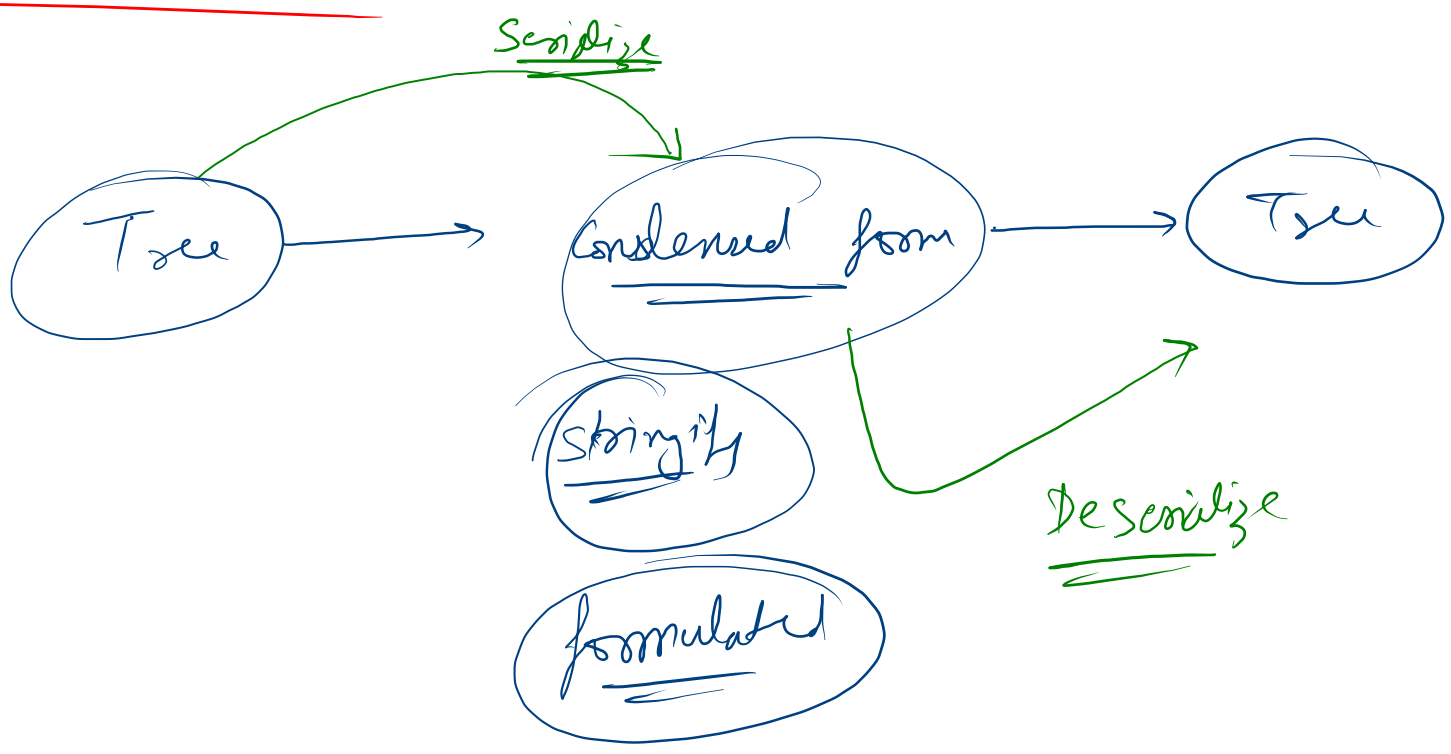
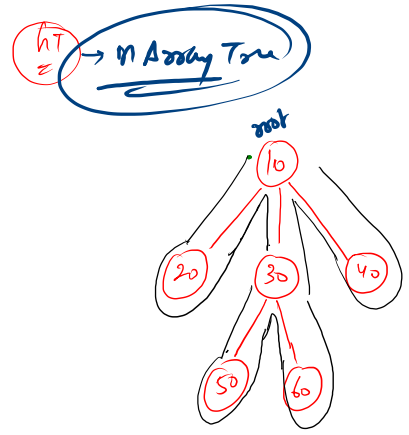
Serialize Deserialize






```java
// Encodes a tree to a single string.
public static String serialize(Node root) {
    StringBuilder sb = new StringBuilder();
    serializeHelper(root,sb);
    return sb.toString();
}

public static void serializeHelper(Node root,StringBuilder sb){
    sb.append(root.val+",");
    for(Node child : root.children){
        serializeHelper(child,sb);
    }
    sb.append("null,");
}
```

