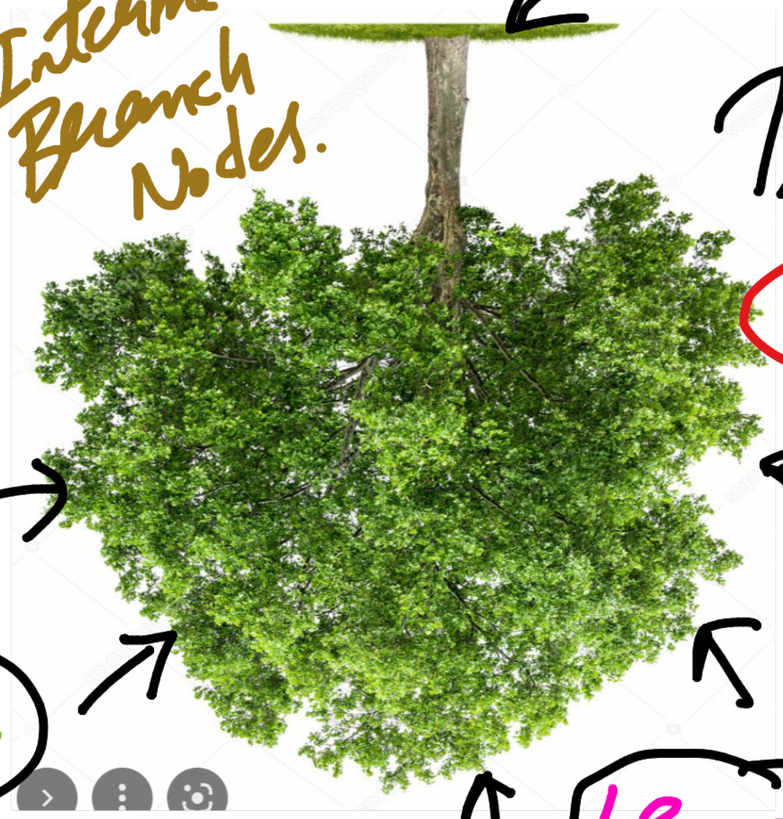
Trees

Root

Intermediate Branch Nodes.

