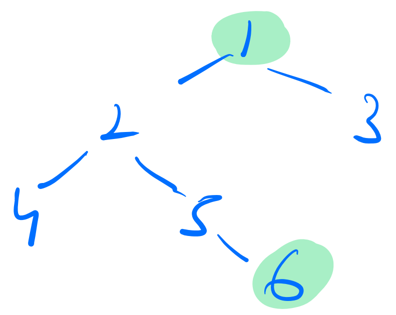
DP on trees

We define functions for nodes of trees, which we recursively calculate based on children of a nodes.
One of the states in the DP is node i, denoting we are solving for subtree of node i.

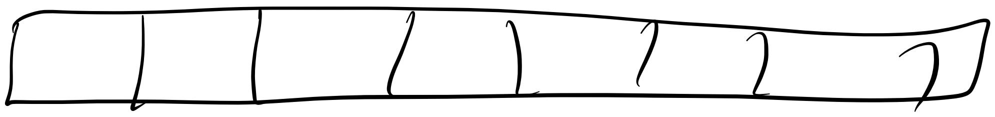
Lets see some examples

Q1 Given a tree, N nodes 1-N.
Each node has C_i coins. Choose a set of nodes such that
1) No 2 nodes are adjacent
2) Sum of coins is max
Return this max sum of coins.



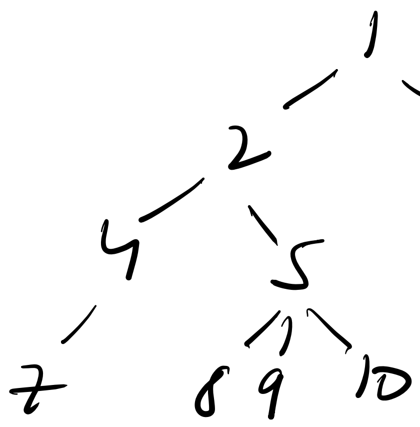
If parent is taken $\Rightarrow$
cannot take a child

Have we seen something like this before



House Robber

$dp(i) \Rightarrow \max(dp(i-1),\ coins[i] + dp(i-2))$



dp[node][can_this_be_taken]

2 scenarios

you take 1

coins [1]
$+ \sum dp[child][false]$

you dont take 1

$\sum dp[child][true]$

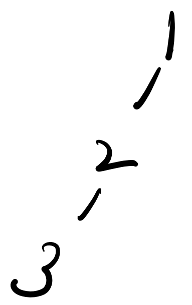
Generalise

$dp(node, true) \rightarrow coins[i] + \sum dp[child][false]$
$\quad\quad\quad\quad\quad\quad \rightarrow \sum dp[child][true]$

$dp(node, false) \rightarrow \sum dp[child][true]$

$$\begin{tikzcd}
 & & node \arrow[dll, dash] \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] \arrow[drr, dash] & & \\
c_1 & c_2 & c_3 & c_4 & c_5
\end{tikzcd}$$

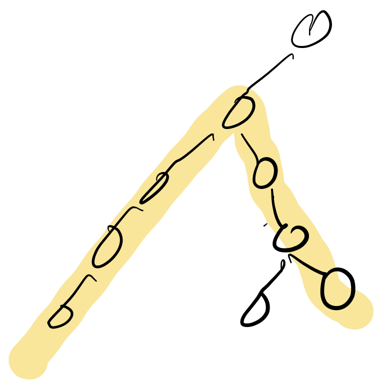
```
int maxCoins (vector <int> adj [N], int coins[]){
    int dp [N][2]
    // i,0 → false        i,1 → true
    ans = dfs (1, 1)
    return ans.
}

int dfs ( int node, par, int can_take ) {
    if( dp[node][can_take] in cache)
        return dp[node][can_take]

    ans = 0
    if ( can_take == 0 ) {
        for( j = 0; j < adj[i].size; j++ ) {
            cur_child = adj[i][j]
            if( cur_child != par)
                ans += dfs ( cur_child, 1)
        }
    }
else {
    ans1 = coins [node]
    for (j = 0; j < adj [i].size; j++) {
        cur_child = adj [i] [j]
        ans1 += dfs ( cur_child, 0 )
    }

    ans2 = 0
    for (j = 0; j < adj [i].size; j++) {
        cur_child = adj [i] [j]
        ans2 += dfs ( cur_child, 1 )
    }

    ans = max ( ans1 , ans2 )
}
dp [node] [can_take] = ans
return  ans

}
```



