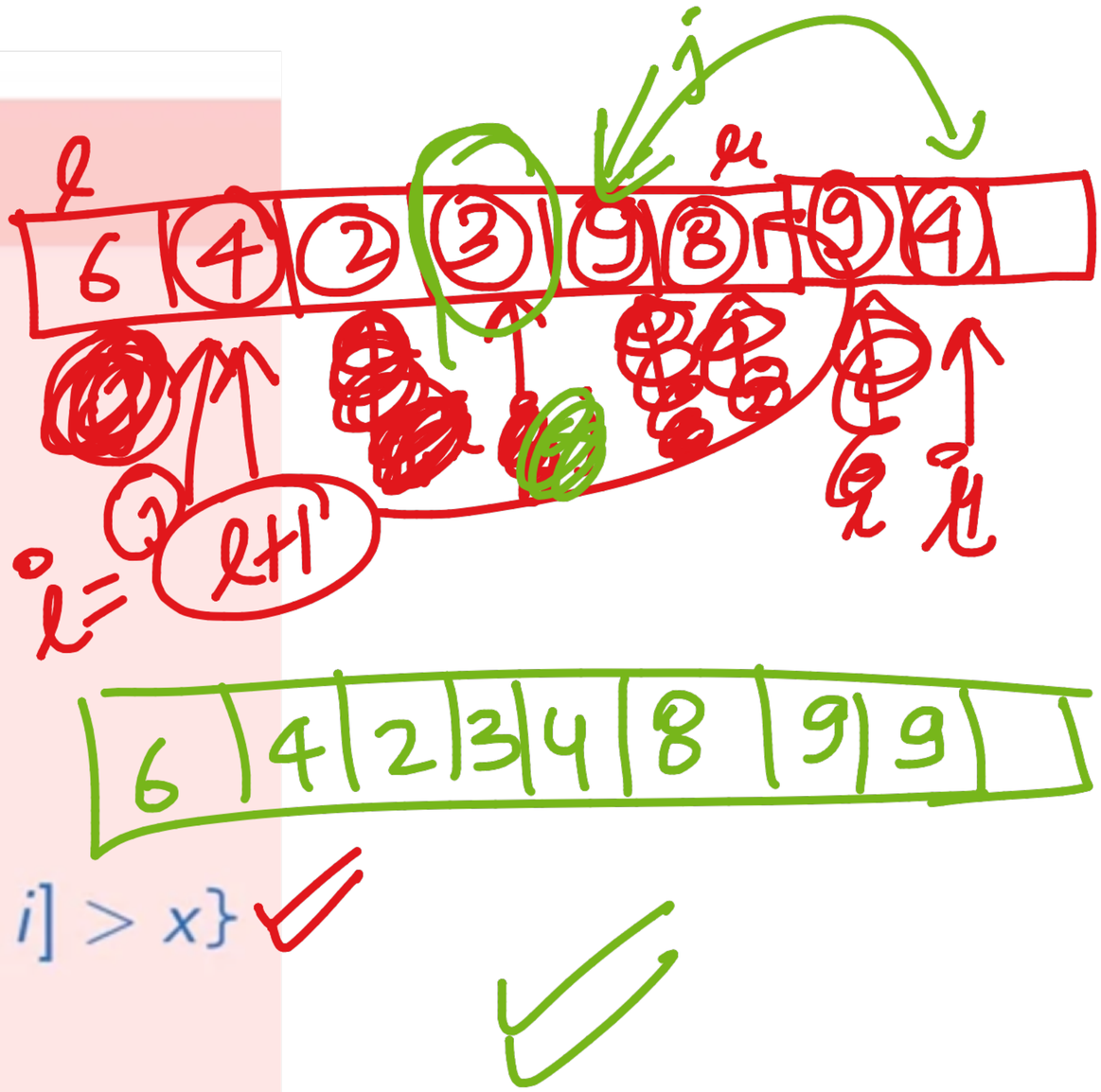
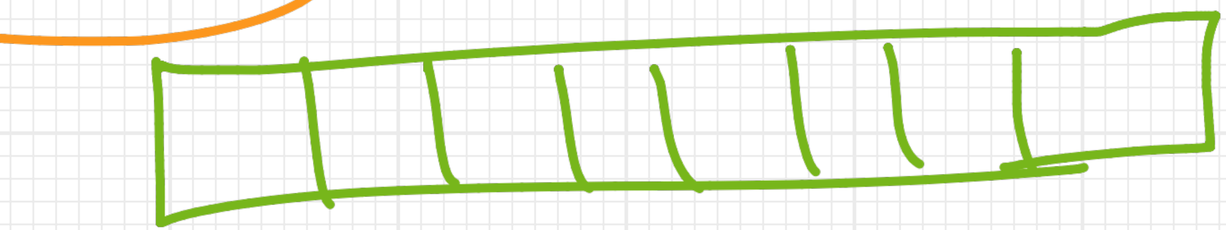
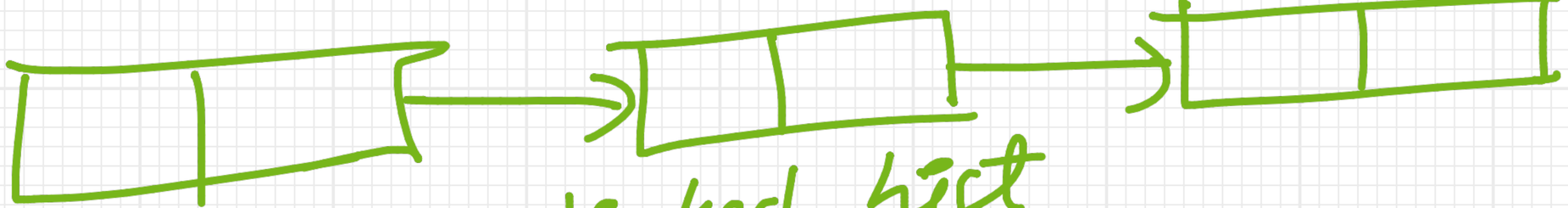
```
Partition(A, ℓ, r)
x ← A[ℓ]                {pivot}
j ← ℓ
for i from ℓ + 1 to r:
    if A[i] ≤ x:
        j ← j + 1
        swap A[j] and A[i]
    {A[ℓ + 1 . . . j] ≤ x, A[j + 1 . . . i] > x}
swap A[ℓ] and A[j]
return j
```