"10,20,null,30,50,null,60,null,null,40,null,null,"

"10,20,null,30,50,null,60,null,null,40,null,null,"

S.split(",")

S = "10,20,null,30,50,null,60,null,null,40,null,null,"

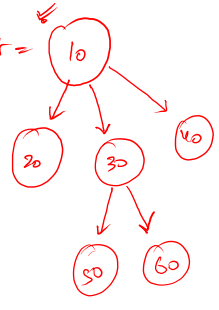
0	1	2	3	4	5	6	7	8	9	10	11
"10"	"20"	"null"	"30"	"50"	"null"	"60"	"null"	"null"	"40"	"null"	"null"

[]arr =>

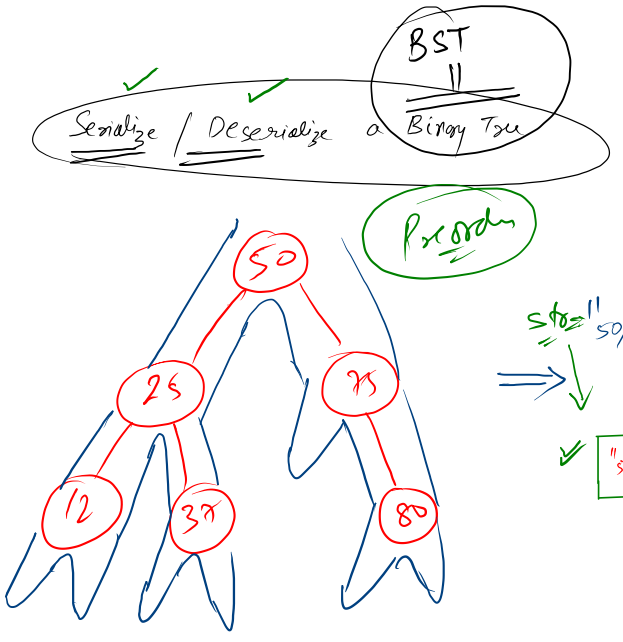
<Node> st =

arr[i] == "null" ret

arr[i].equals("null") => True




```java
public static Node deserialize(String str) {
    String[] arr = str.split(",");

    Stack<Node> st = new Stack<>();
    Node root = new Node(Integer.parseInt(arr[0]));
    st.push(root);
    int idx = 1;
    while(st.size() > 0){
        String vl = arr[idx++];
        if(vl.equals("null")){
            st.pop();
        }else{
            Node newNode = new Node(Integer.parseInt(vl));
            st.peek().children.add(newNode);
            st.push(newNode);
        }
    }
    return root;
}
```

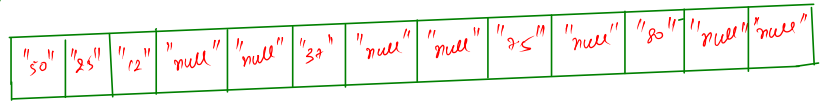


H.W.

str.split(",")

start
0 → left
1 → right
2 → pop()

str = "50, 25, 12, null, null, 37, null, null, 75, null, 80, null, null"