Leaf Nodes



Root

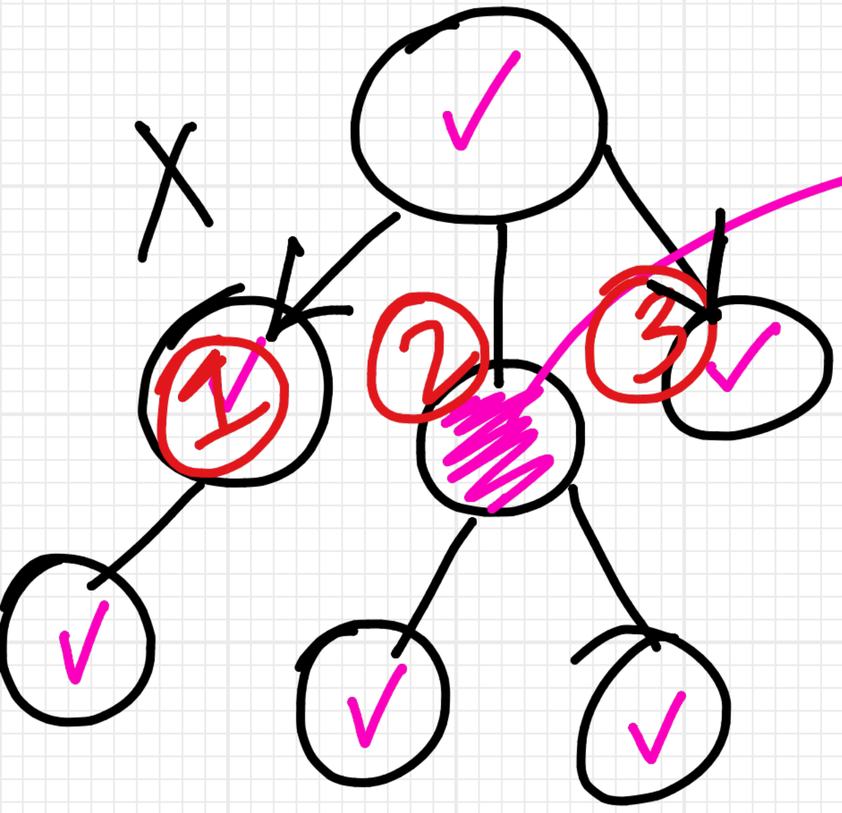
Trees are Non Linear DS

Not in a line

Linear DS

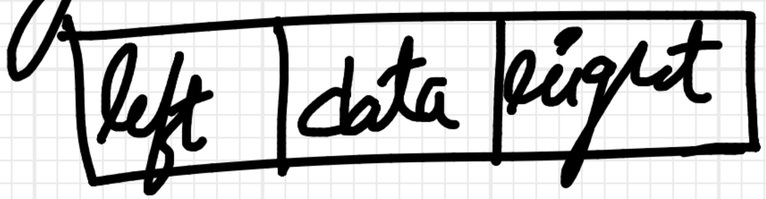