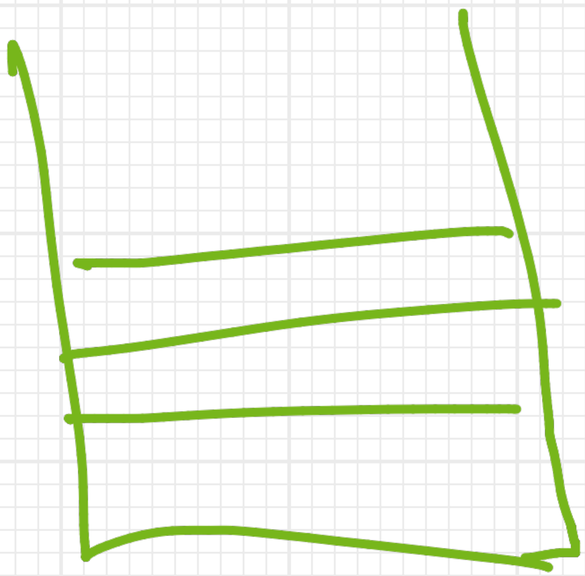
Linear DS



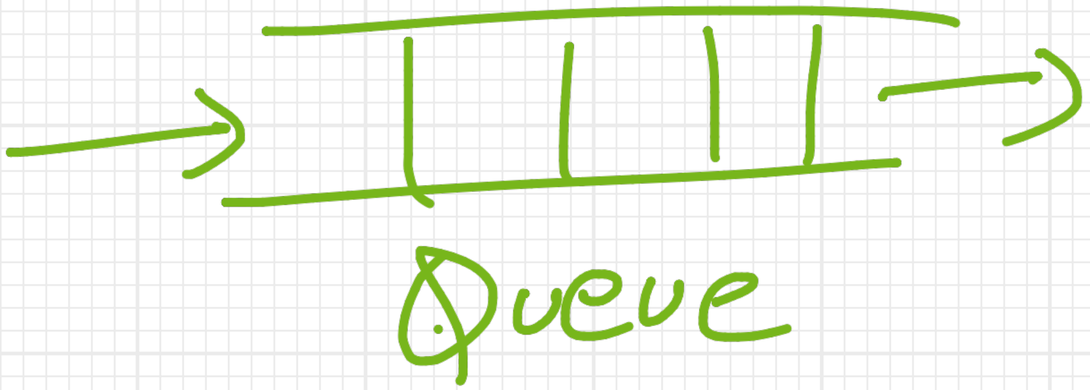
Array



Linked list



Stack



Queue

Non-Linear DS

Lingo

Language

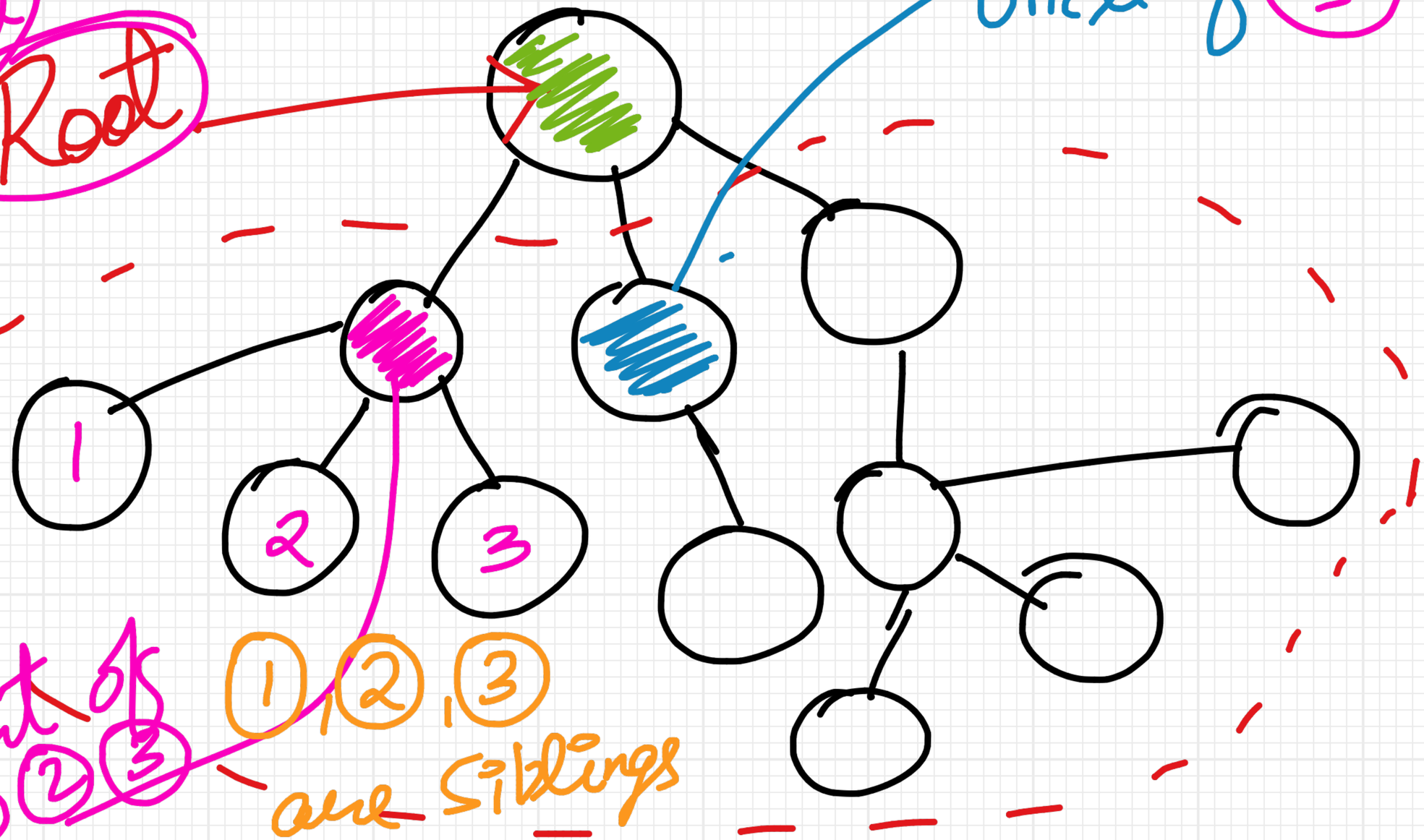
Root

Child of Root

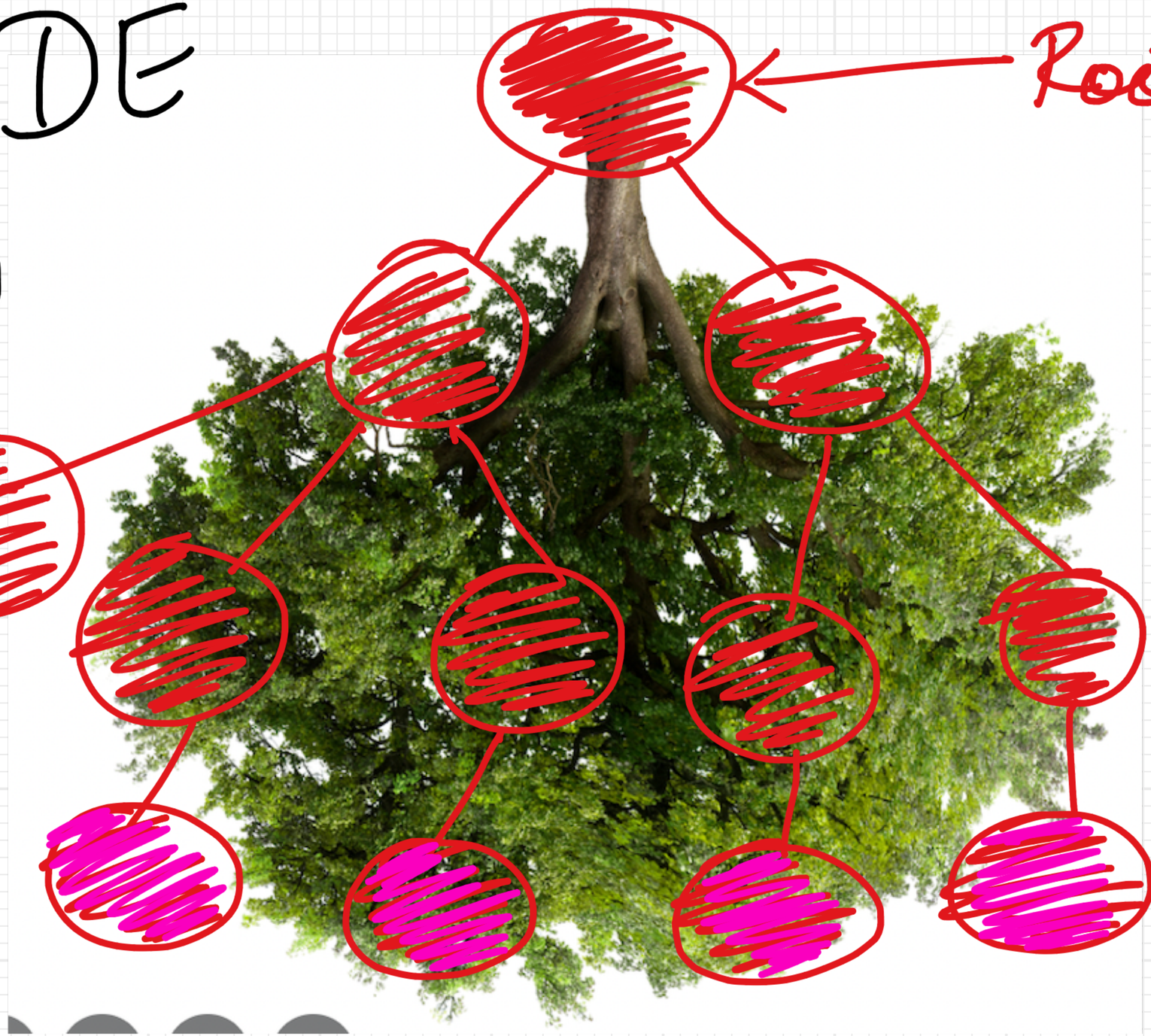
Uncle of 3

