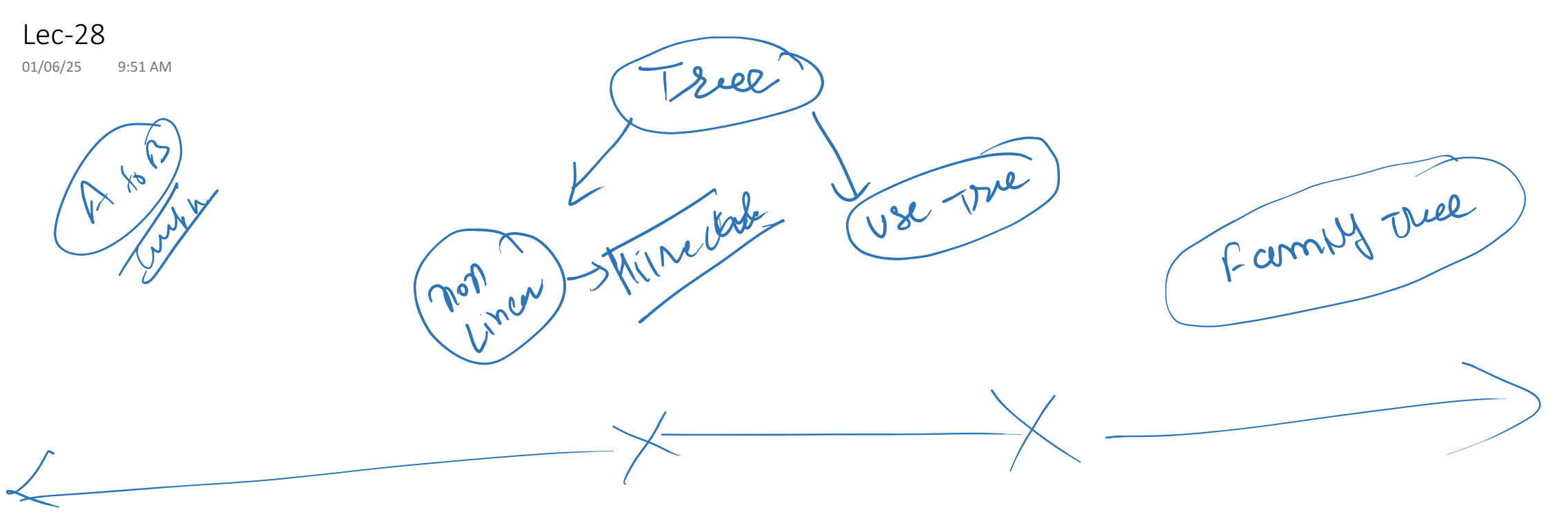
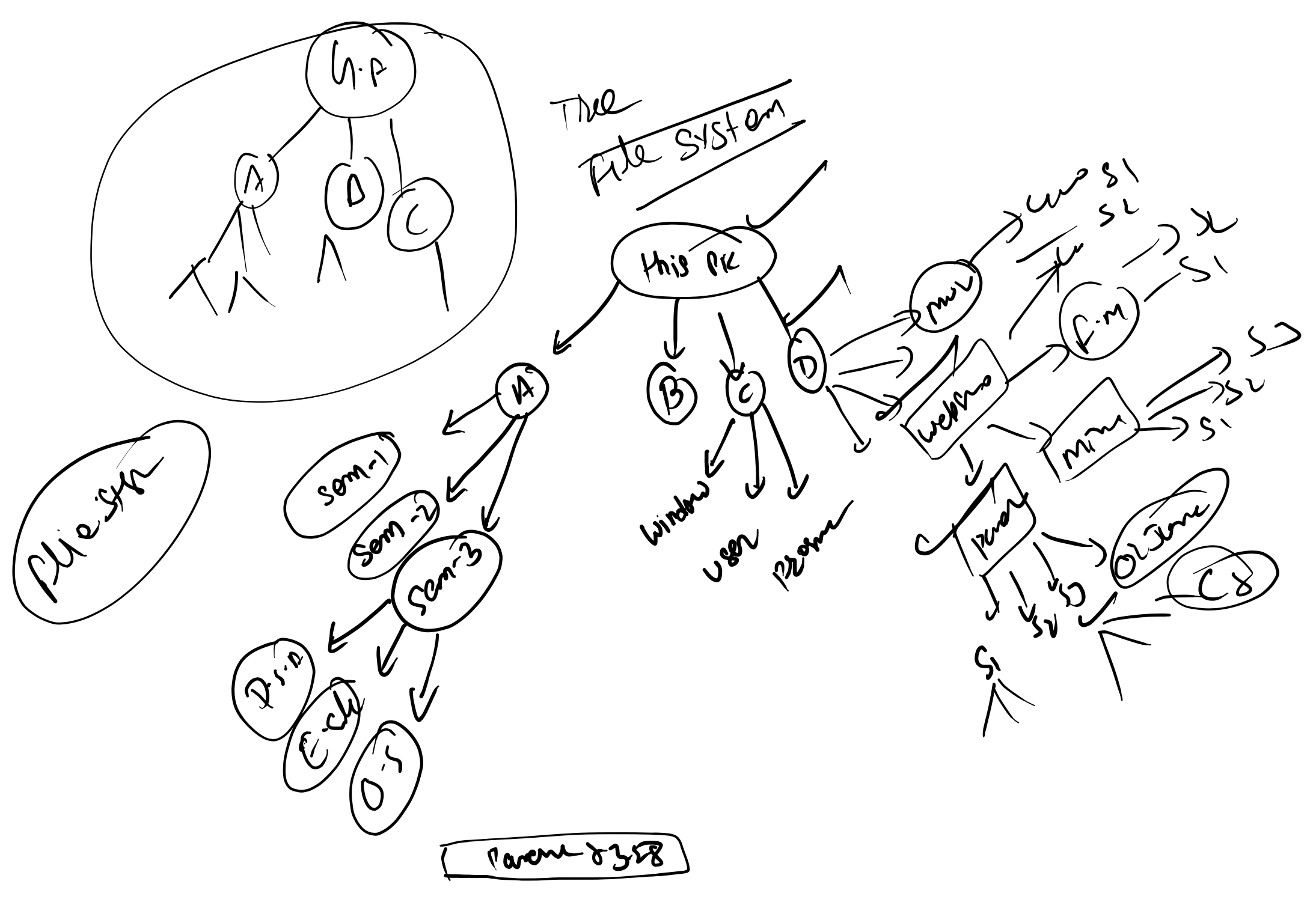
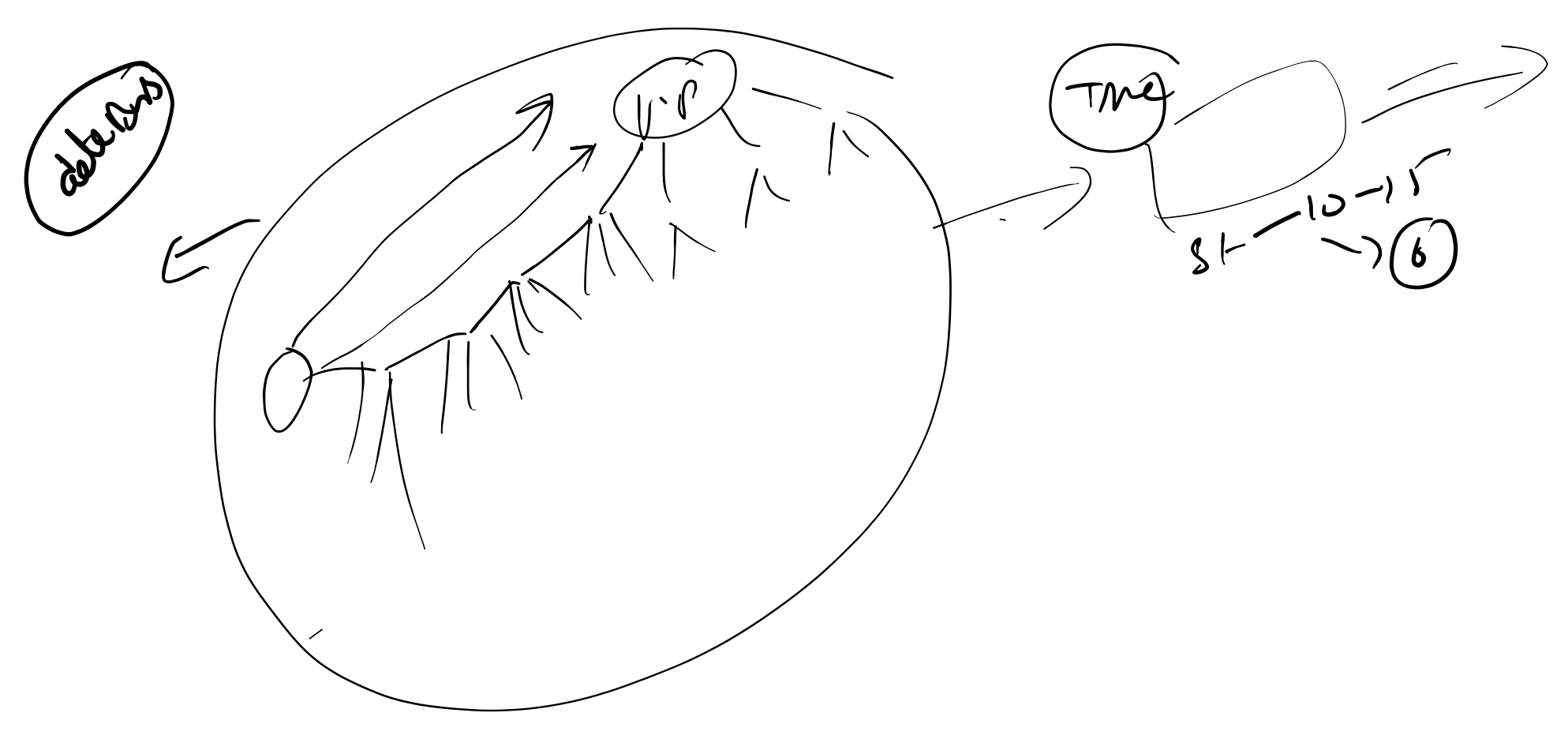
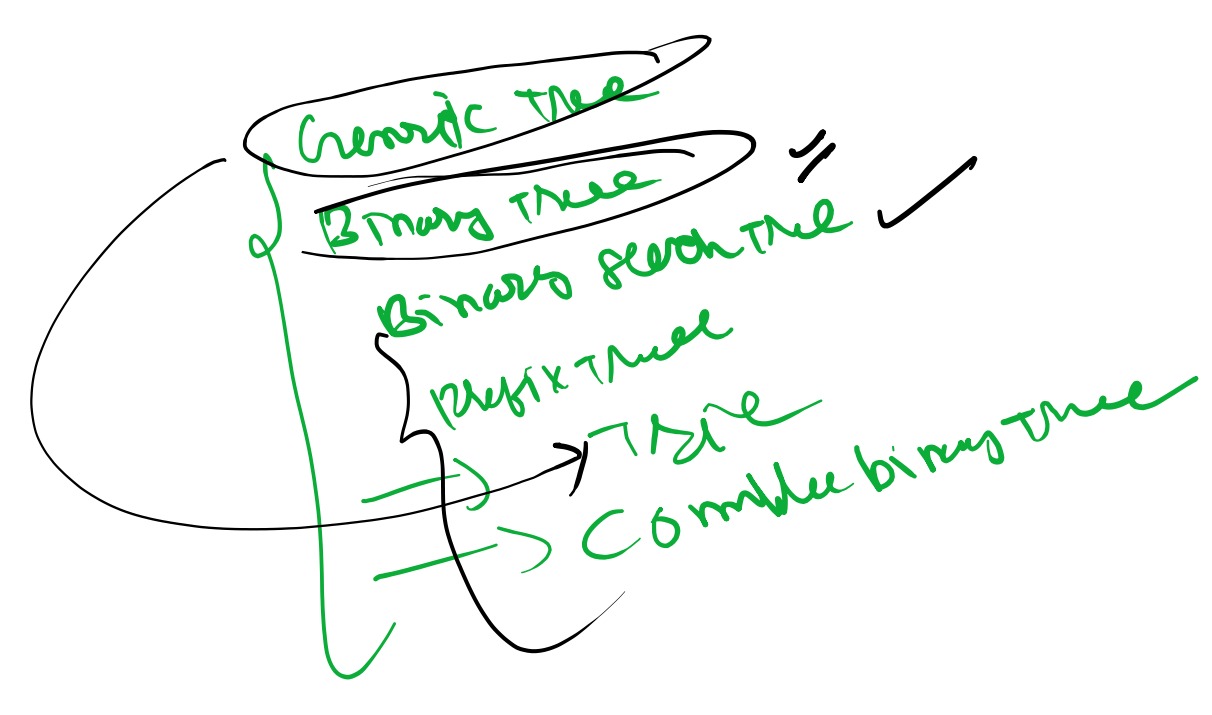
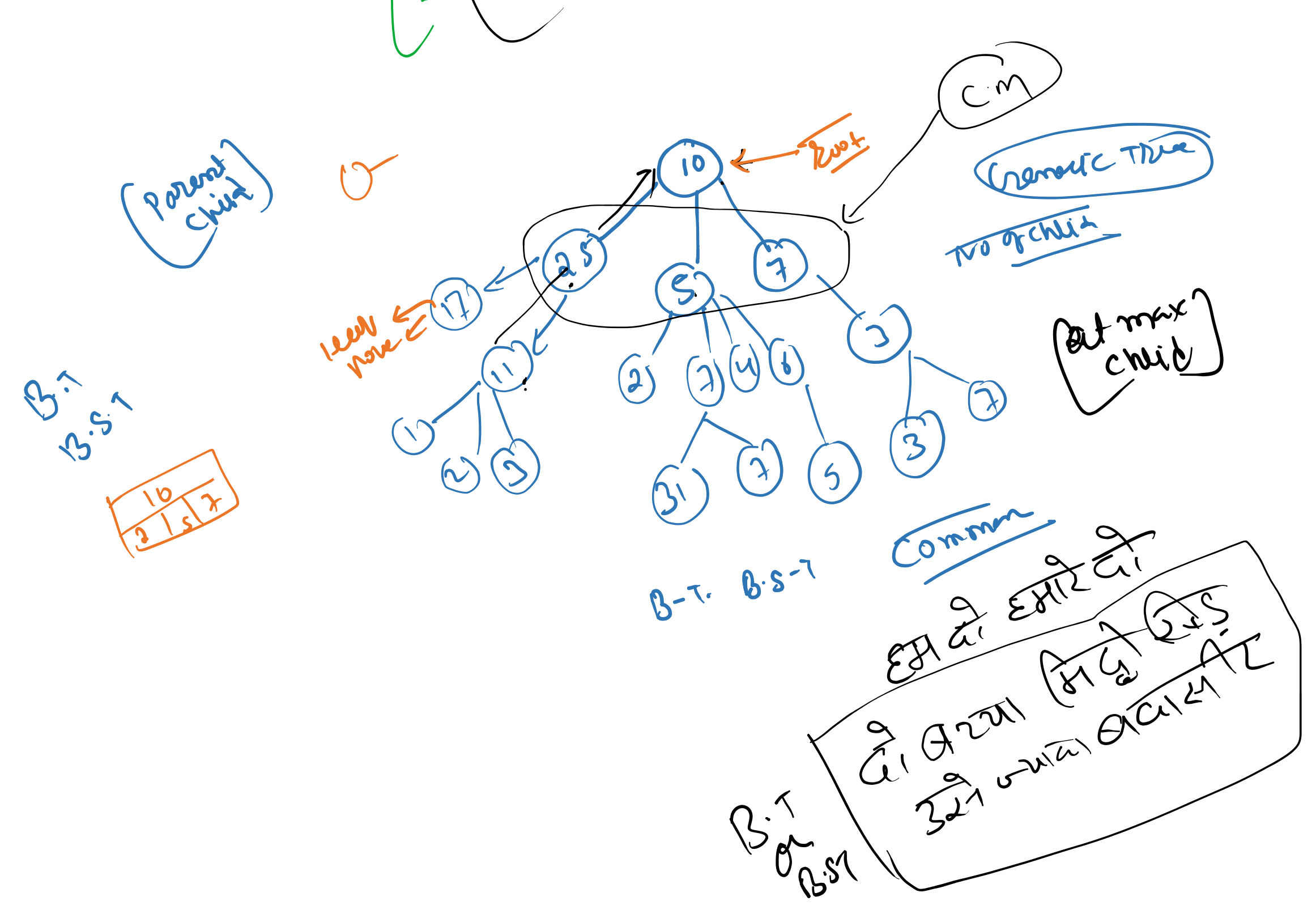
Lec-28

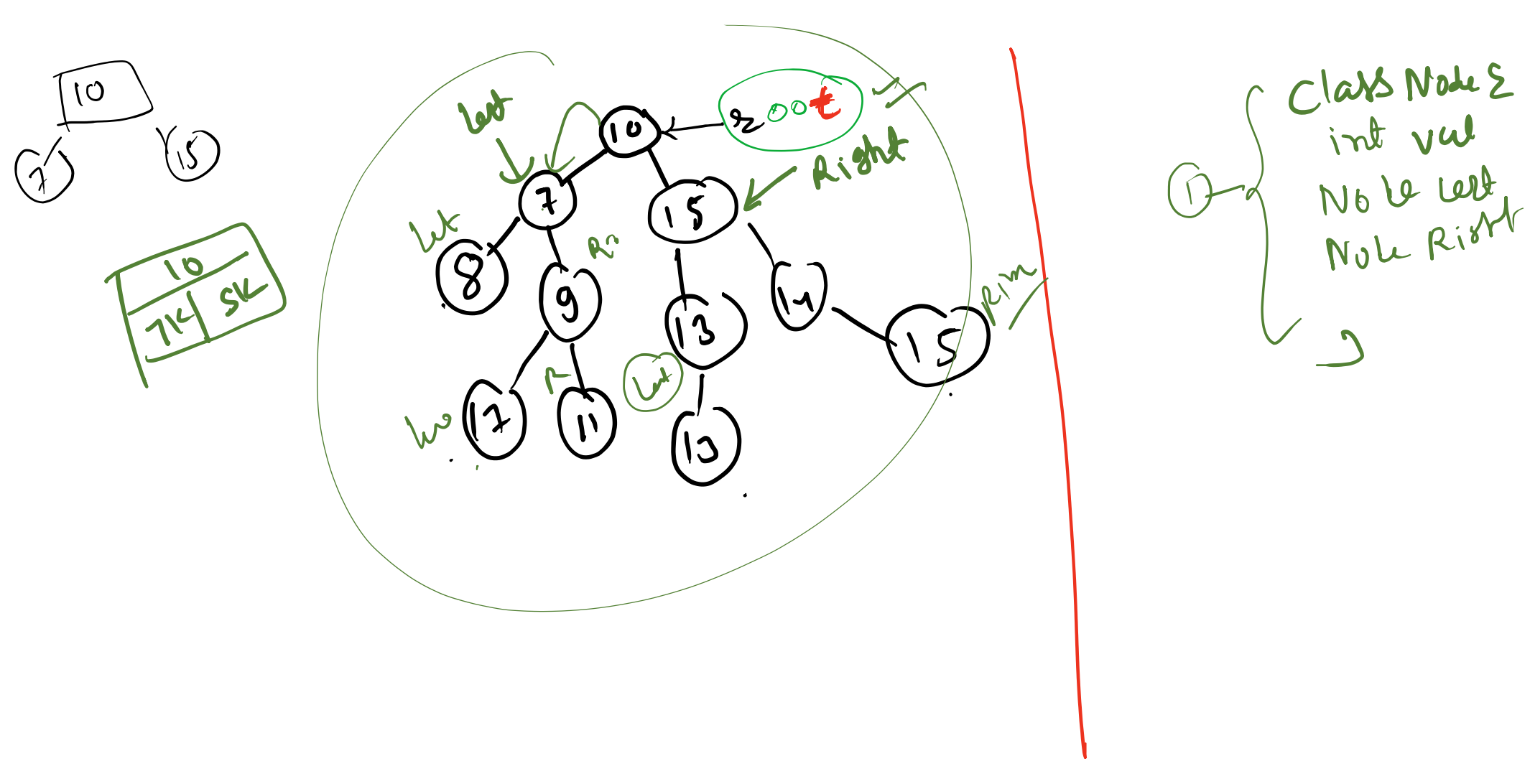