Array, Stack, Queue, Linkedlist

How do you represent Node in Python

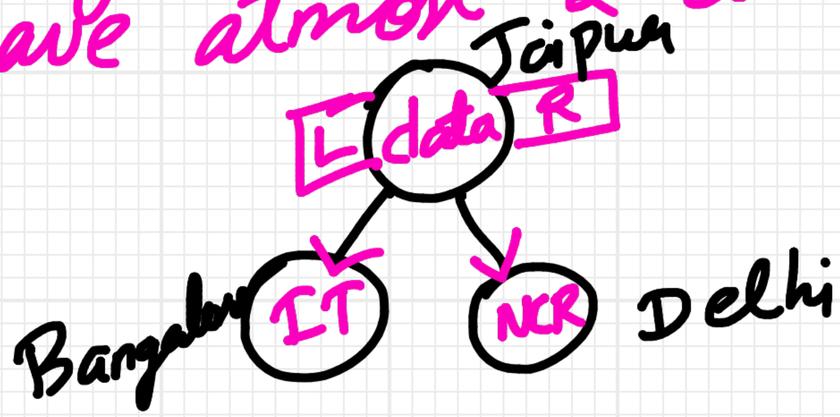


Node

Let's say it is a Binary Tree

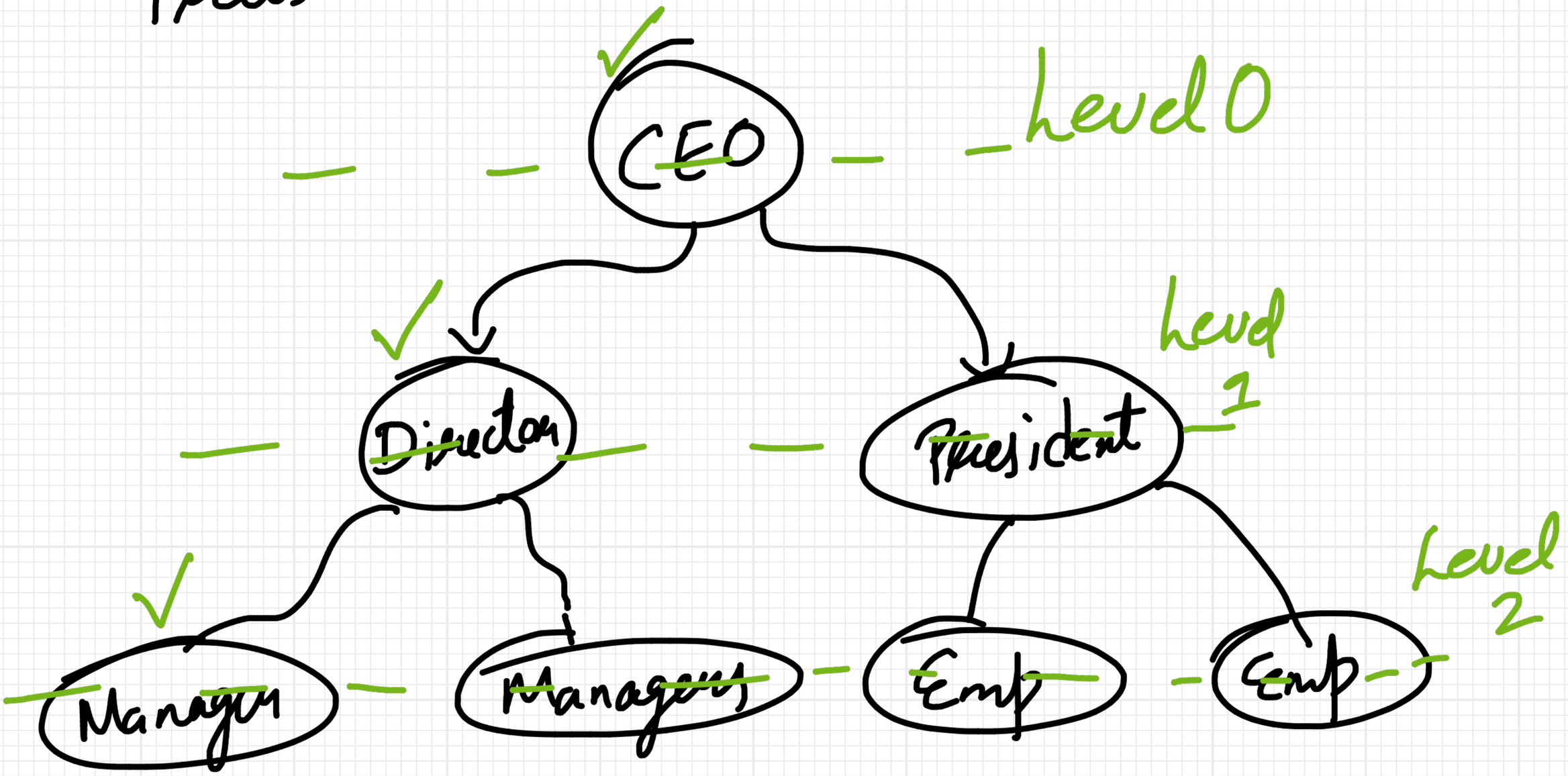


Binary Tree => Each node must have atmost 2 childen

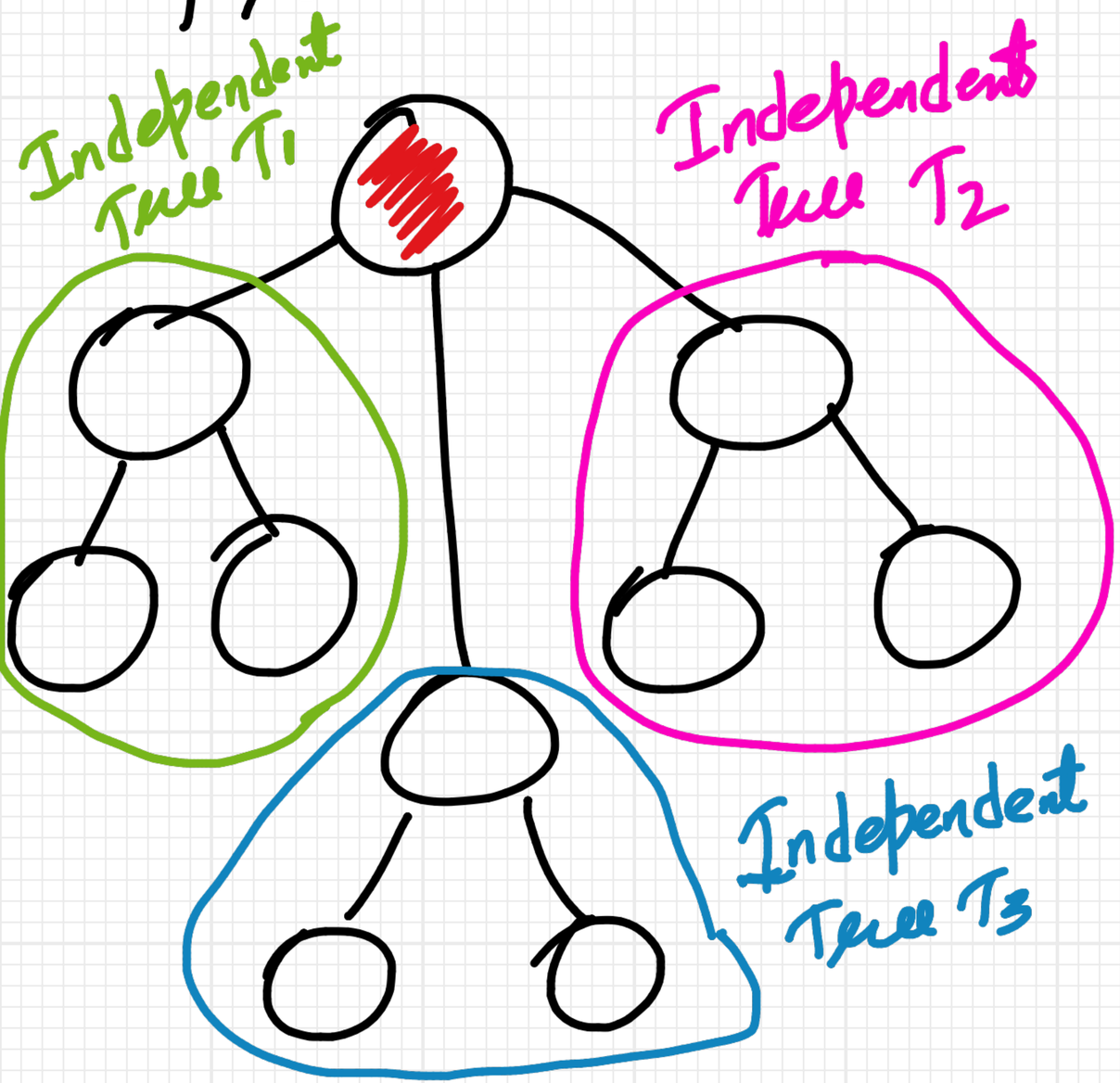


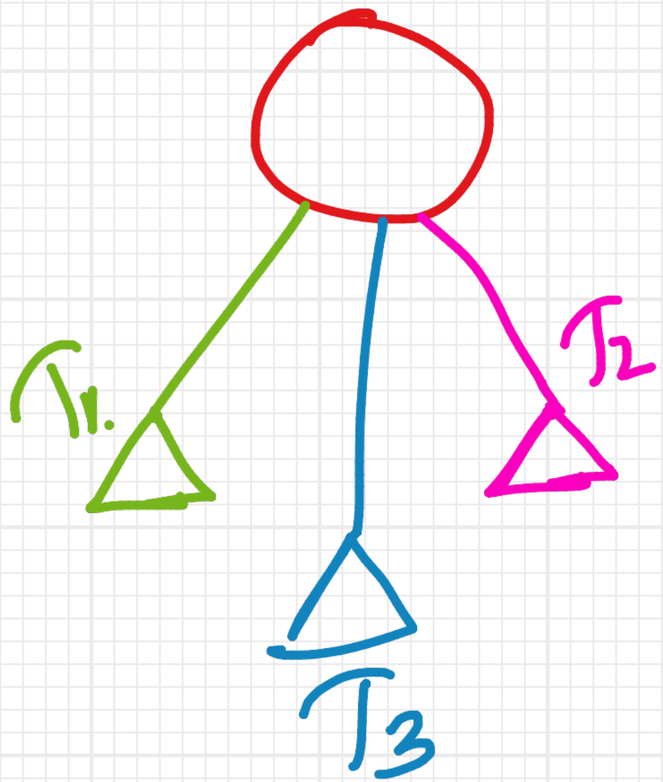