Parent of 1 2 3

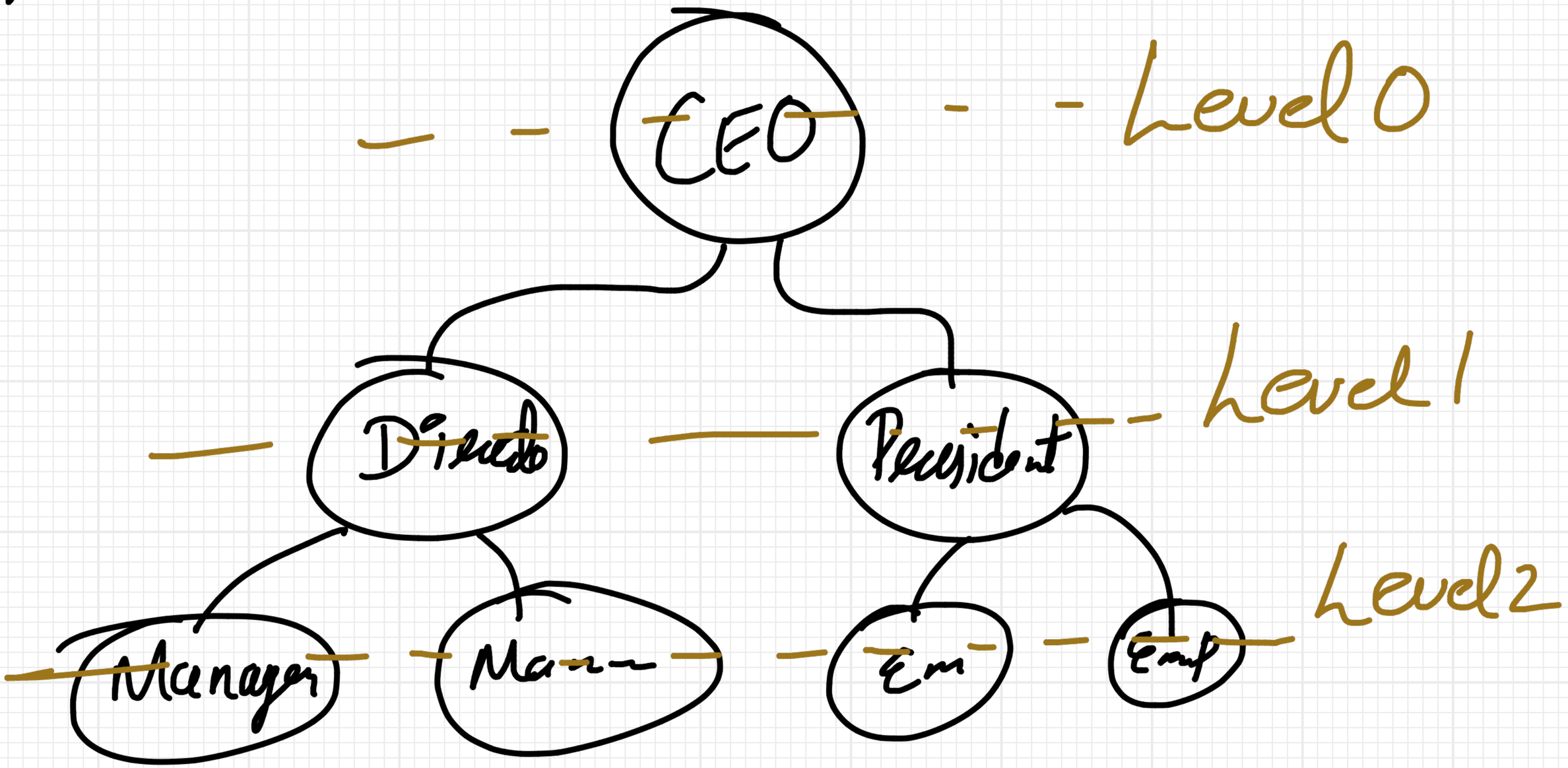
1 2 3 are Siblings



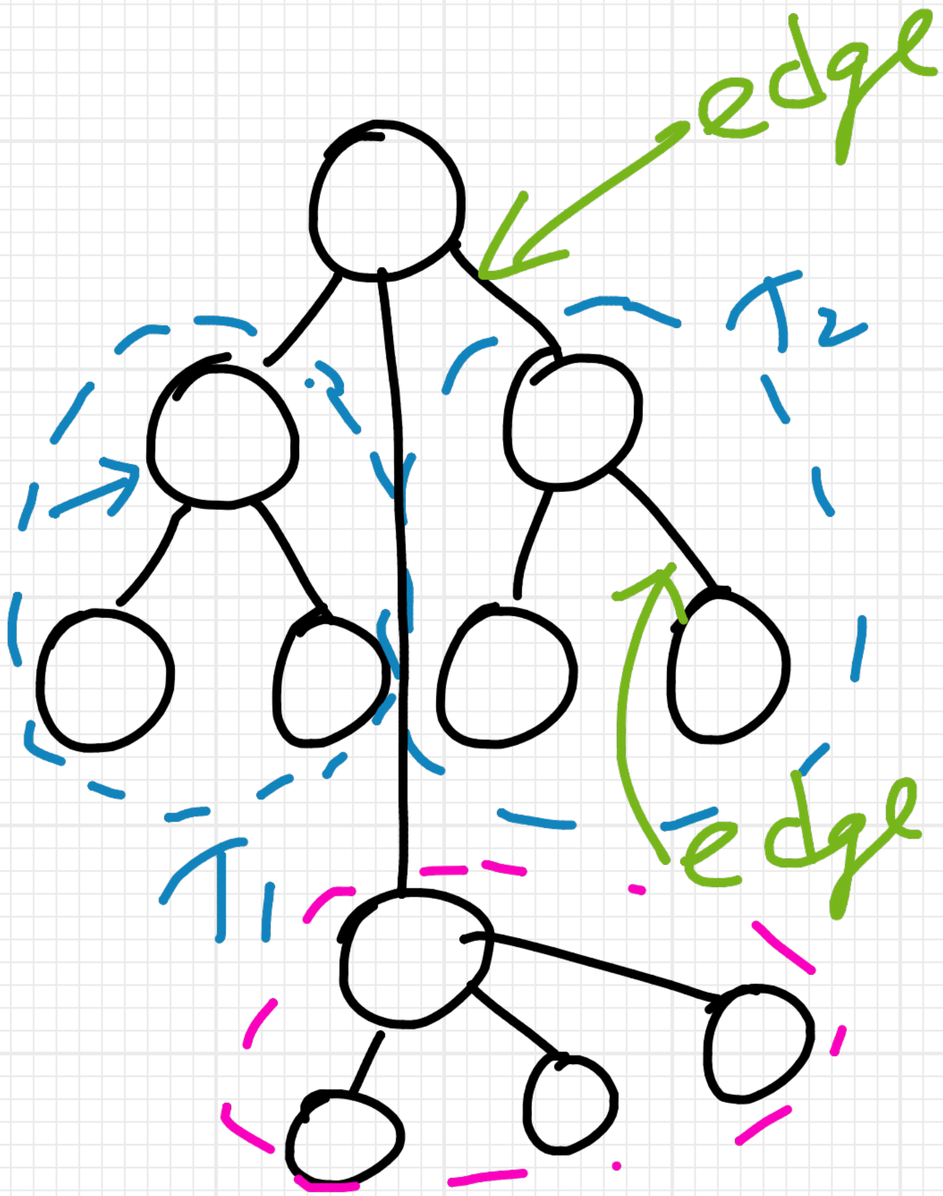
NODE

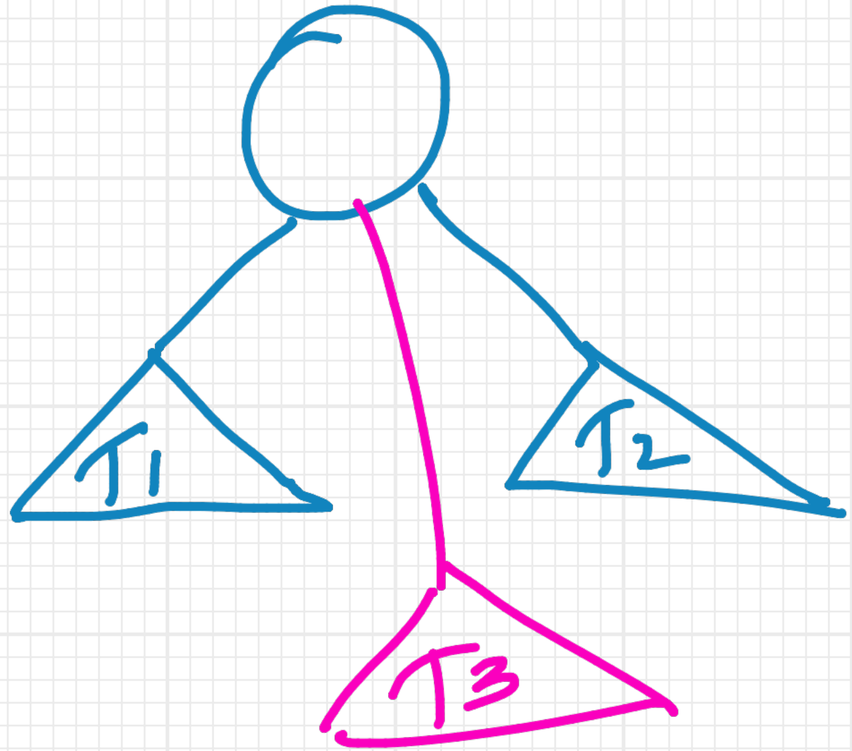


Hierarchical data



Tree → Recursive DS



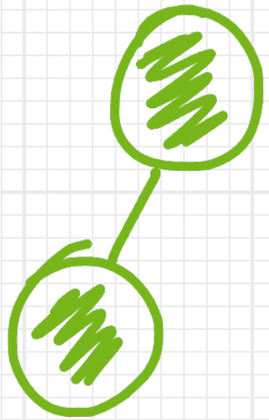


Tree

* edges = nodes - 1

