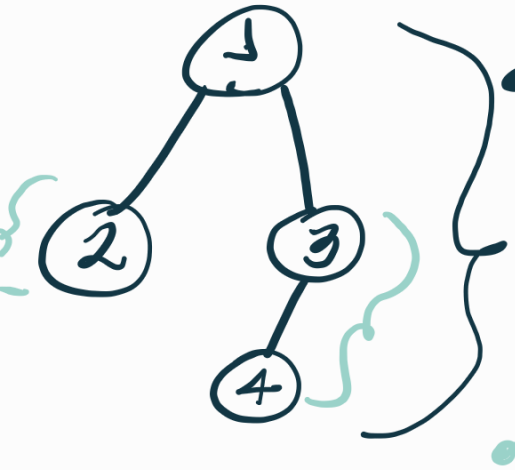
⇒ An tree is considered to be balanced if and only if, all its subtrees at left and right differ in height by at max 1.



- Main subtree max height 3
  L → L and R differ in 1 (2|3)

- Secondary subtrees also differ in 1 (1,2)

→ At each recursion i have to account that each new node has to be seen as also a new subtree

→ Traverse count function has to be kept simple, only for traversing and counting Nodes

→ Upper Funciton "isBalanced" should be the one applying B. Rules to it

↳ if only one subtree comparation differs in more than 1 in height (count) return false already

↳ Otherwise apply "balance" recursion on both sides of actual node

→ by calling isbalanced for each side you assure that all subtrees are being calculated evenly.

L> Ramification of this flow occurs in && usage

Important:

when (traversing + Counting) and applying logic at each subleve of a tree, you will need at least 2 points of recursion in code

* 1st for (traverse + Count)

* 2nd to apply the logic at each level
  - in this scenario if current level is not balanced it end the flow
    - Otherwise it goes deeper on each side until the end.