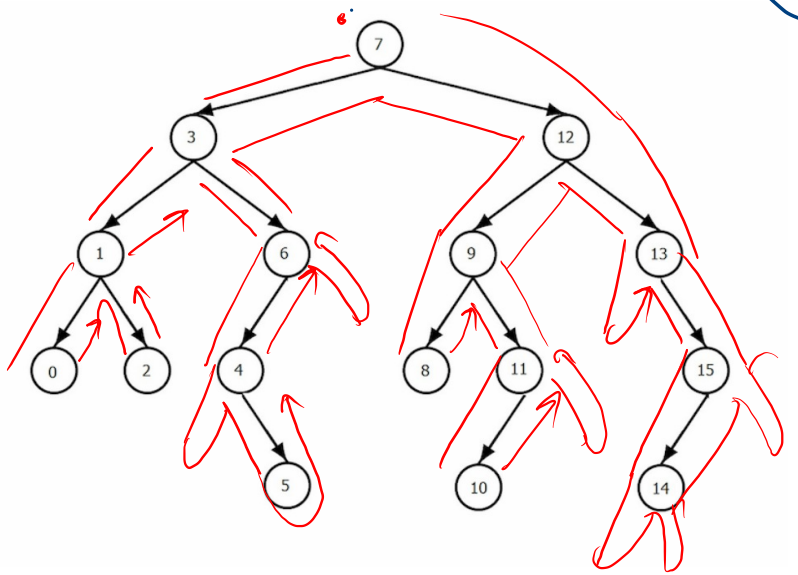
Construct a Binary Tree

Single Child Nodes

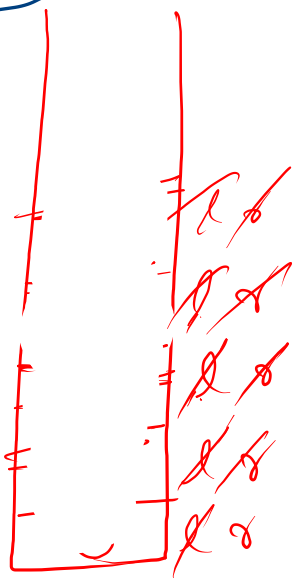
6	4	11	13	15

```
public static ArrayList<Integer> exactlyOneChild(TreeNode root) {
  ArrayList<Integer> ans = new ArrayList<>();
  return ans;
}
```

H.W.



a



b

6	4	11	13	15

P S

```
void exactlyOneChild(TreeNode root, ArrayList<Integer> ans){
if(root == null || (root.left==null && root.right==null))
{
  return;
}
if(root.left ==null || root.right ==null){
    ans.add(root.data);
}
exactlyOneChild(root.left, ans);
exactlyOneChild(root.right, ans);
}
```

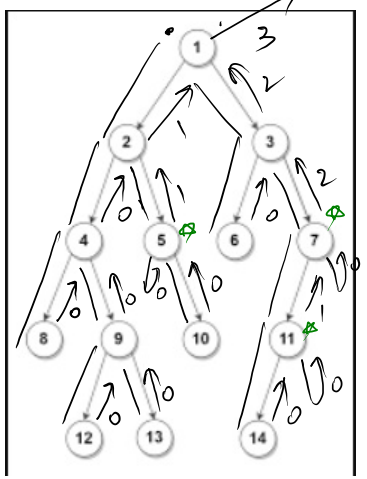
→ 2 child

→ 1 child

→ 0 ch.

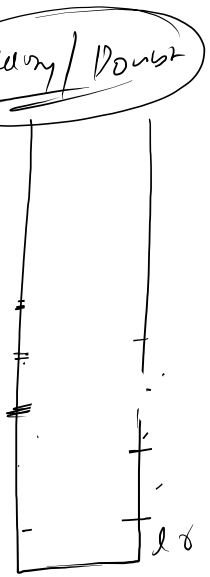
```
static int count = 0;
p s void exactlyOneChild(TreeNode root, ArrayList<Integer> ans){
    if(root == null || (root.left==null && root.right==null))
    {
        return;
    }
    if(root.left==null || root.right==null){
        ans.add(root.data);  count++;
    }
    exactlyOneChild(root.left, ans);
    exactlyOneChild(root.right, ans);
}
```