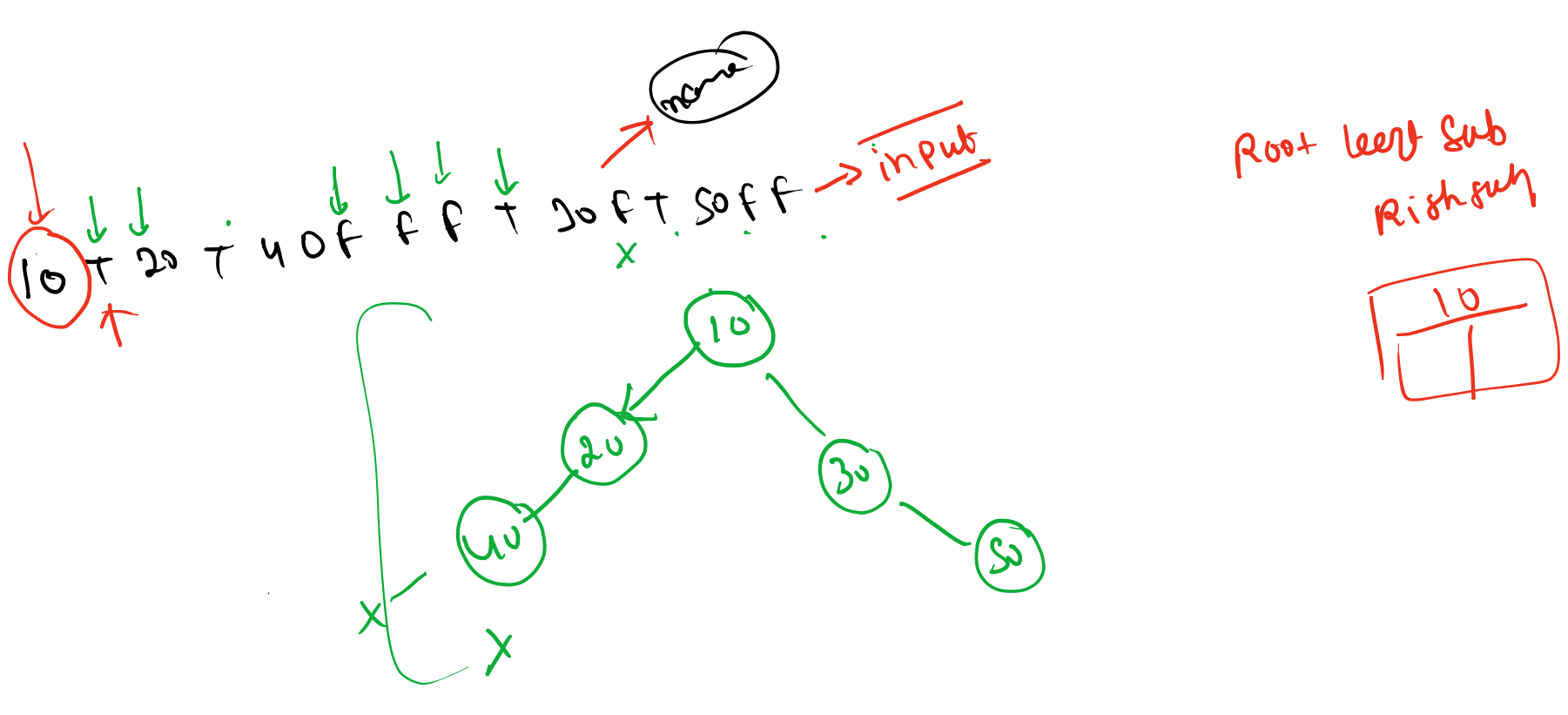




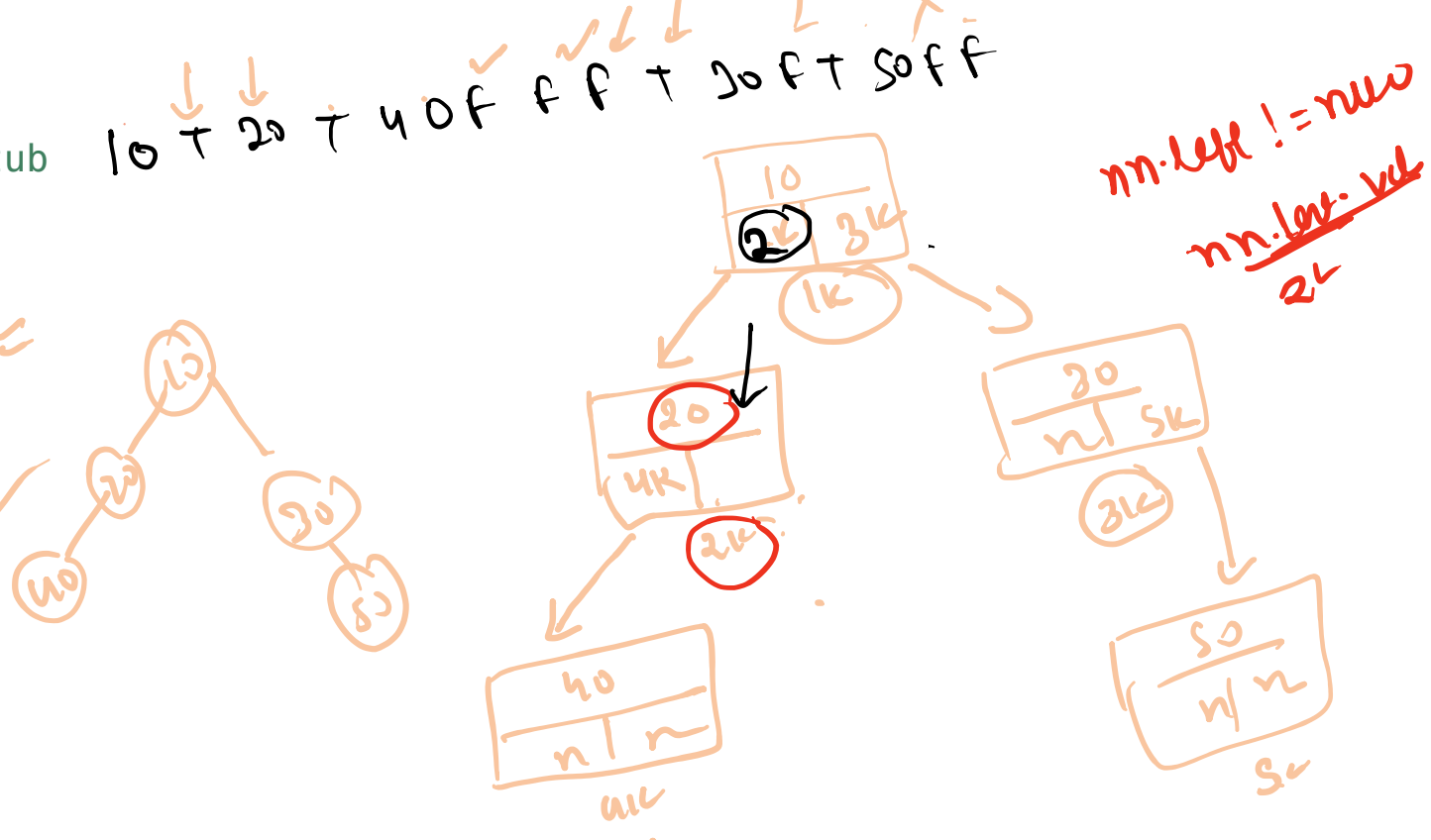






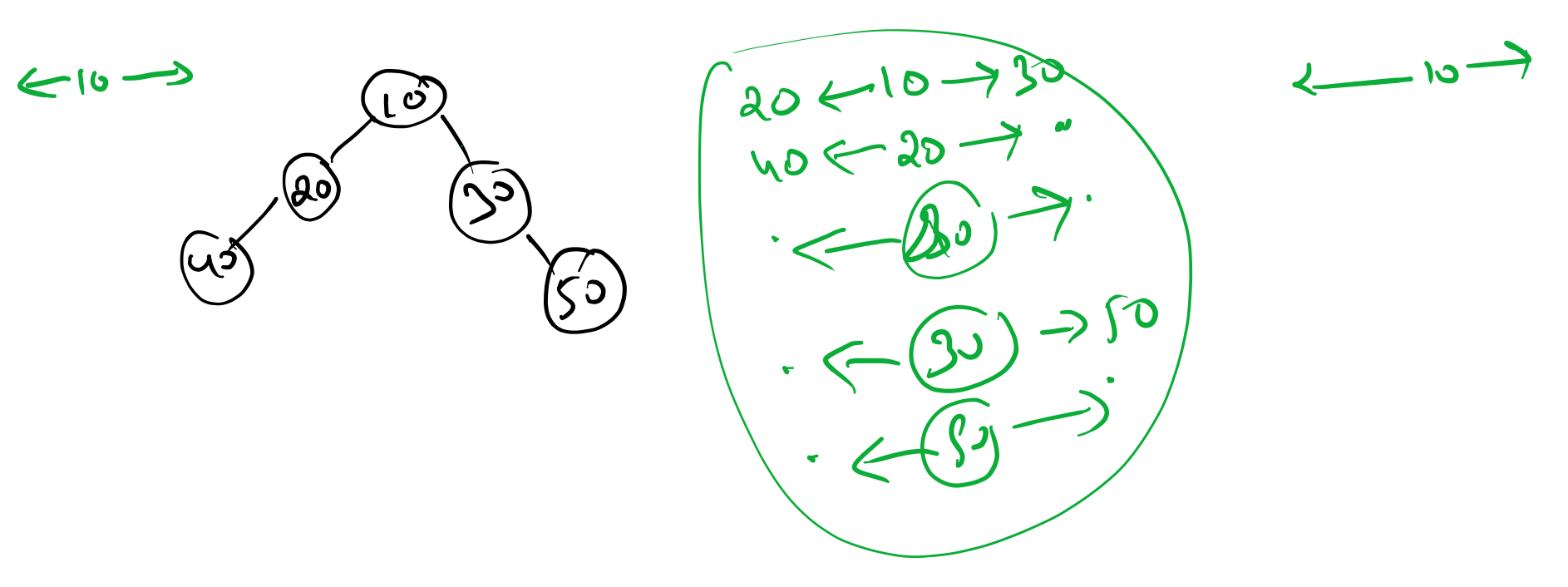


```
private Node CreateTree() {
	// TODO Auto-generated method stub
	int item =sc.nextInt();
	Node nn = new Node();
	nn.val=item;
