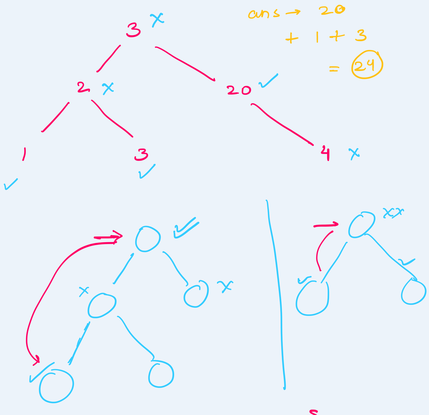
# DP on trees ..

Define functions for nodes of the trees, which we calculate recursively based on children of a nodes.

One of the states in our DP usually is a node i, denoting that we are solving for the subtree of node (i).

## Q House robber III





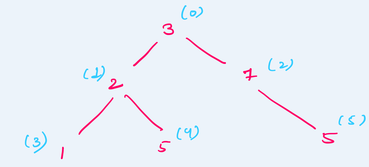

Two choices:

left = 5 ——— 3

```
rob = node.val + helper(node.left, parent_robbed = true)
                + helper(node.right, parent_robbed = true)
not_rob = helper(node.left, parent_robbed = false)
        + helper(node.right, parent_robbed = false)
```

dp_rob

dp_not_rob




```
tree = [3,2,7,1,5,4]
graph = {0:[1,2]
         1:[3,4]
         2:[5]
         3:[]
         4:[]
         5:[]}
```

$$\begin{bmatrix} dp\_rob_i = tree_i + \sum_{child} dp\_not\_rob_{child} \\ dp\_not\_rob_i = \sum_{child} \max(dp\_rob_{child}, dp\_not\_rob_{child}) \end{bmatrix}$$

## Q Binary tree cameras

solve( node ) — child in state 1

(1) All nodes below a 1st node is covered, except for curr node.

child state 1/2

(2) All nodes below 2 including this node are covered, but there is no camera here

(3) All nodes below 2 including this node are covered, and there is a camera here

## Q Maximum Product of Splitted Binary Tree

n-1 edges, n nodes, n-1 diff possible ways