Single child parent node

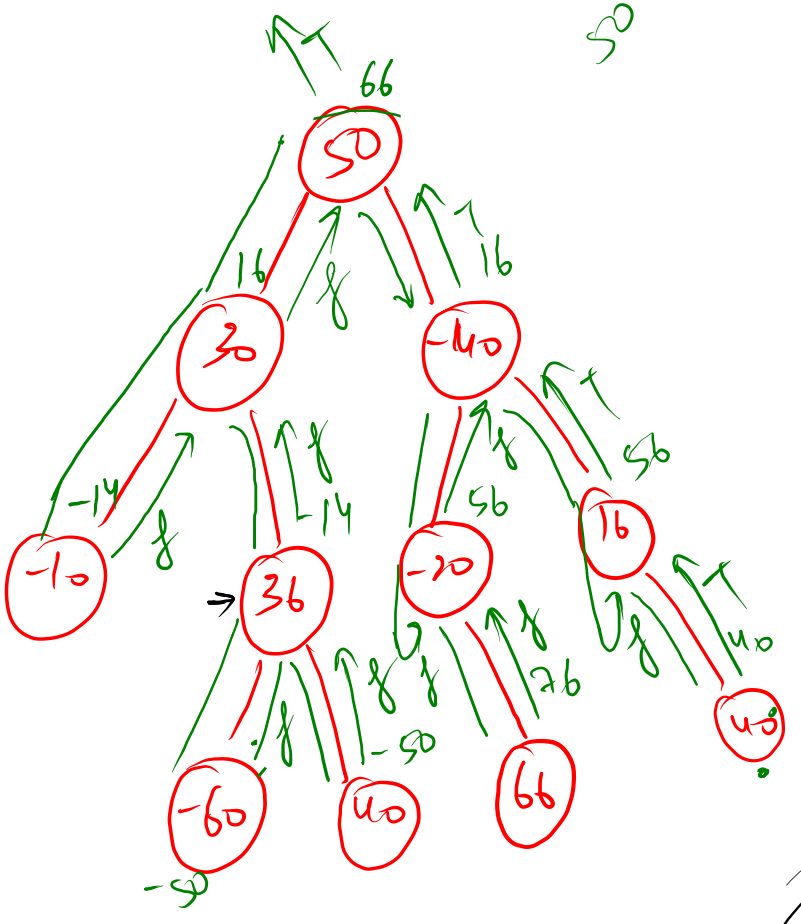


H.W.

Query / Doubt



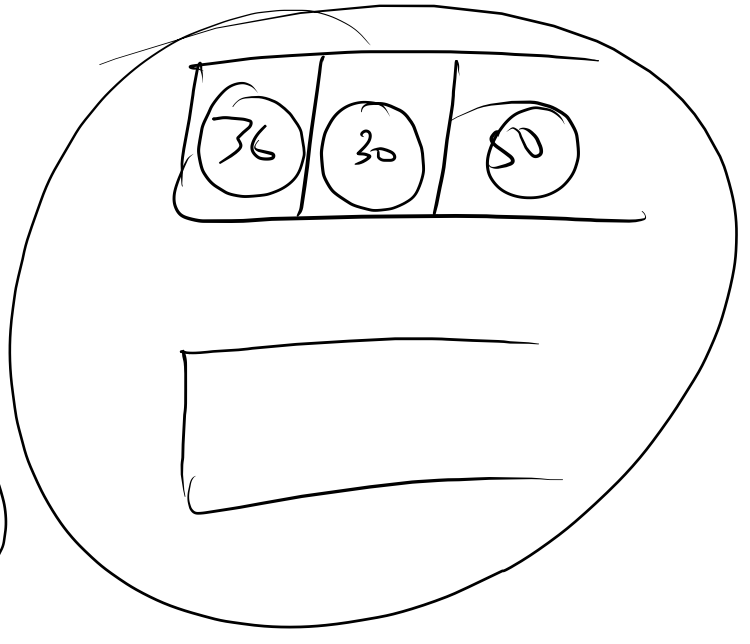
```
p s int solve(TreeNode root){
    if(root==null || (root.left==null && root.right==null)){
        return 0;
    }
    int ans=0;
    ans+=solve(root.left);
    ans+=solve(root.right);
    if(root.left==null || root.right==null){
        ans=ans+1;
    }
    return ans;
}
```