1, 2, 3 are children of root.

Trees are Hierarchical D.S



Trees are Recursive D.S





Q1. Who is Root?

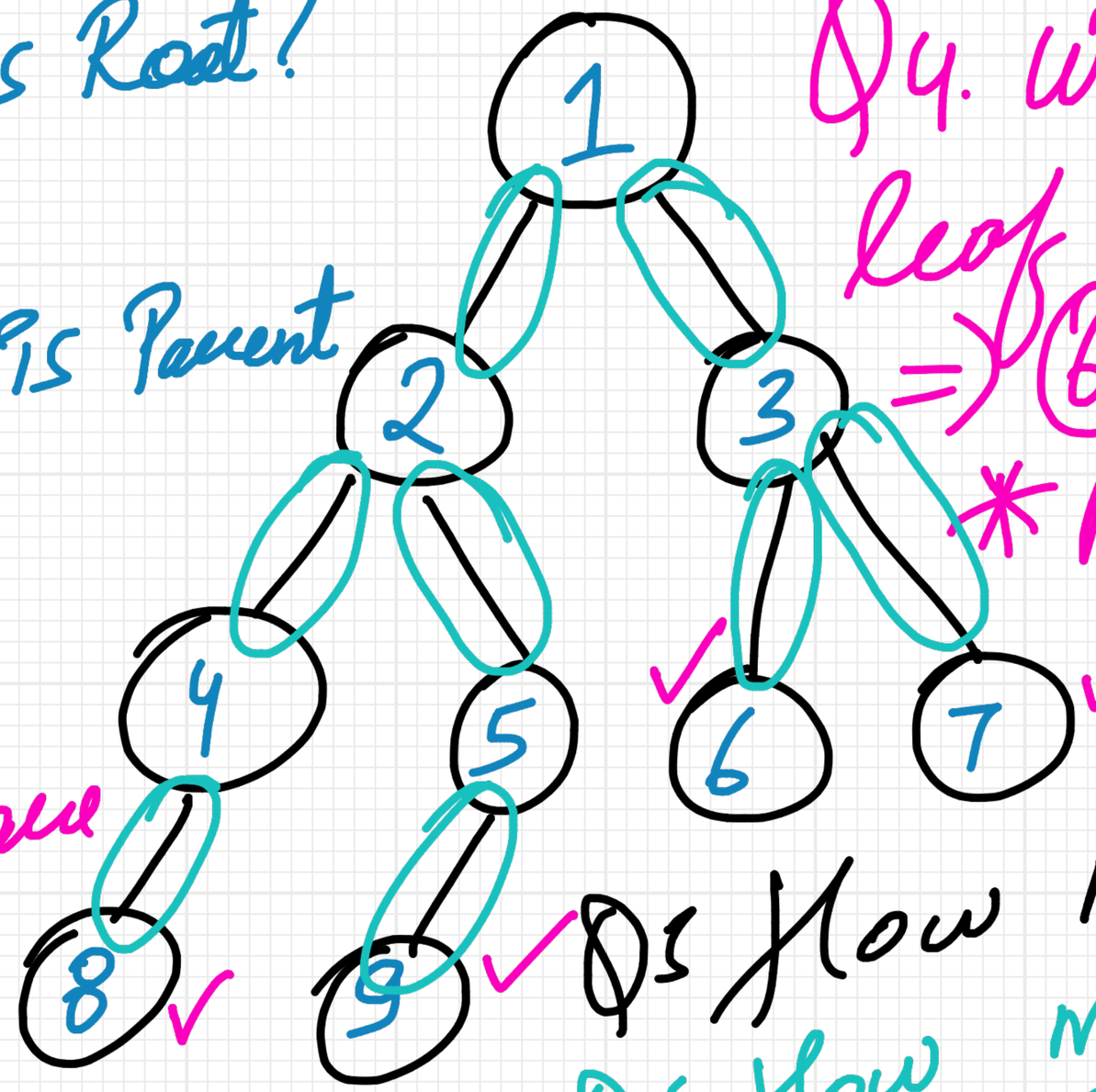
⇒ (1)