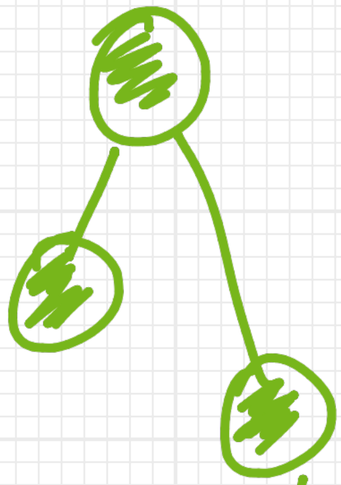
1 Node

0 edges



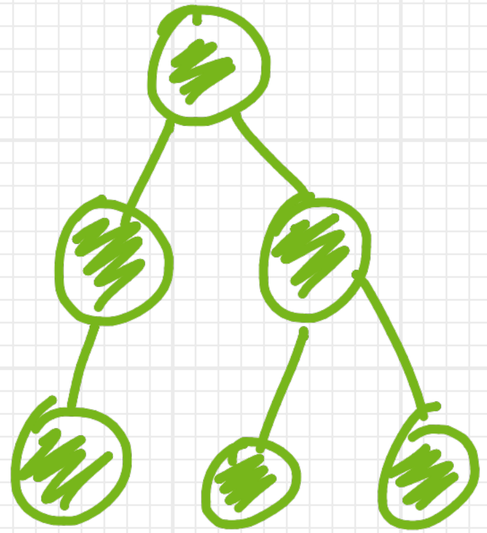
2 Nodes

1 edge



3 Nodes

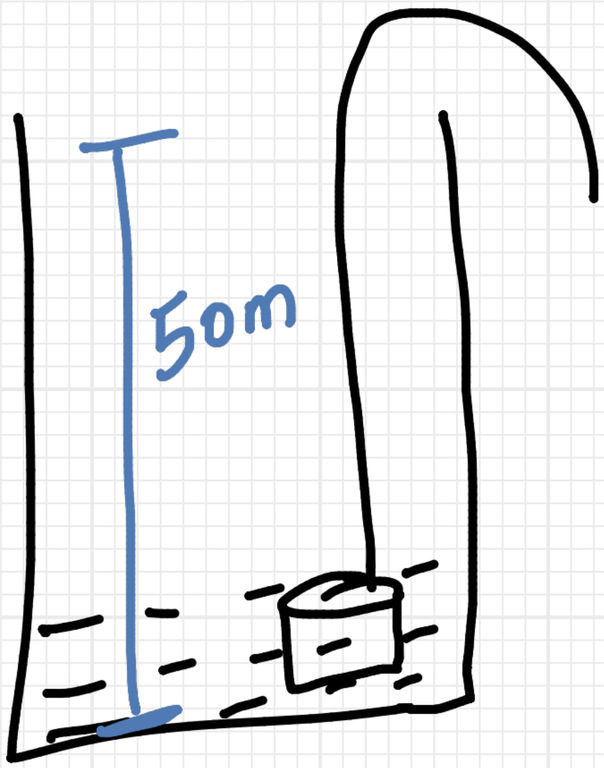
2 edges

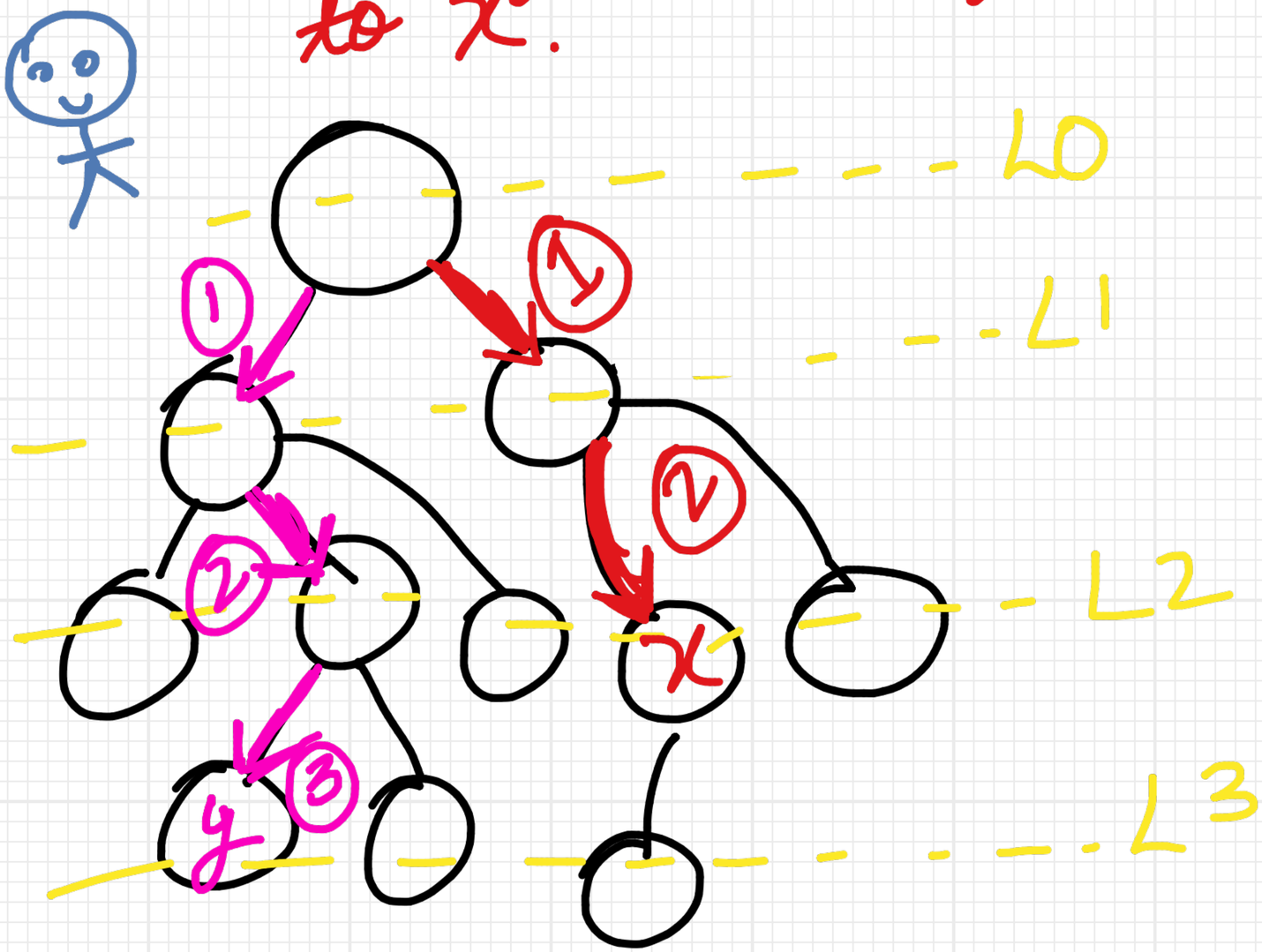


6 Nodes

5 edges

* Depth of x ⇒ No. of edges from root to x.