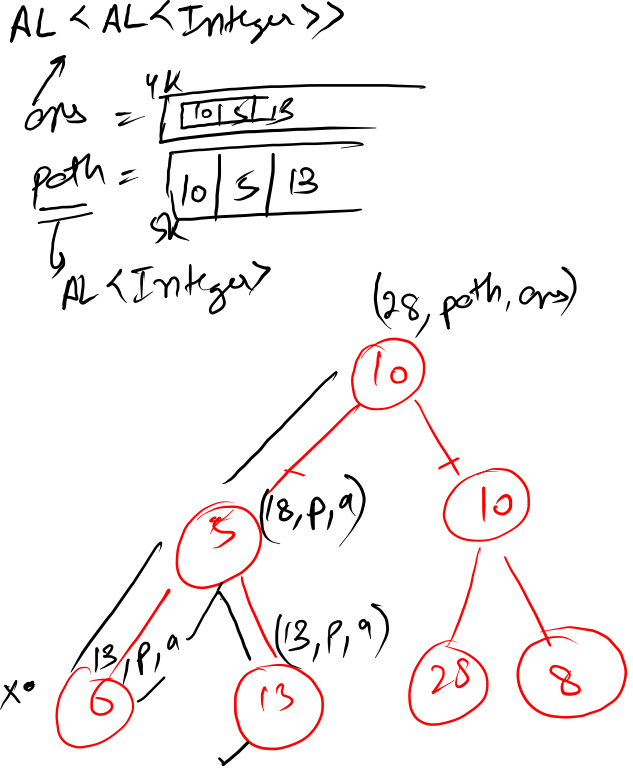


if (leq) {

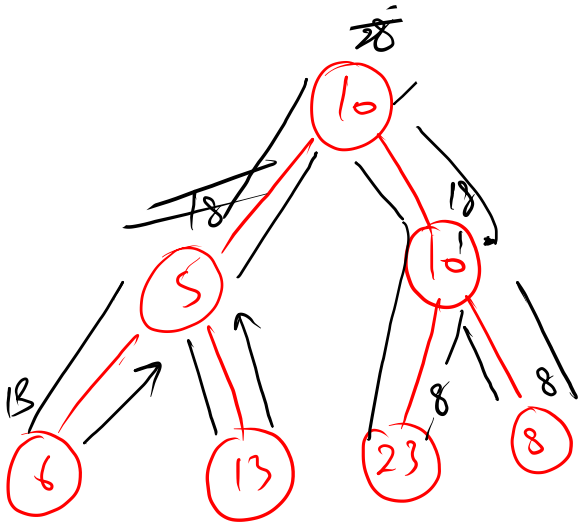
tar - n. val == 0

}

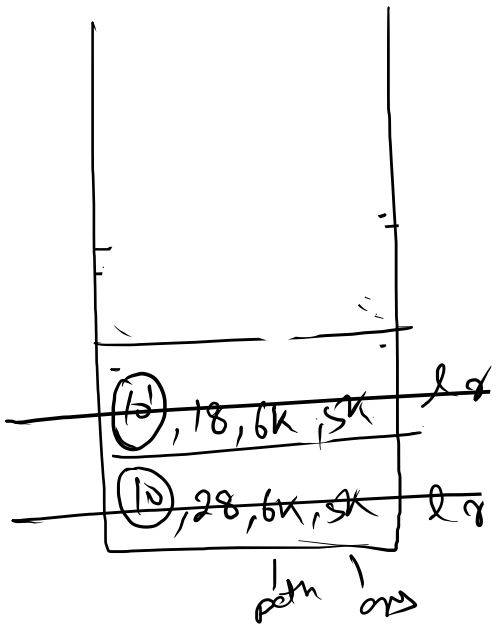


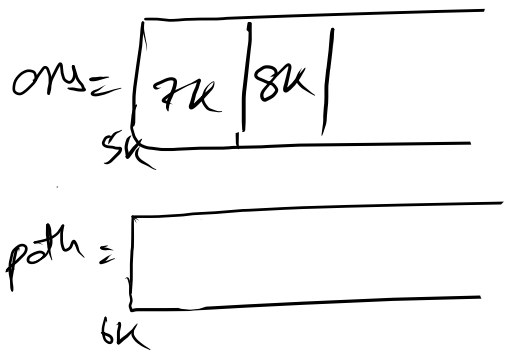


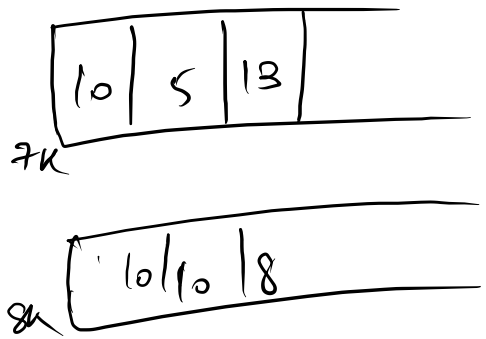

```java
public static boolean hasPathSum(TreeNode root, int targetSum) {
  if(node == null){
      return false;
  }
  if(root.left == null && root.right == null){
      return (targetSum-root.val == 0);
  }
  boolean res = hasPathSum(root.left,targetSum-root.val) || hasPathSum(root.right,targetSum-root.val);
  return res;
}
```




```java
public static void pathSumH(TreeNode root , int targetSum , ArrayList<Integer> path , ArrayList<ArrayList<Integer>> ans){
    if(root == null){
        return;
    }
    if(root.left == null && root.right == null){
        if(targetSum-root.val == 0){
            ArrayList<Integer> tmp = new ArrayList<>();
            for(int val : path){
                tmp.add(val);
            }
            tmp.add(root.val);
            ans.add(tmp);
        }
        return;
    }

    path.add(root.val);

    pathSumH(root.left,targetSum-root.val,path,ans);
    pathSumH(root.right,targetSum-root.val,path,ans);

    path.remove(path.size()-1);
}
public static ArrayList<ArrayList<Integer>> pathSum(TreeNode root, int targetSum) {
    ArrayList<ArrayList<Integer>> ans = new ArrayList<>();
    pathSumH(root,targetSum,new ArrayList<Integer>() , ans);
  return ans;
}
```