Q2. Who is Parent of (5)

⇒ (2)

Q3. Who are children of (2)

⇒ (4), (5)



Q4. What are leaf nodes here?

⇒ (6) (7) (8) (9)

* No children

Q5. How many Nodes?

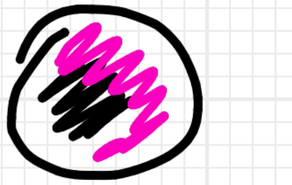
⇒ 9 Nodes

Q6. How many Edges?

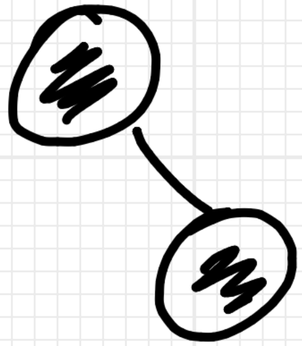
⇒ 8 Edges

For Trees

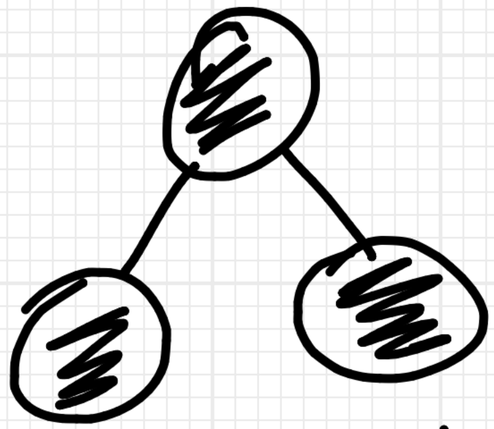
$$\text{Edges} = \text{Nodes} - 1$$