* Height of x : No of edges in longest path from x to leaf*

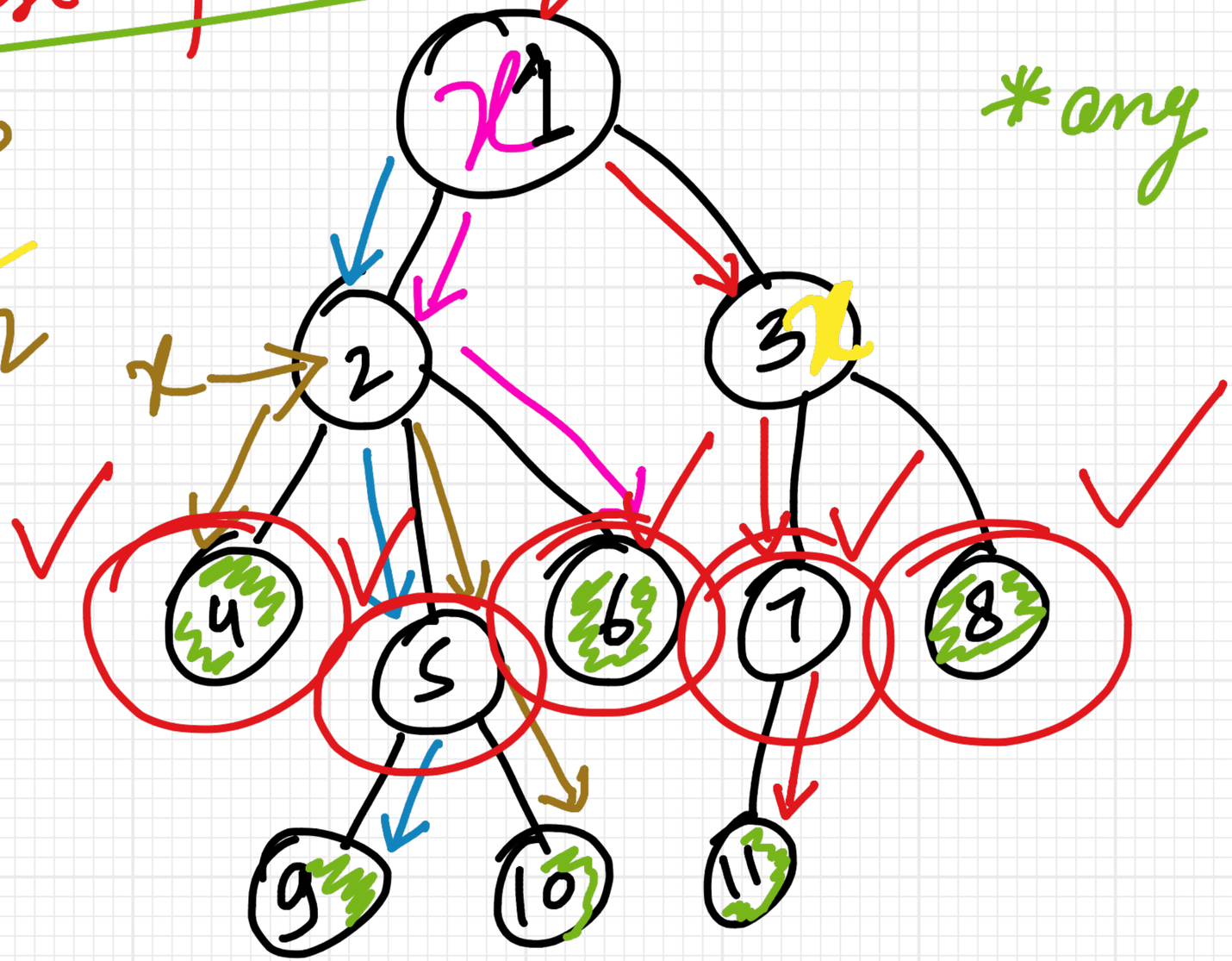
*any leaf node

Root = 3

(3) = 2

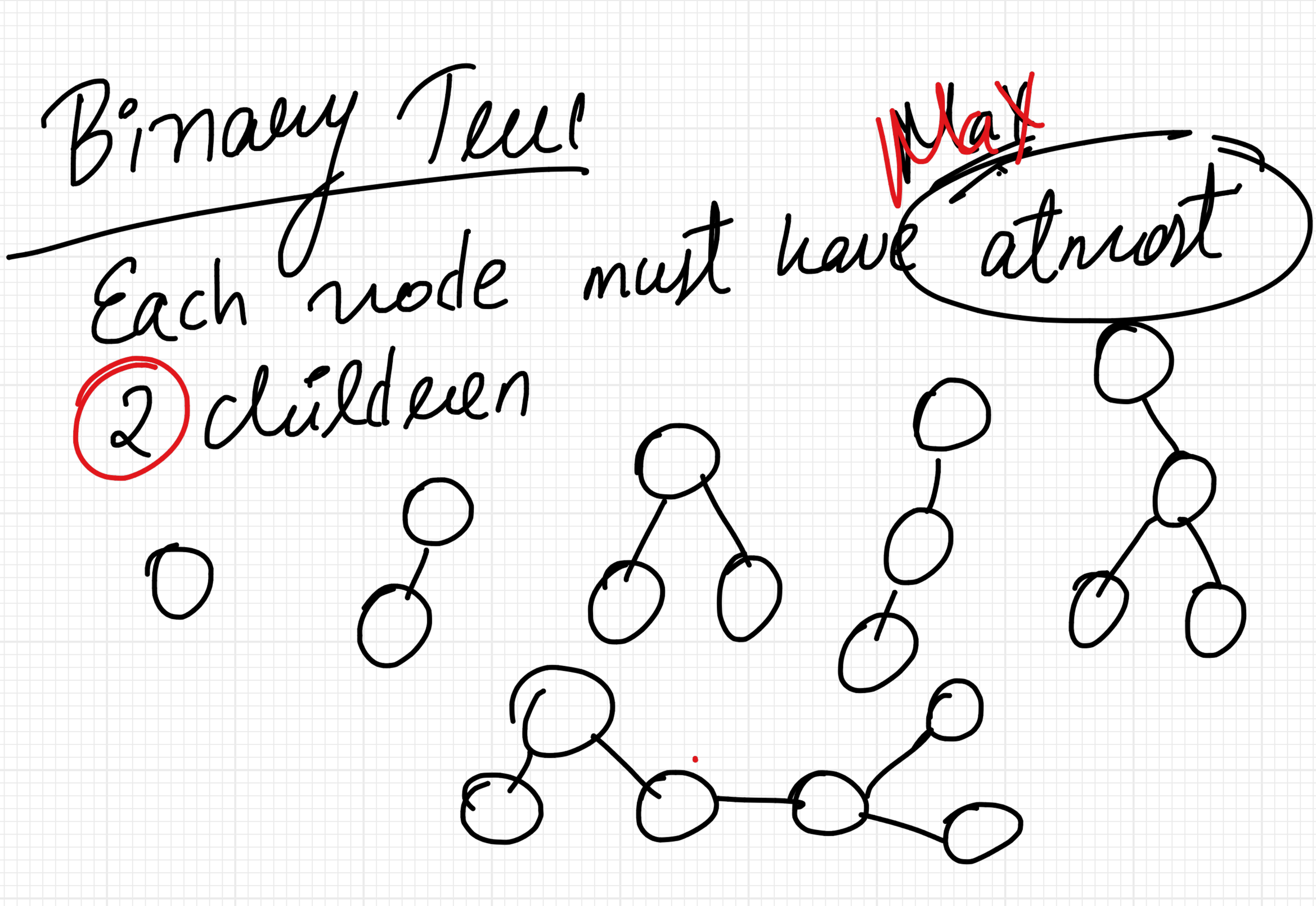
(2) = 2

Leafs Nodes



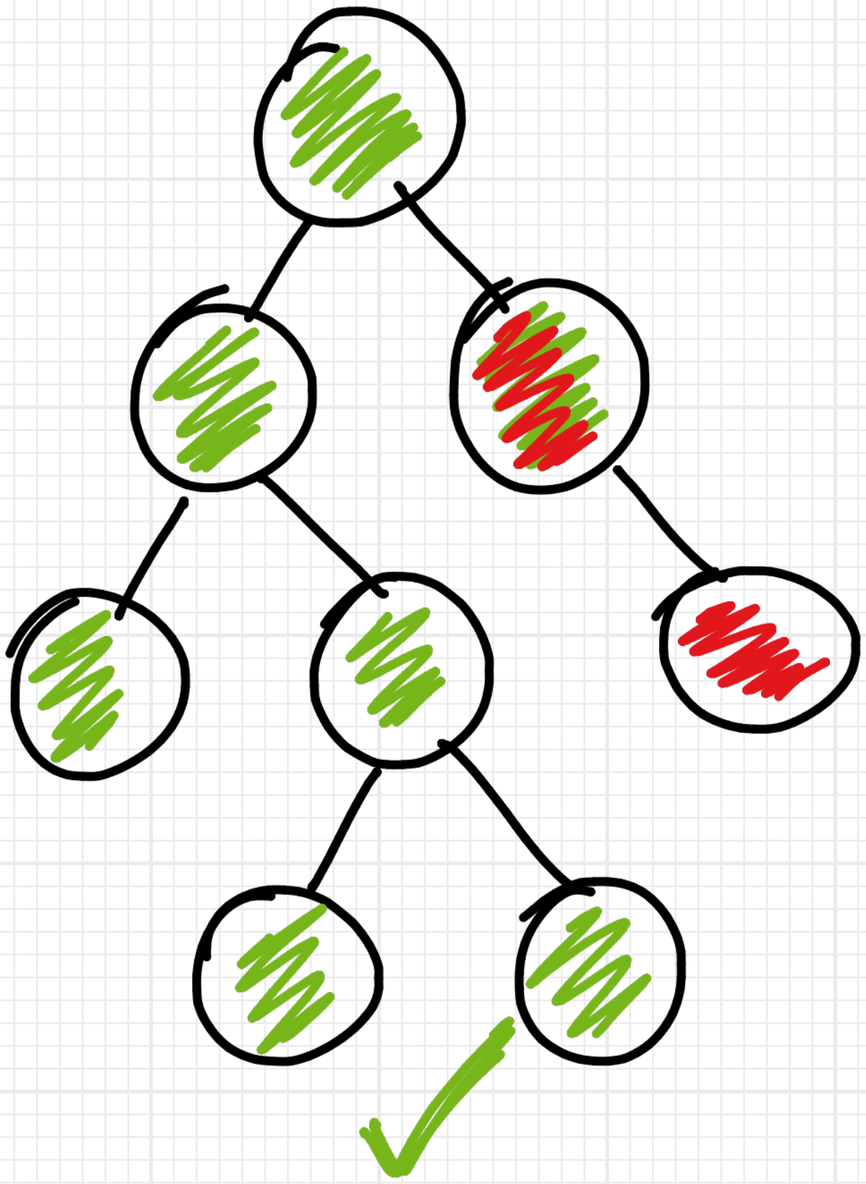
Binary Tree

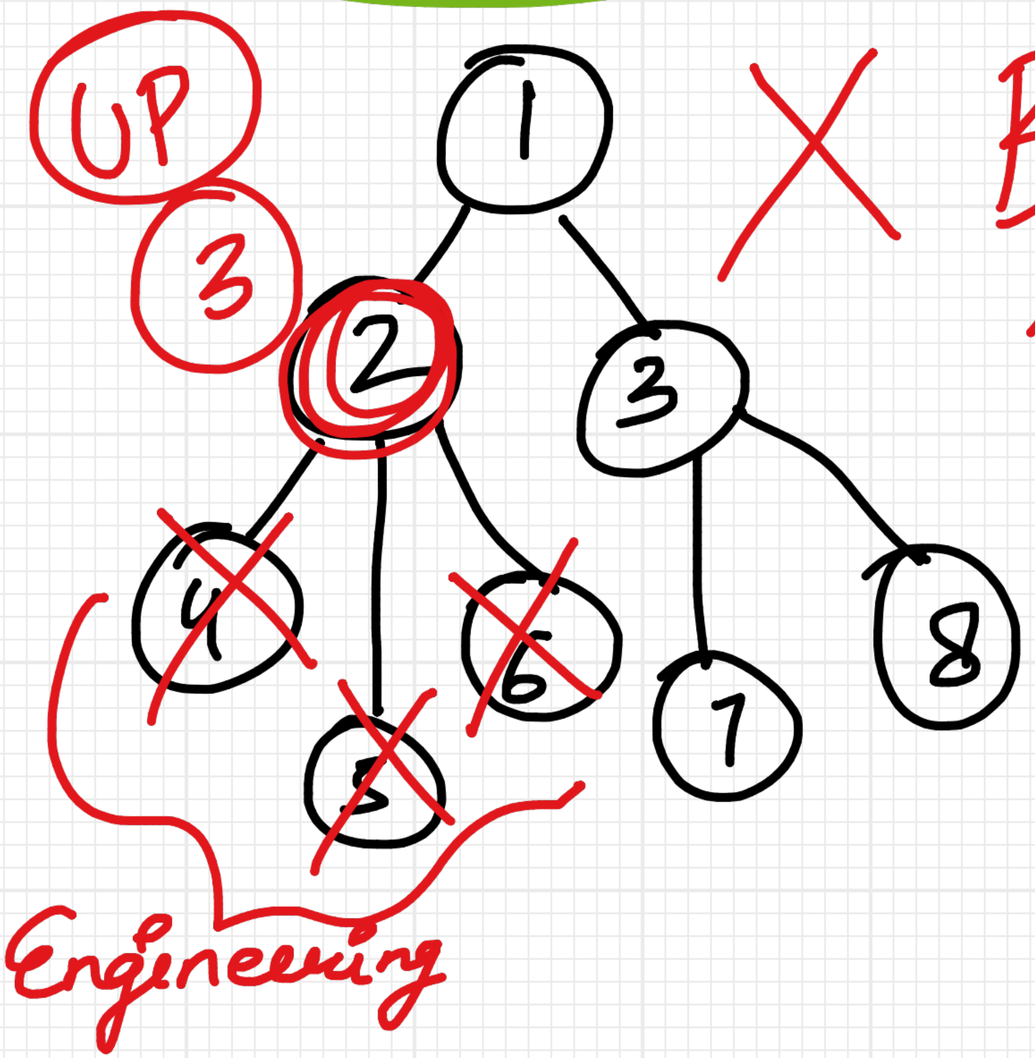
Each node must have Max atmost 2 children