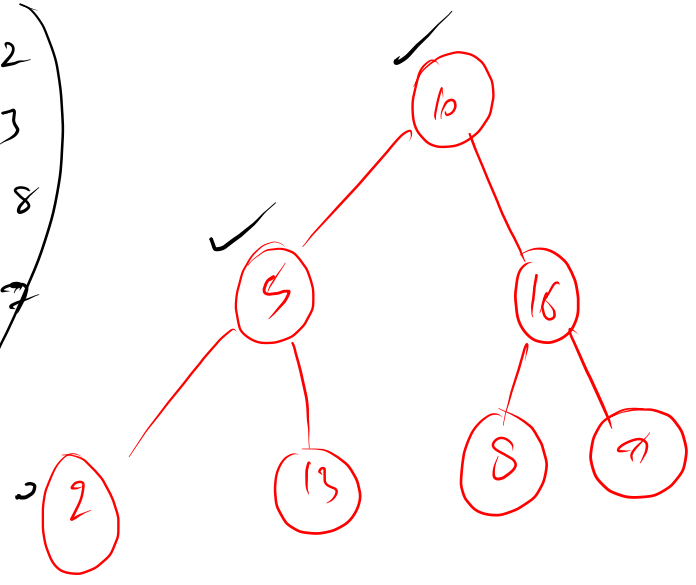


10 → 5 → 2

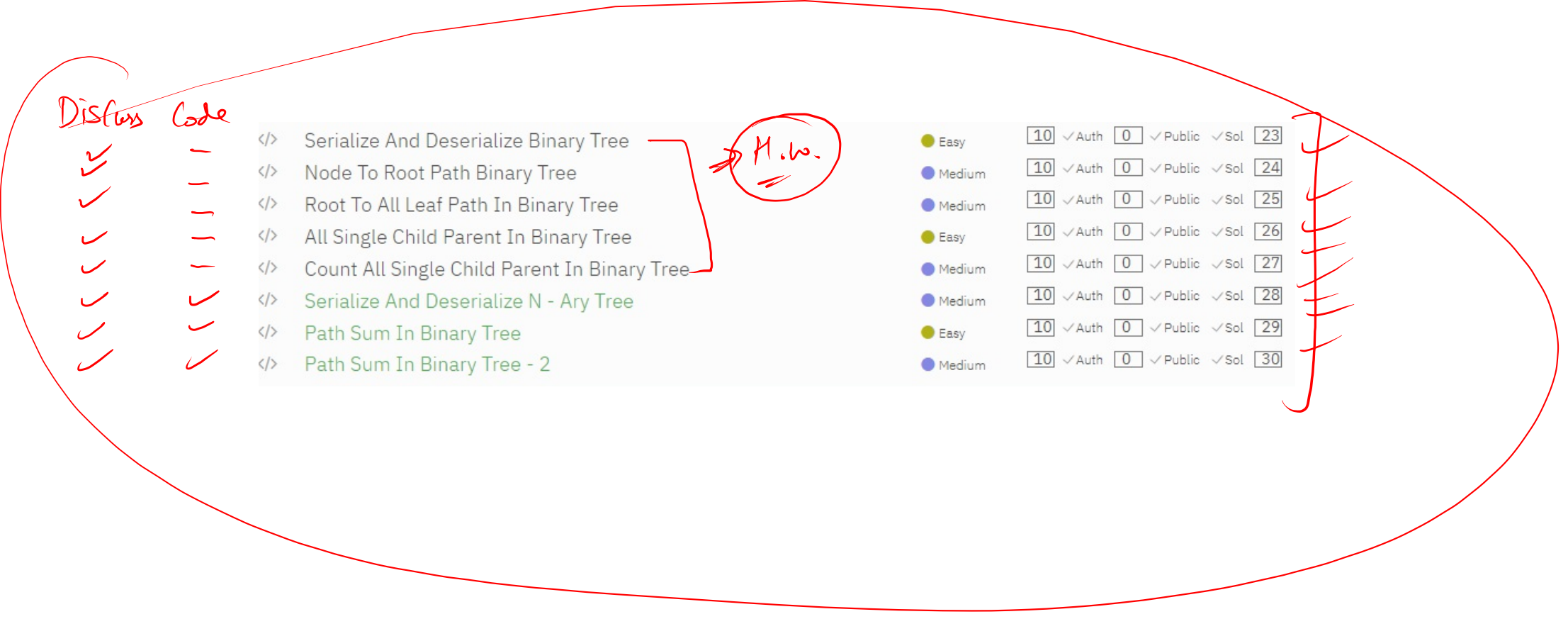
10 → 5 → 13

10 → 16 → 8

10 → 16 → 7




```
p s v solve( root, AL<Int> path, AL<AL<Int>> ans){

}
```


Discuss
Code
Serialize And Deserialize Binary Tree
Node To Root Path Binary Tree
Root To All Leaf Path In Binary Tree
All Single Child Parent In Binary Tree
Count All Single Child Parent In Binary Tree
Serialize And Deserialize N - Ary Tree
Path Sum In Binary Tree
Path Sum In Binary Tree - 2
M.W.
Easy
Medium
Medium
Easy
Medium
Medium
Easy
Medium
10 Auth 0 Public Sol 23
10 Auth 0 Public Sol 24
10 Auth 0 Public Sol 25
10 Auth 0 Public Sol 26
10 Auth 0 Public Sol 27
10 Auth 0 Public Sol 28
10 Auth 0 Public Sol 29
10 Auth 0 Public Sol 30