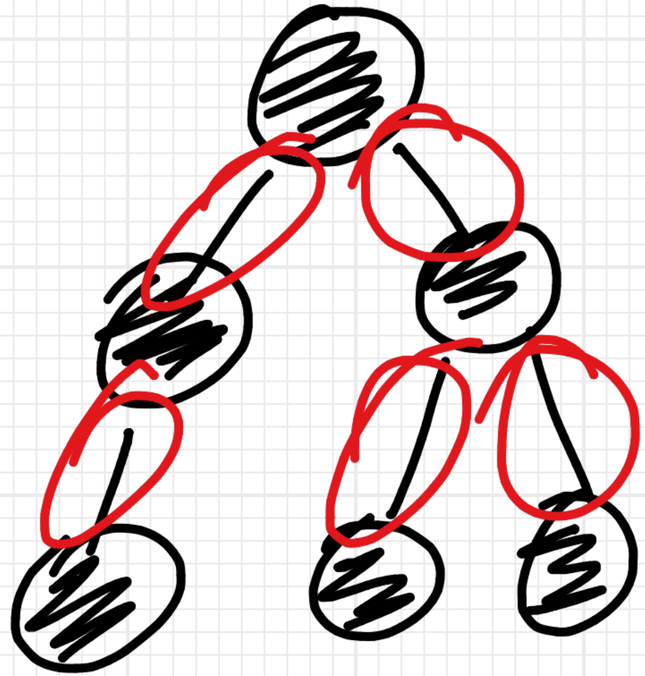
1 Node
0 edges



2 Nodes
1 edge



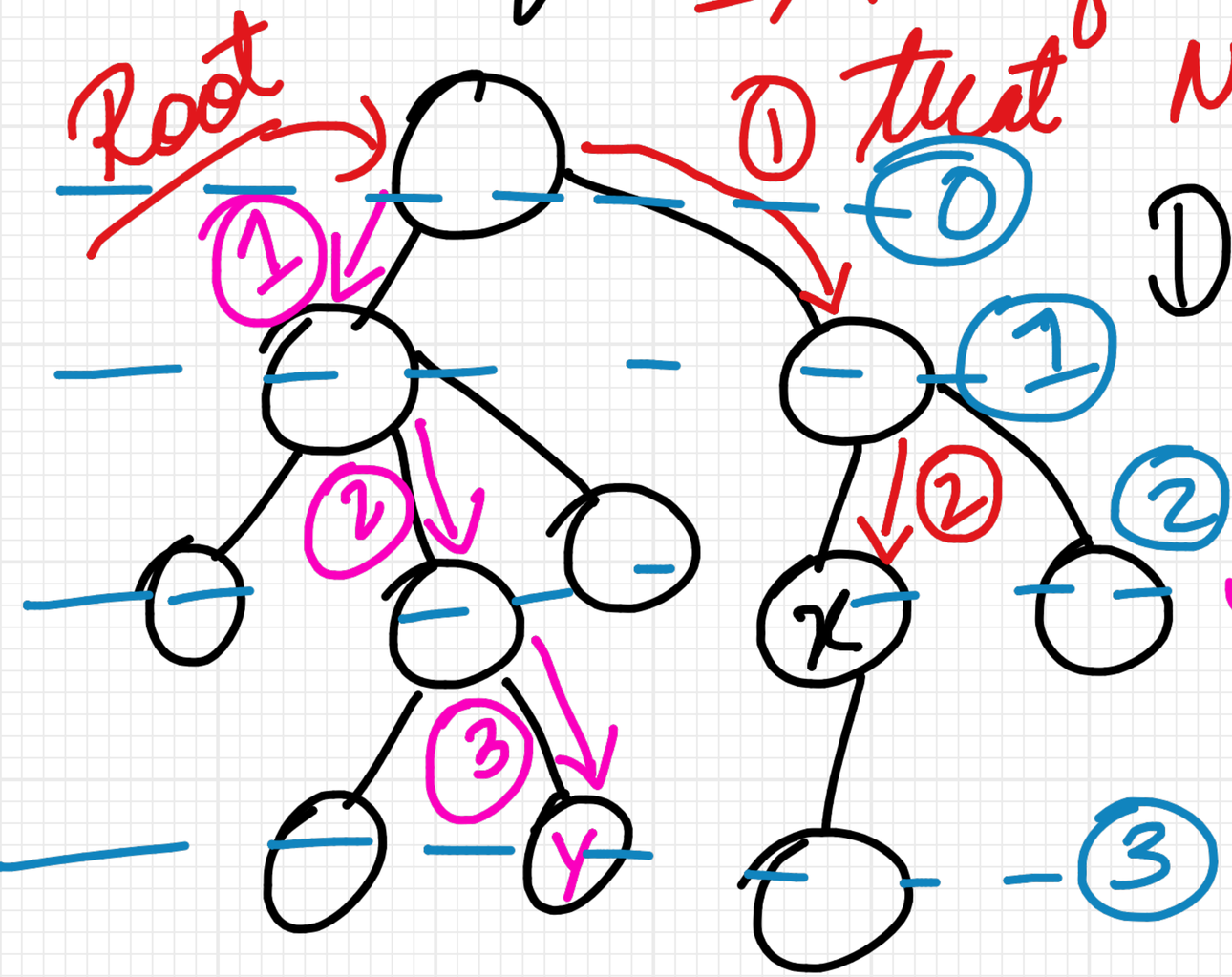
3 Nodes
2 Edges



6 Nodes
5 edges

Depth of Node

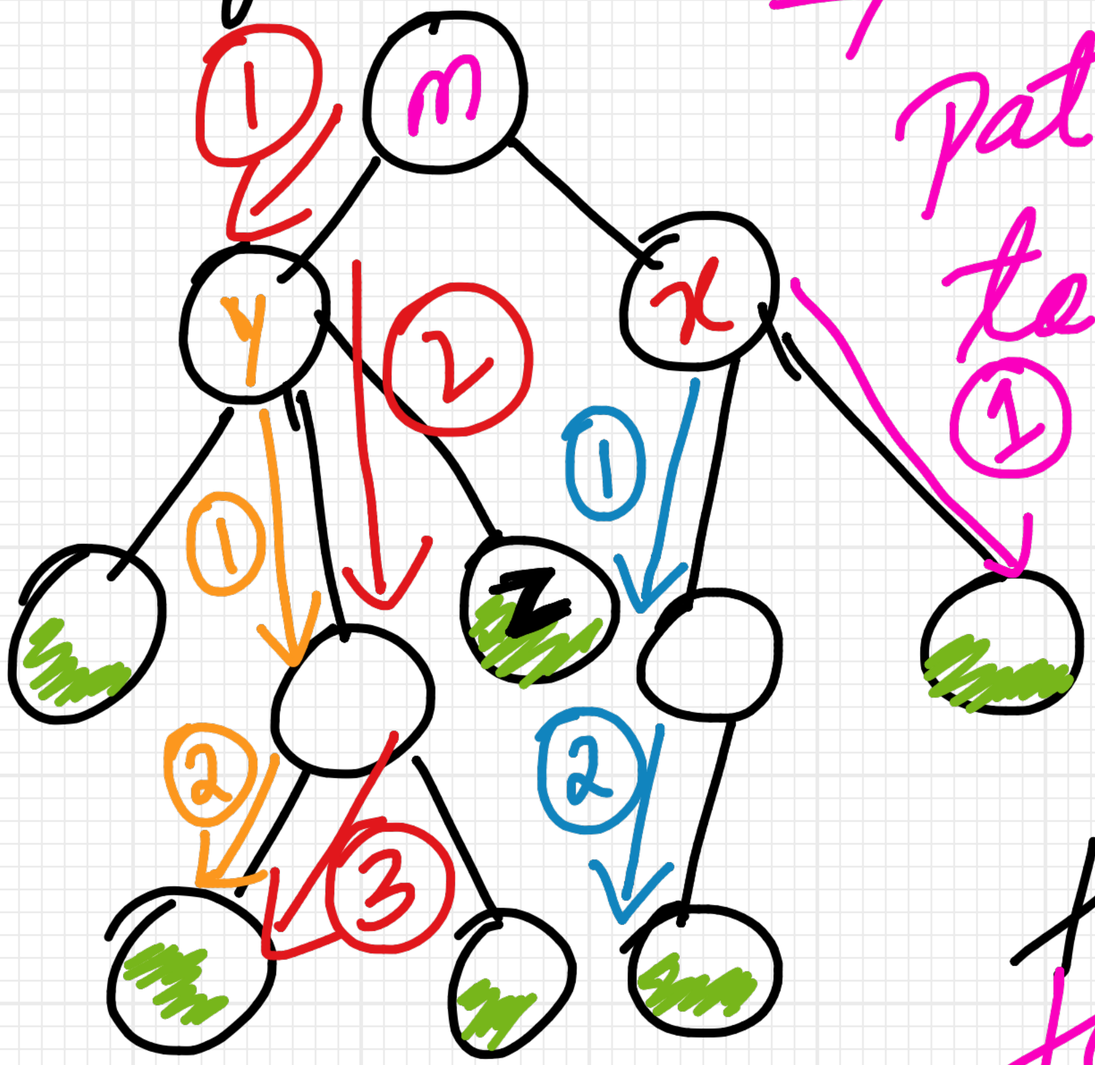
=> No. of Edges from root to that Node



Depth of x = 2

Depth of y = 3

Height of Node



⇒ No. of edges in *longest path from that node to any leaf.