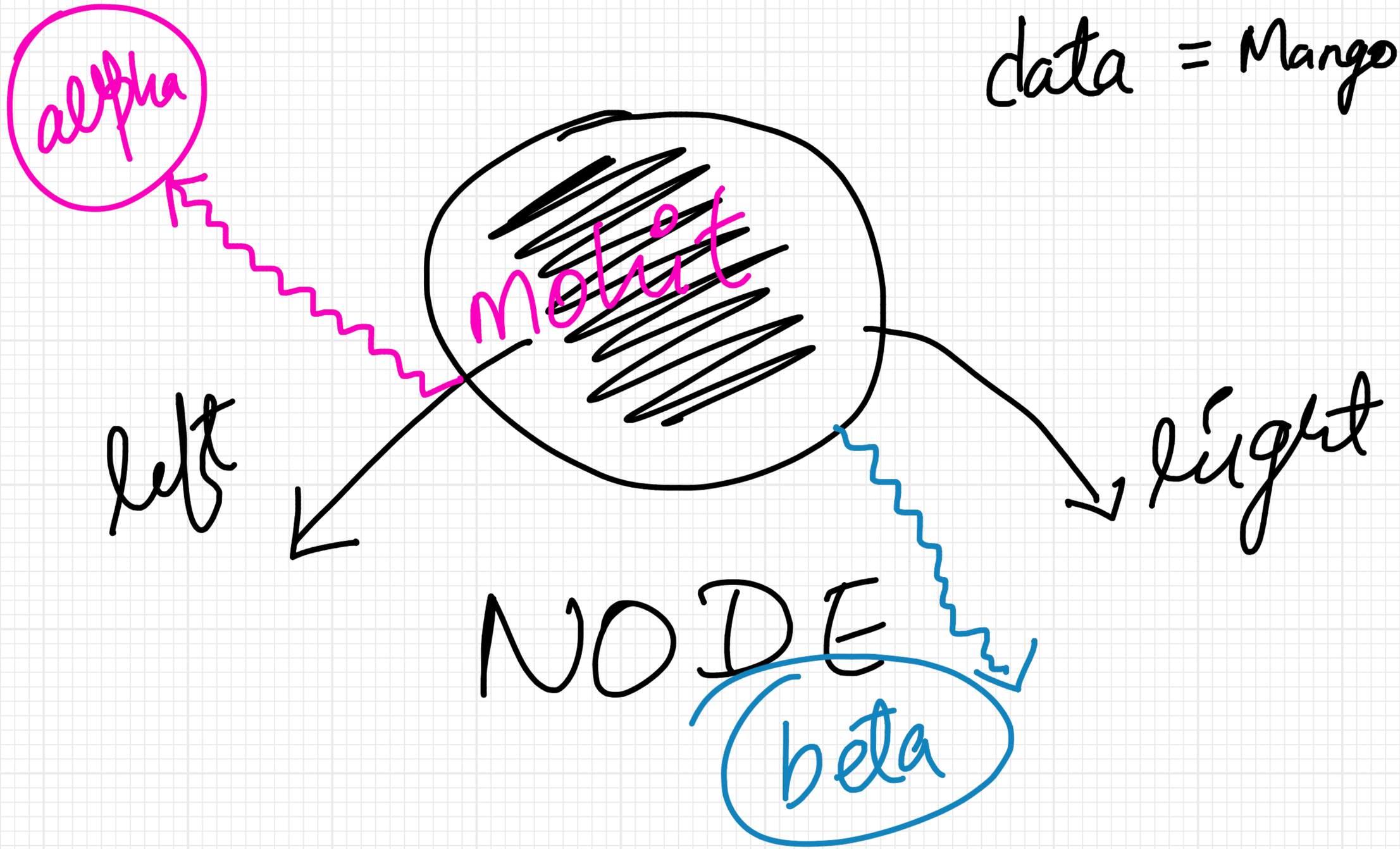
Each node must have atmost 2 children

0 1 2





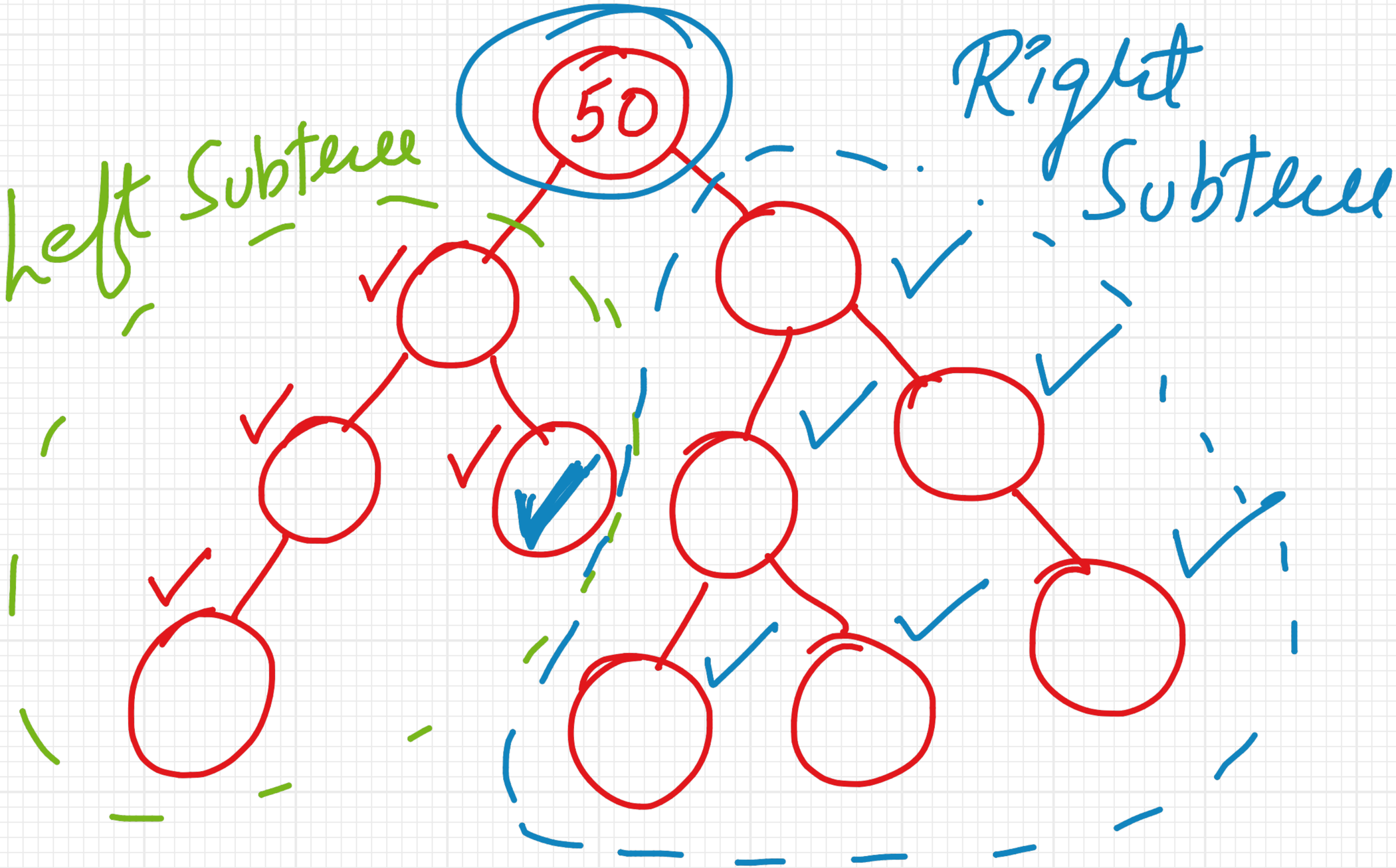
X Binary Tree.