	boolean hlc =sc.nextBoolean();
	if(hlc) {
		nn.left=CreateTree();
	}
	boolean hrc=sc.nextBoolean();
	if(hrc) {
		nn.right=CreateTree();
	}
	return nn;
}
```

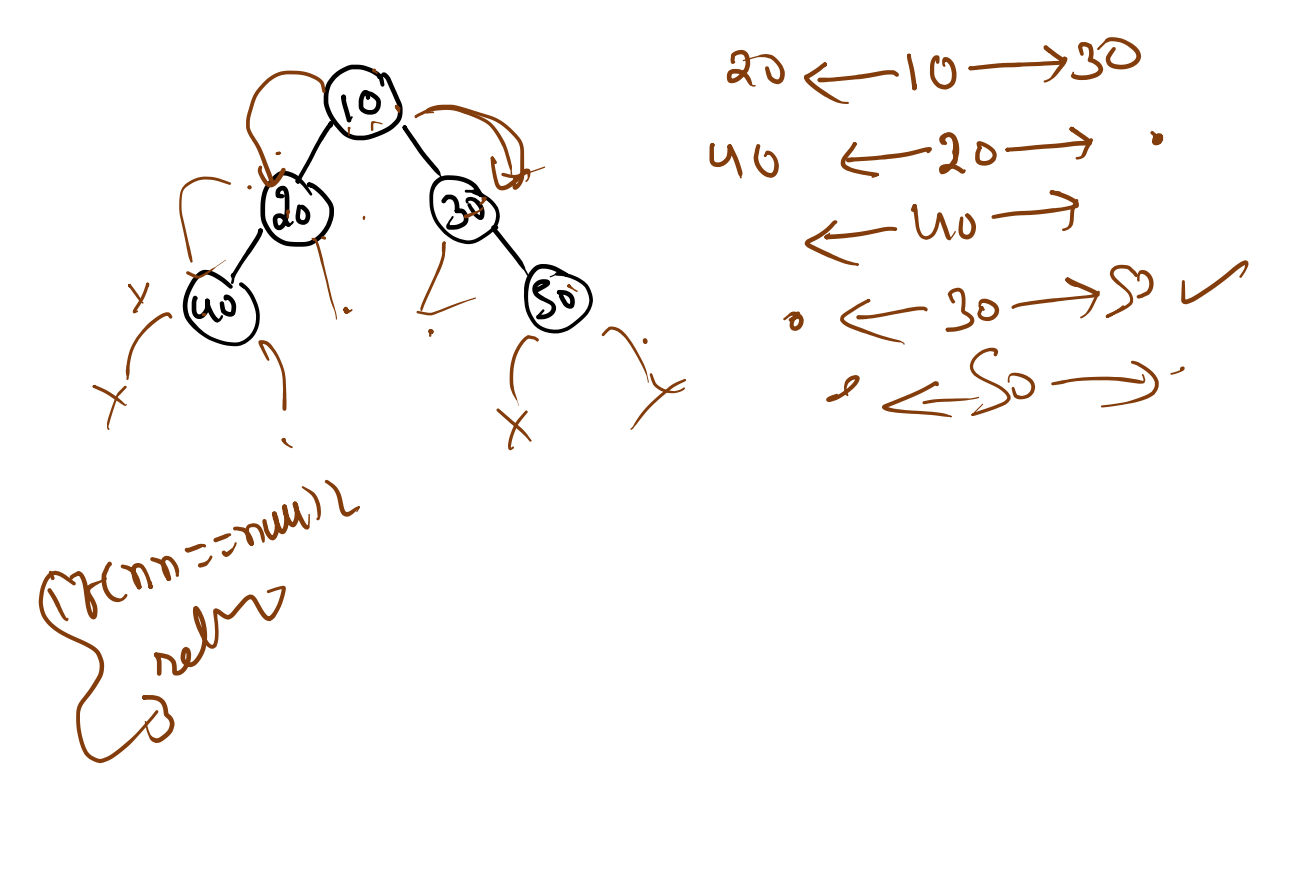


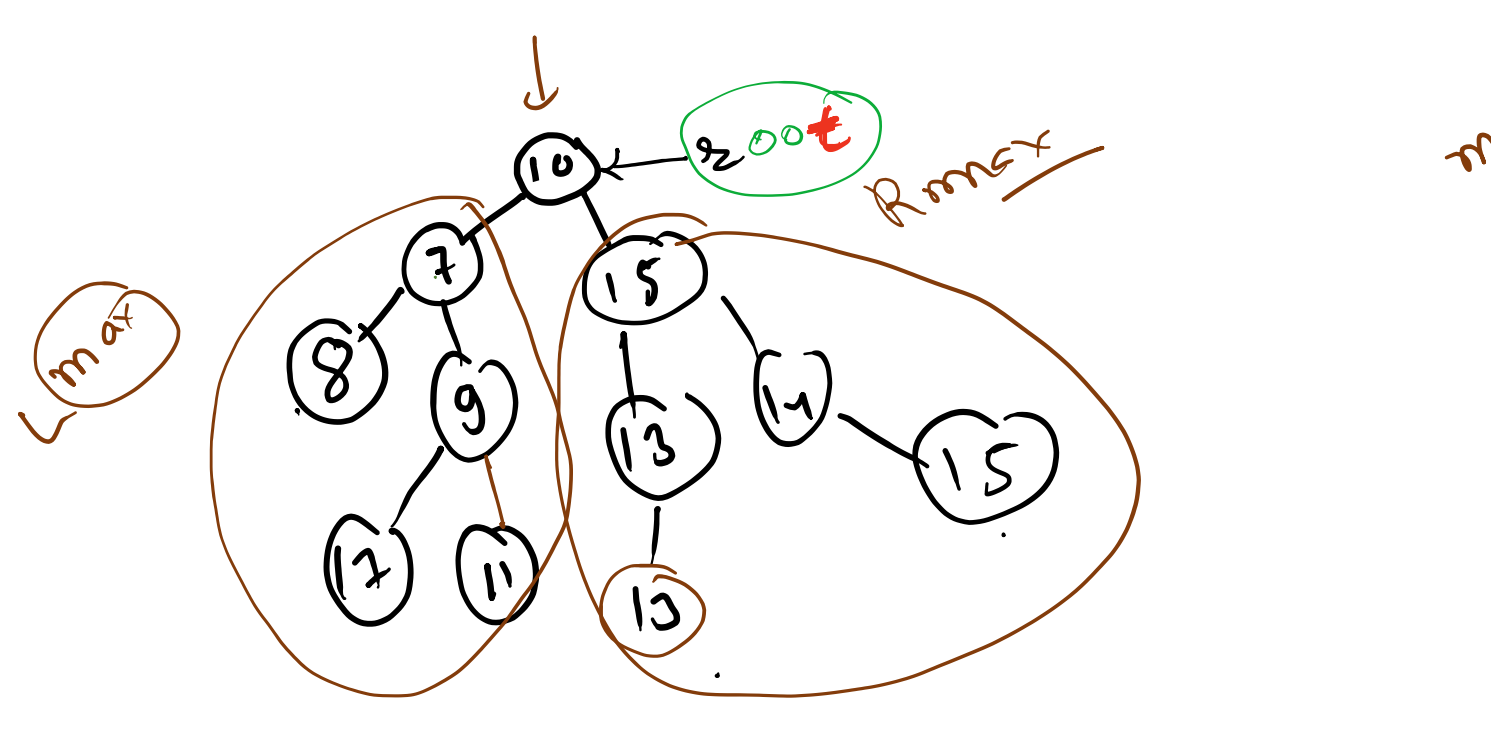
20←10→ 30