adj(1)	2
adj(2)	1 3
adj(3)	2

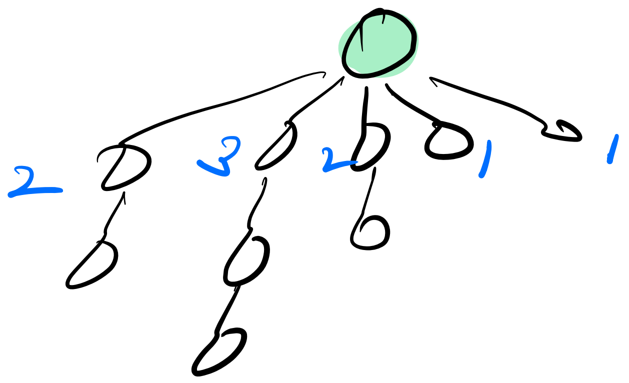
Q2 Given a tree, find the diameter

Diameter ⇒ Longest path b/w any 2 nodes.



When standing at node i ⇒

- Longest path starts at i & goes into subtree $f(i)$
- Longest path passing through i $g(i)$

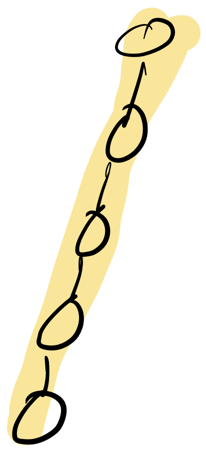


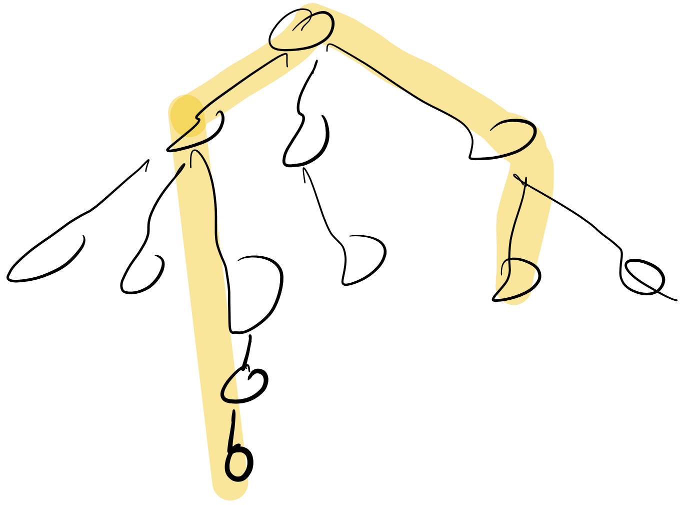
$f(i) = 1 + \max(f(\text{child}))$

$g(i) = 1 + \text{sum of 2 biggest in } \{f_{c_1}, f_{c_2}, f_{c_3} \ldots\}$

$f(i) \Rightarrow$ One arm maximum

$g(i) \Rightarrow$ Two arm maximum.





Diameter $\Rightarrow$ max(all f values &&
all g values)

node

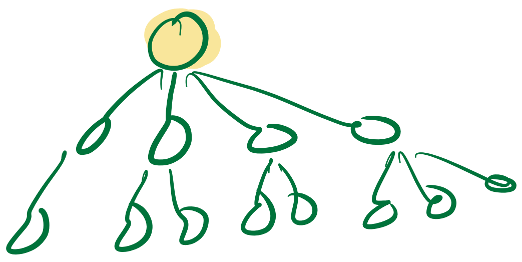
2 biggest f values.

f_values in a arraylist
sort this arraylist
get the last & second last

Q3 Given a tree, find no of diff subtrees

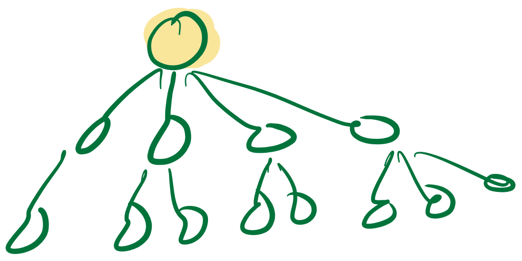
S(node) ⇒ Subtree rooted at node

f(node) ⇒ no of subtrees of S(node) which includes node



$$f(\text{node}) = \underset{\text{mult}}{\prod} \left(1 + f(\text{child})\right)$$

g(node) = no of subtrees not rooted at node



$$g(i) = \sum f(\text{child}) + g(\text{child})$$

$$\text{ans} = f(1) + g(1)$$

{done}