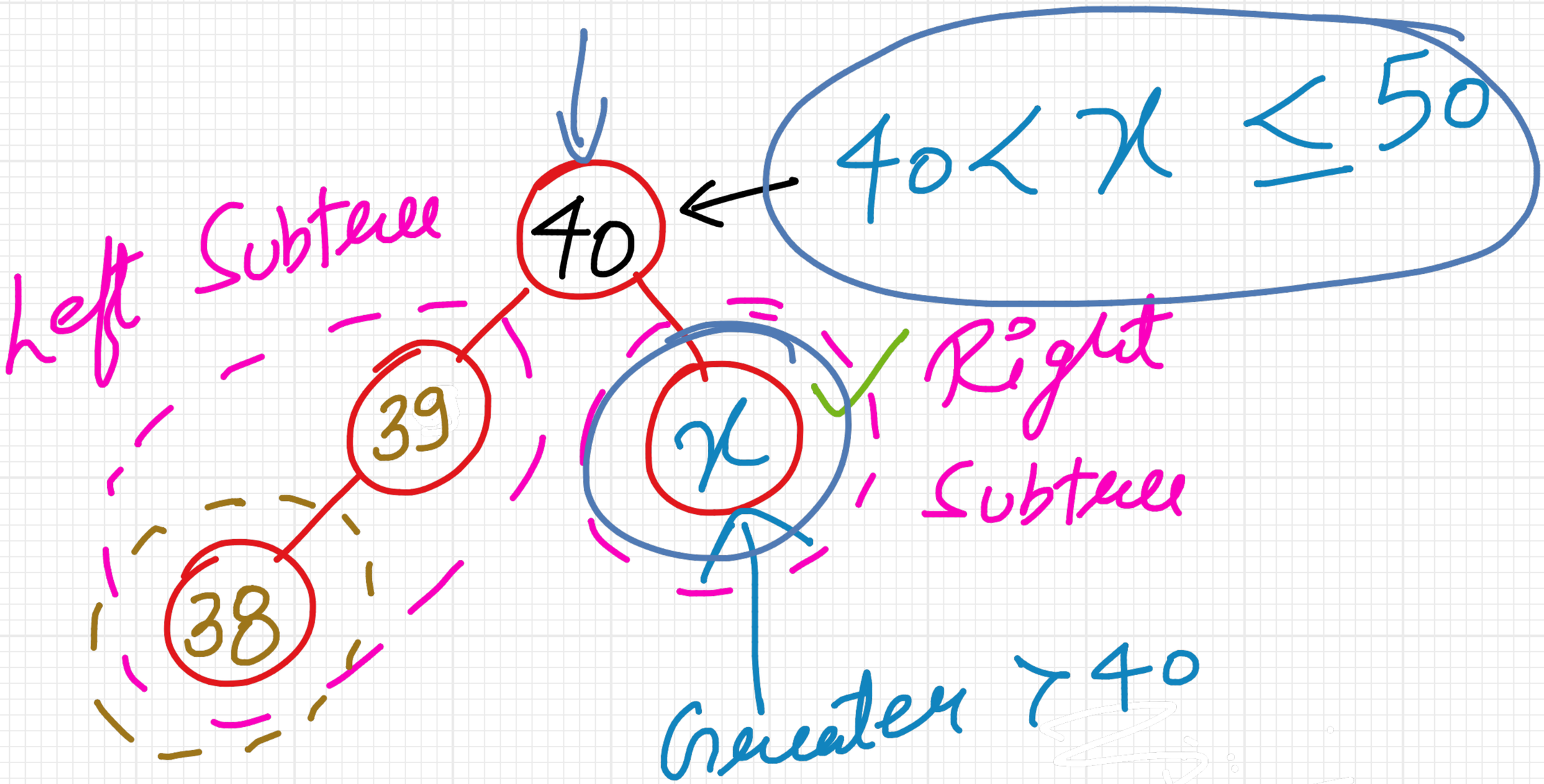

alpha
molut
data = Mango
left
right
NODE
beta

Binary Search Tree

A binary tree is a tree in which for each node, value of all the nodes in left subtree is lesser, and value of each node in right subtree is greater

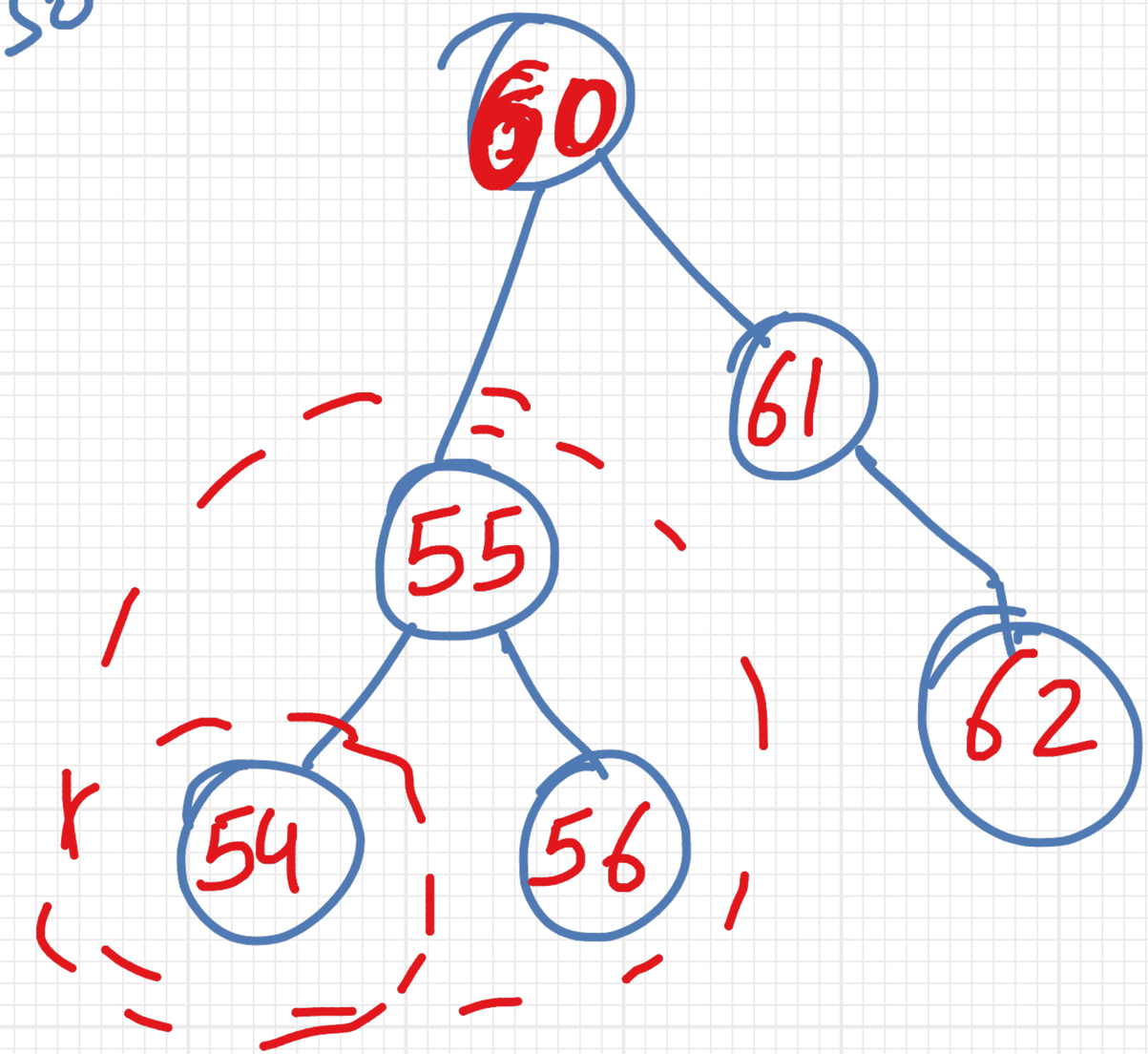
search O(log n)