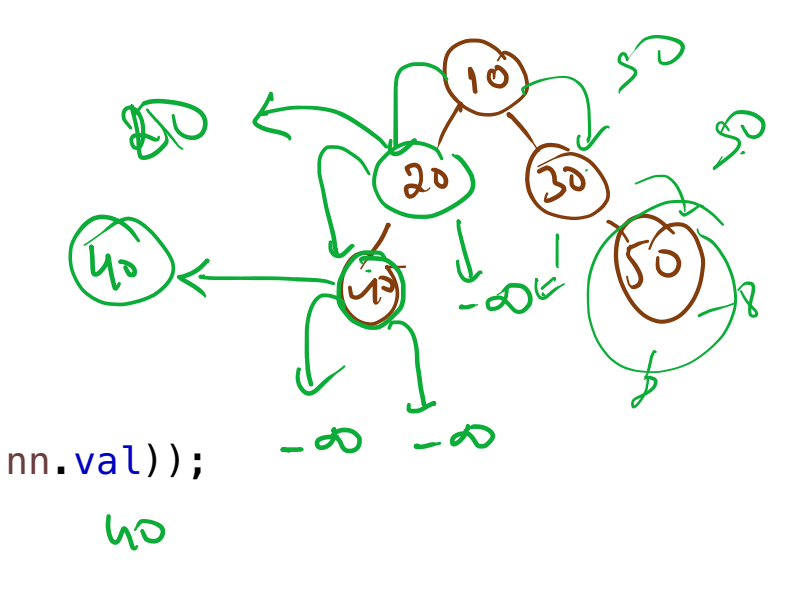
```
public void Display(Node nn) {
	String s = "<--" + nn.val + "-->";
	if(nn.left!=null) {
		s= nn.left.val+s;
	}
	else {
		s="."+s;
	}
	if(nn.right!=null) {
		s=s+nn.right.val;
	}
	else {
		s=s+".";
	}
}
```



```