50
Left Subtree
Right Subtree

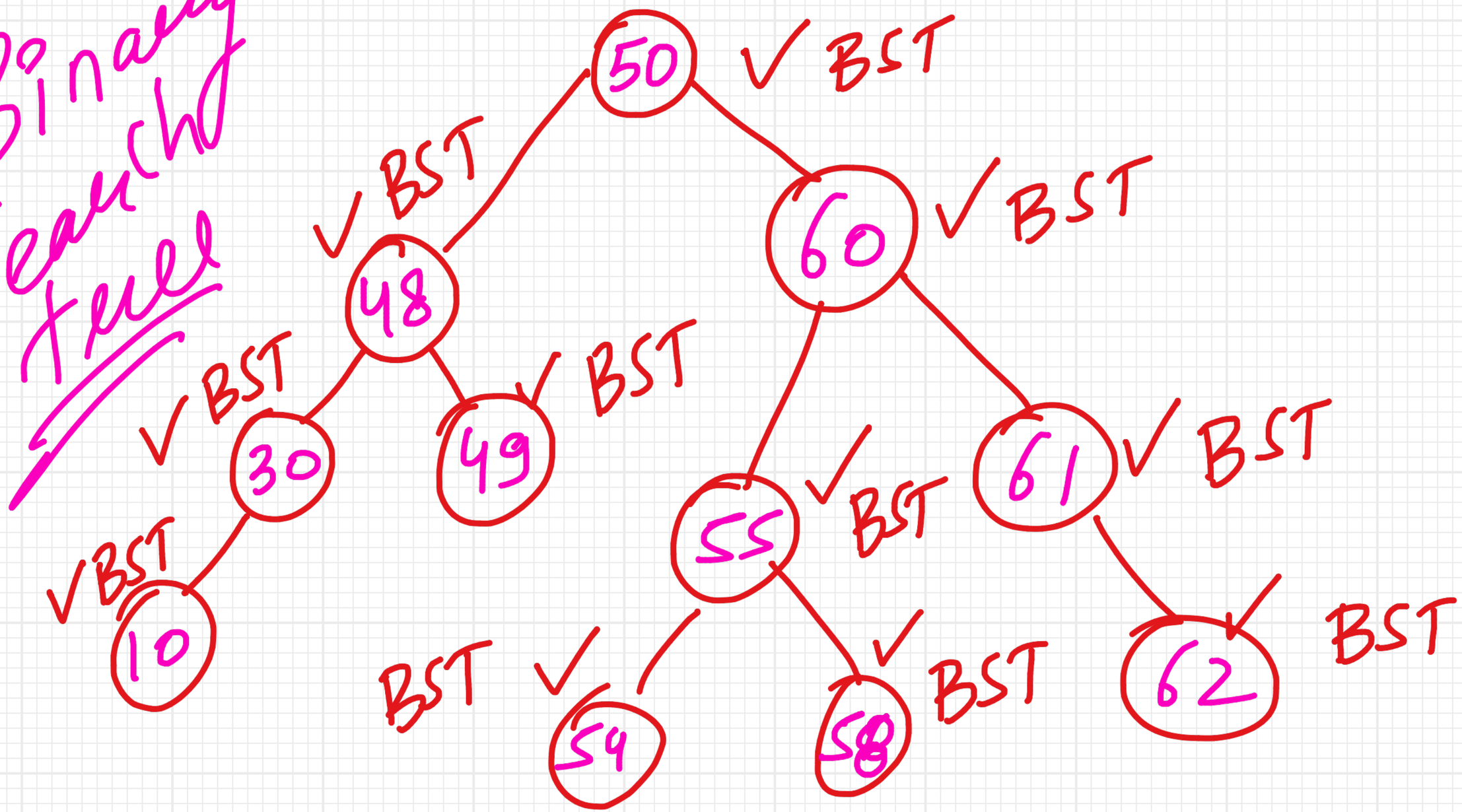

40 < x ≤ 50
Left Subtree
Right
Subtree
40
39
38
x
Greater than 40

Right Subtree with Root = 50

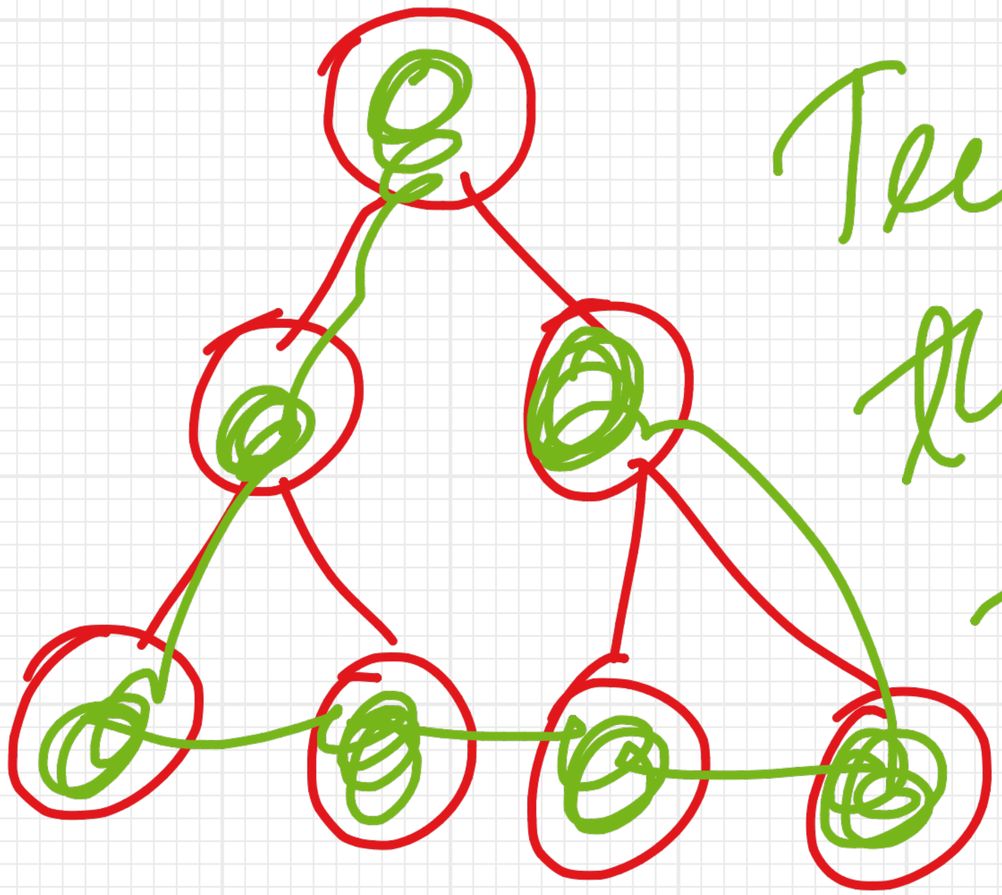
750



Binary Search Tree



Traversal



Travelling through the tree and printing all the values

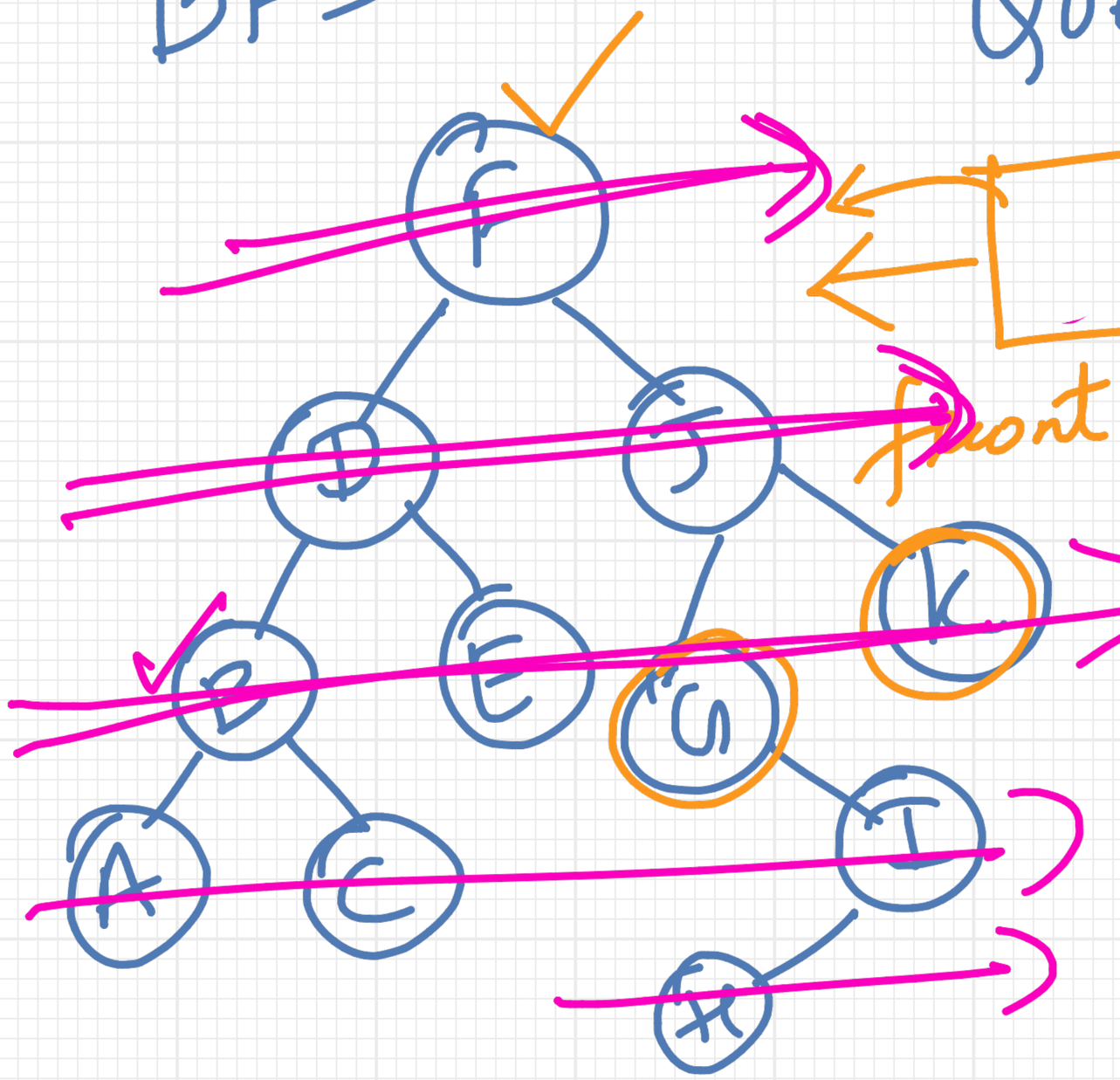
Traversal

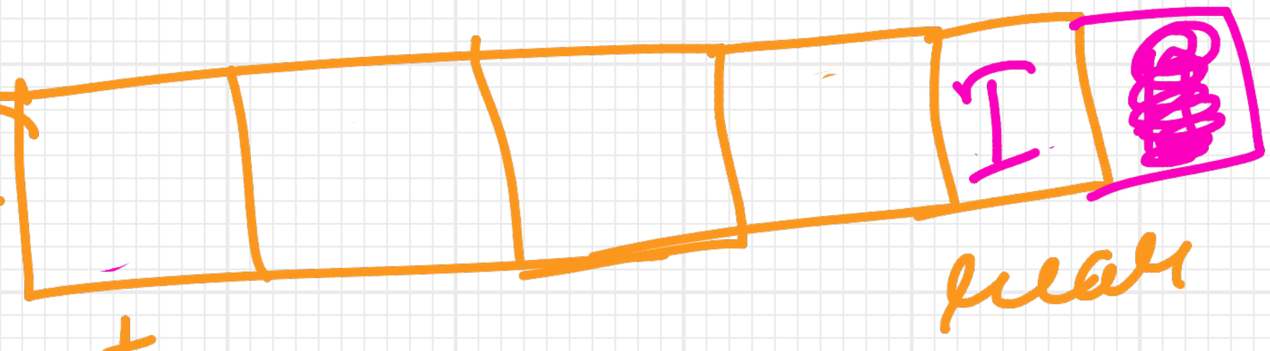
- BFS
  - Breadth first search
- DFS
  - inorder
  - preorder
  - postorder

FDJBEGKACIH

BFS

Queue => FIFO





① Print ele, pop front

② Push it's children from back