public void Display(Node nn) {
	String s = "<--" + nn.val + "-->";
	if(nn.left!=null) {
		s= nn.left.val+s;
	}
	else {
		s="."+s;
	}
	if(nn.right!=null) {
		s=s+nn.right.val;
	}
	else {
		s=s+".";
	}
	System.out.println(s);
	Display(nn.left);
	Display(nn.right);

}
```

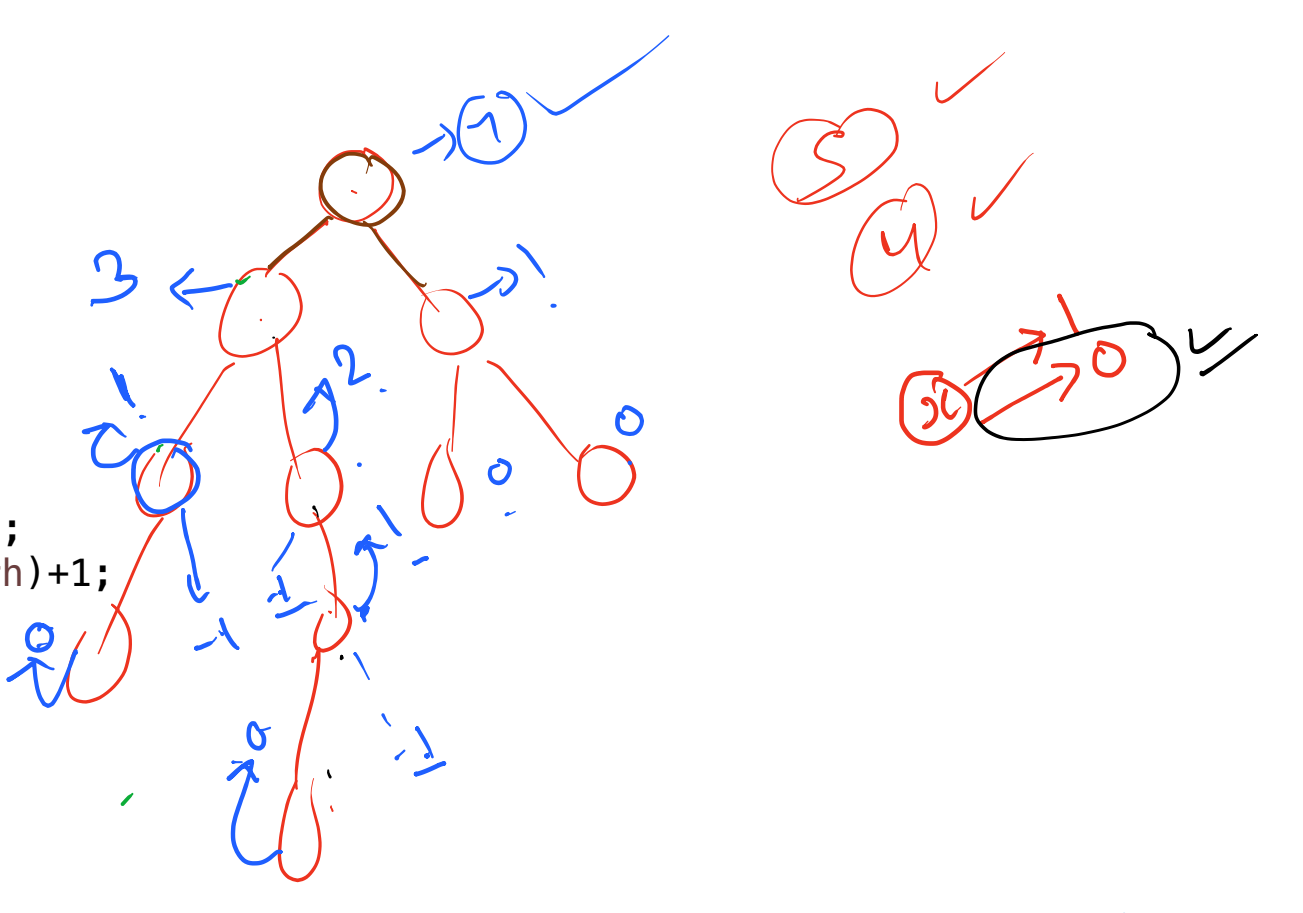


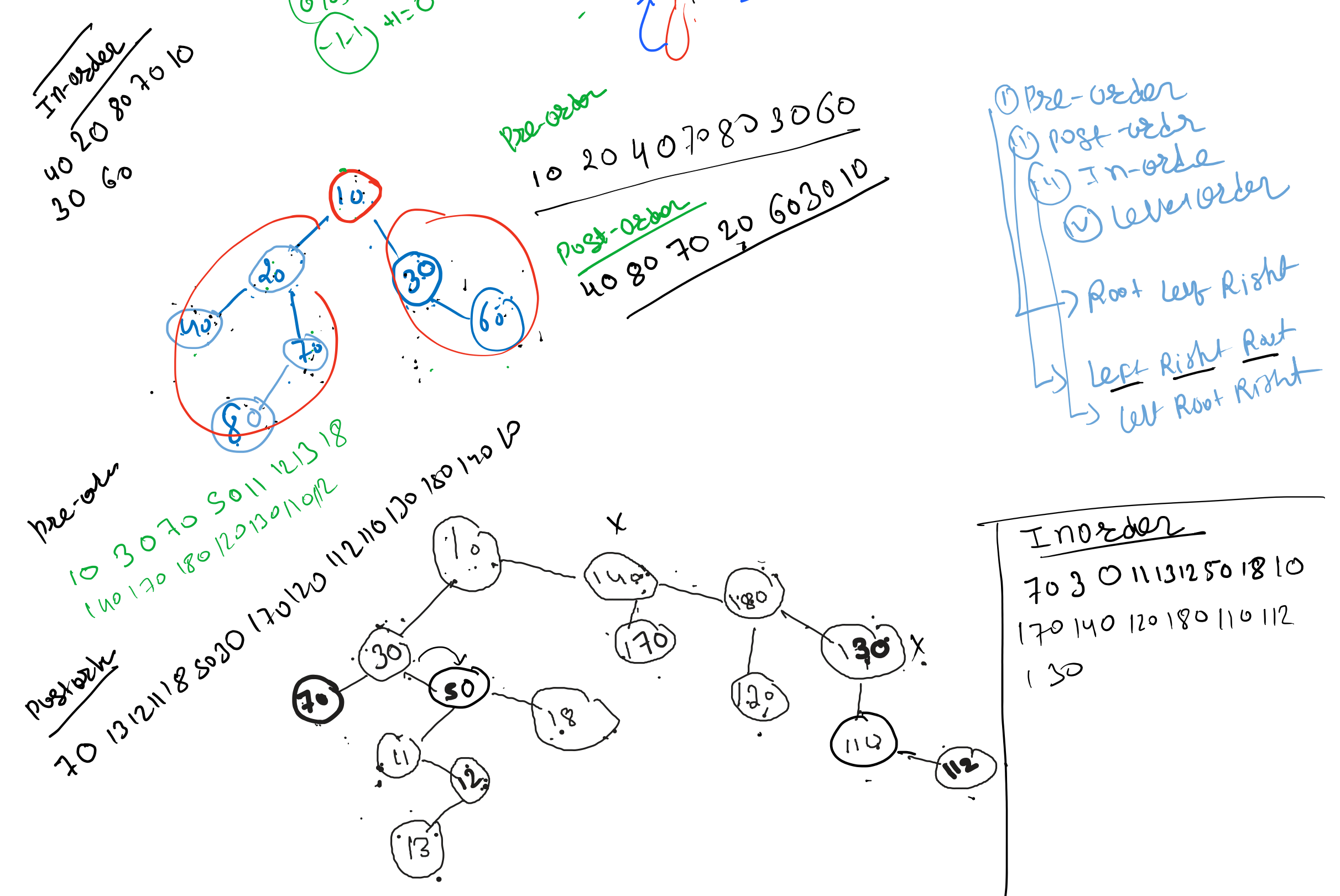


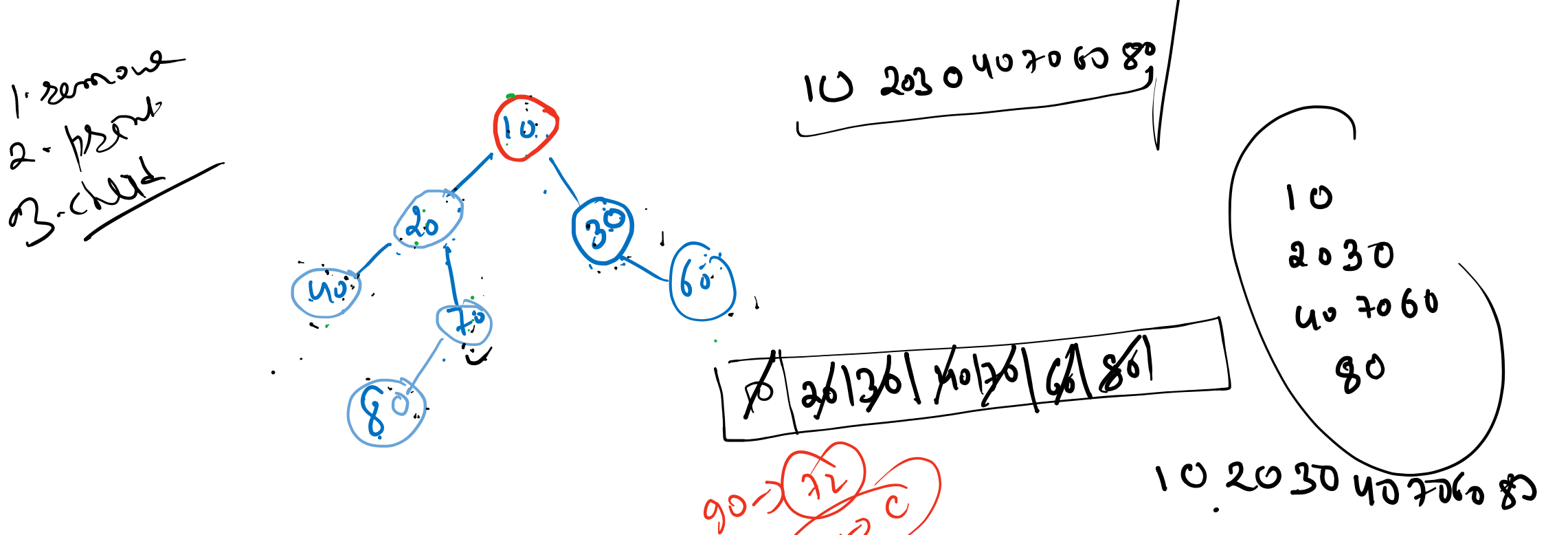


```
private int max(Node nn) {

	int lmax = max(nn.left);
	int rmax = max(nn.right);
	return Math.max(lmax, Math.max(rmax, nn.val));

}
```

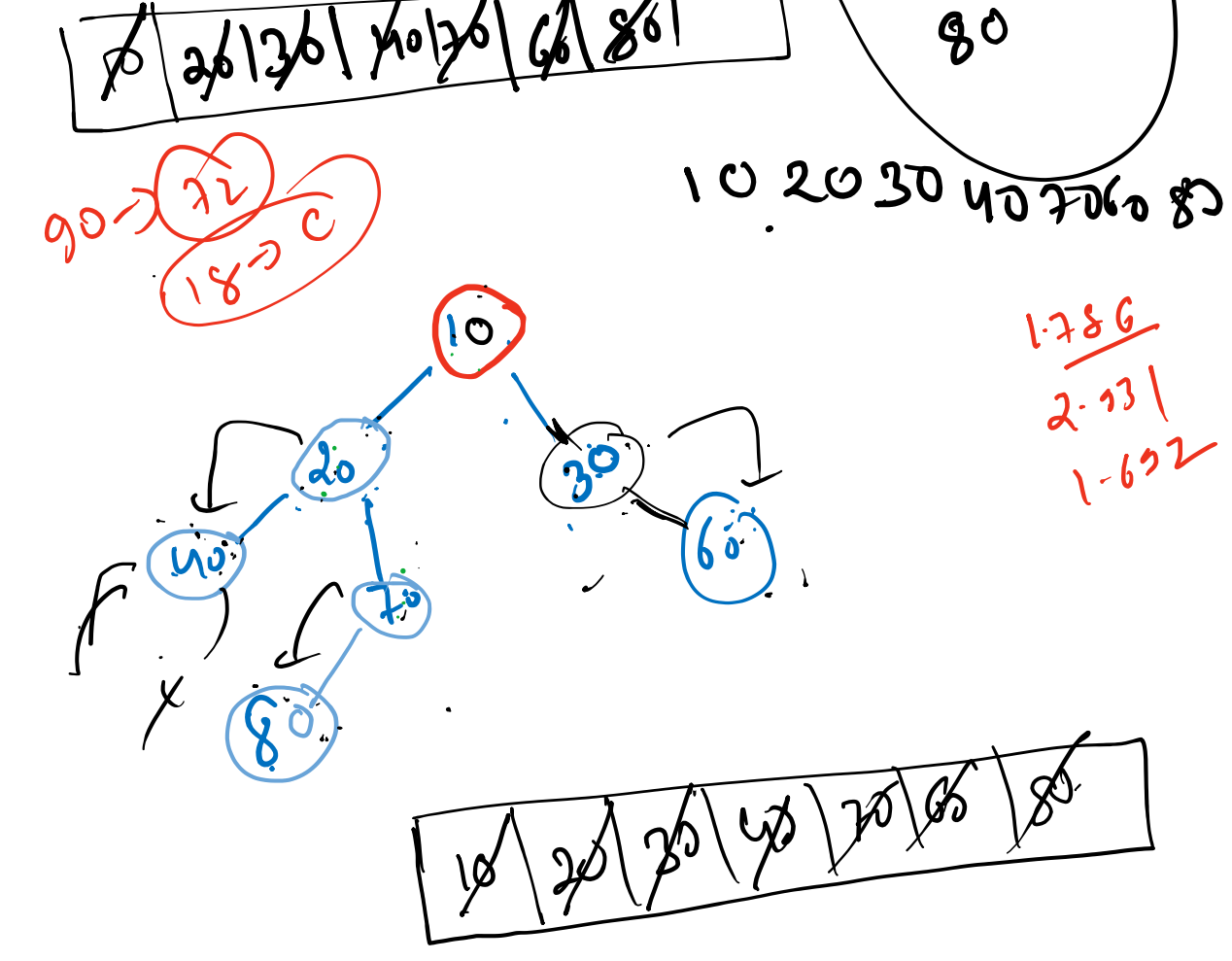
```
private int ht(Node node) {
	if(node==null) {
		return 0;
	}
	int lh=ht(node.left);
	int rh=ht(node.right);
	return Math.max(lh, rh)+1;

}
```







```
public void levelOrder() {
	Queue<Node> q = new LinkedList<>();
	q.add(root);
	while(!q.isEmpty()) {
		Node r=q.poll();
		System.out.print(r.val+" ");
		if(r.left!=null) {
			q.add(r.left);
		}
		if(r.right!=null) {
			q.add(r.right);
		}
	}
	System.out.println();
}
```



10 20 30 40 70 60 80