Height of x = 2

Height of y = 2

Height of z = 0

Height of m = 3

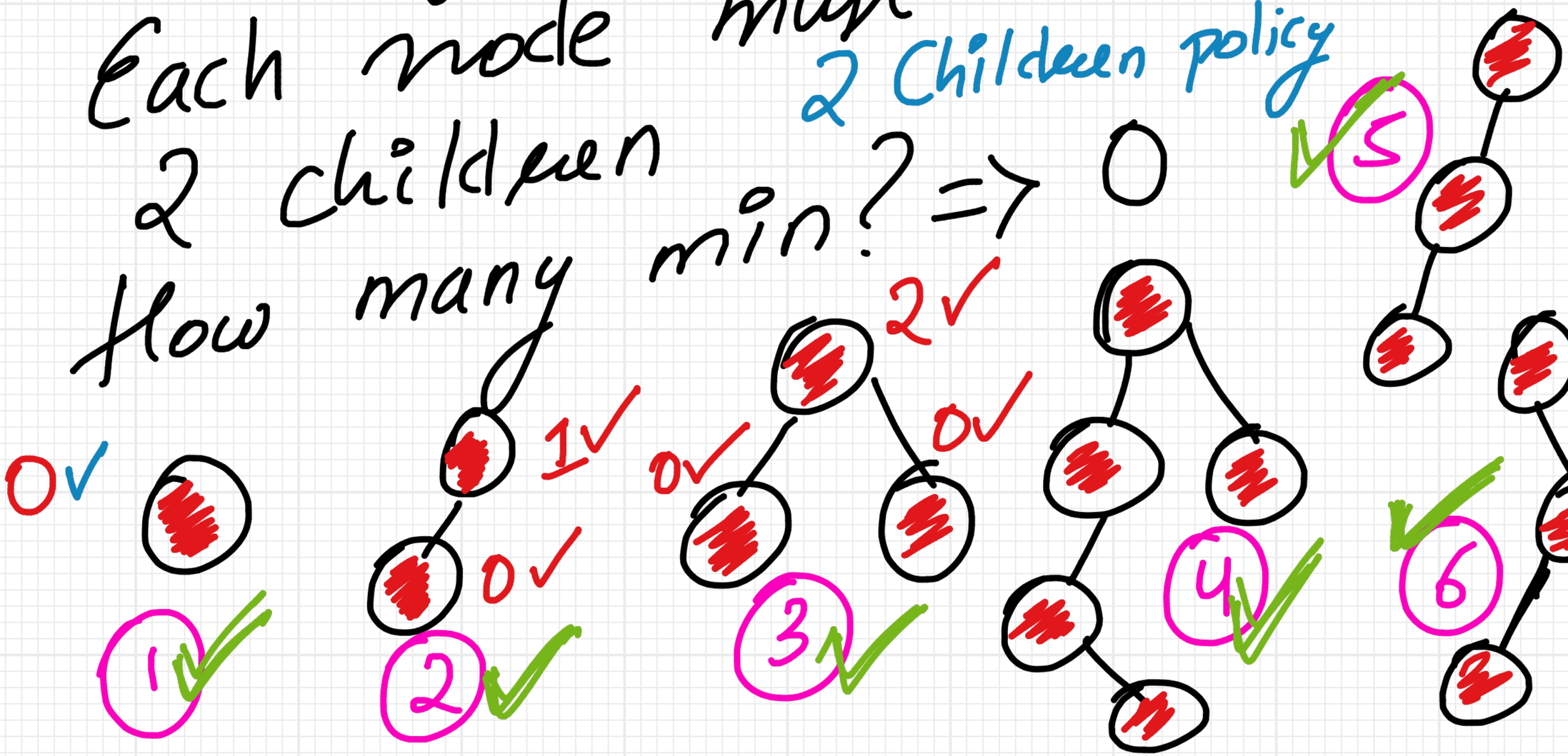
| Height of Tree = Height of Root Node = 3

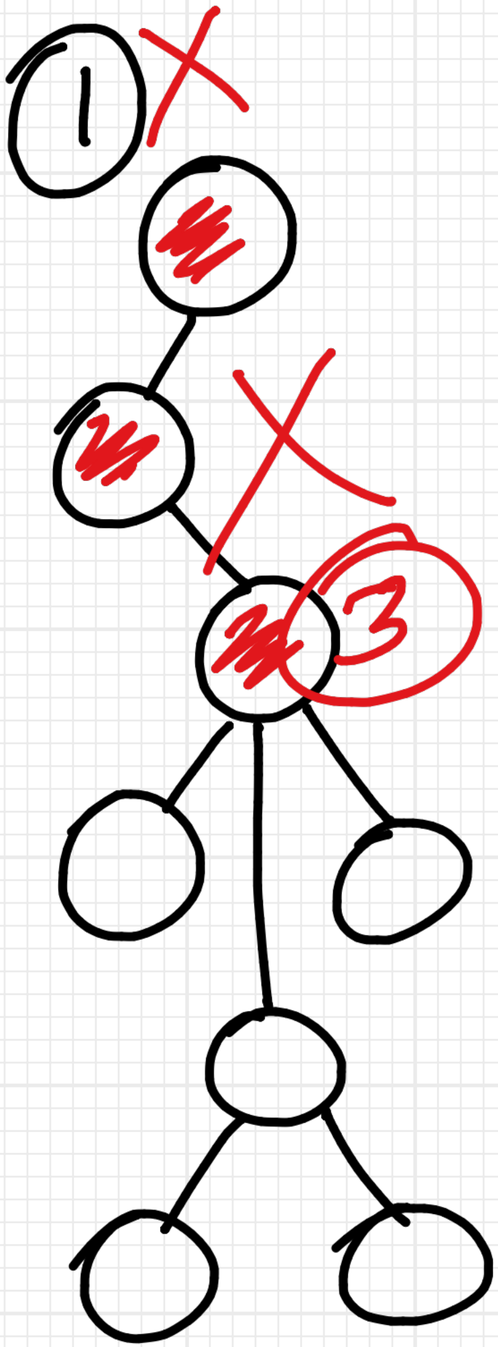
Binary Tree

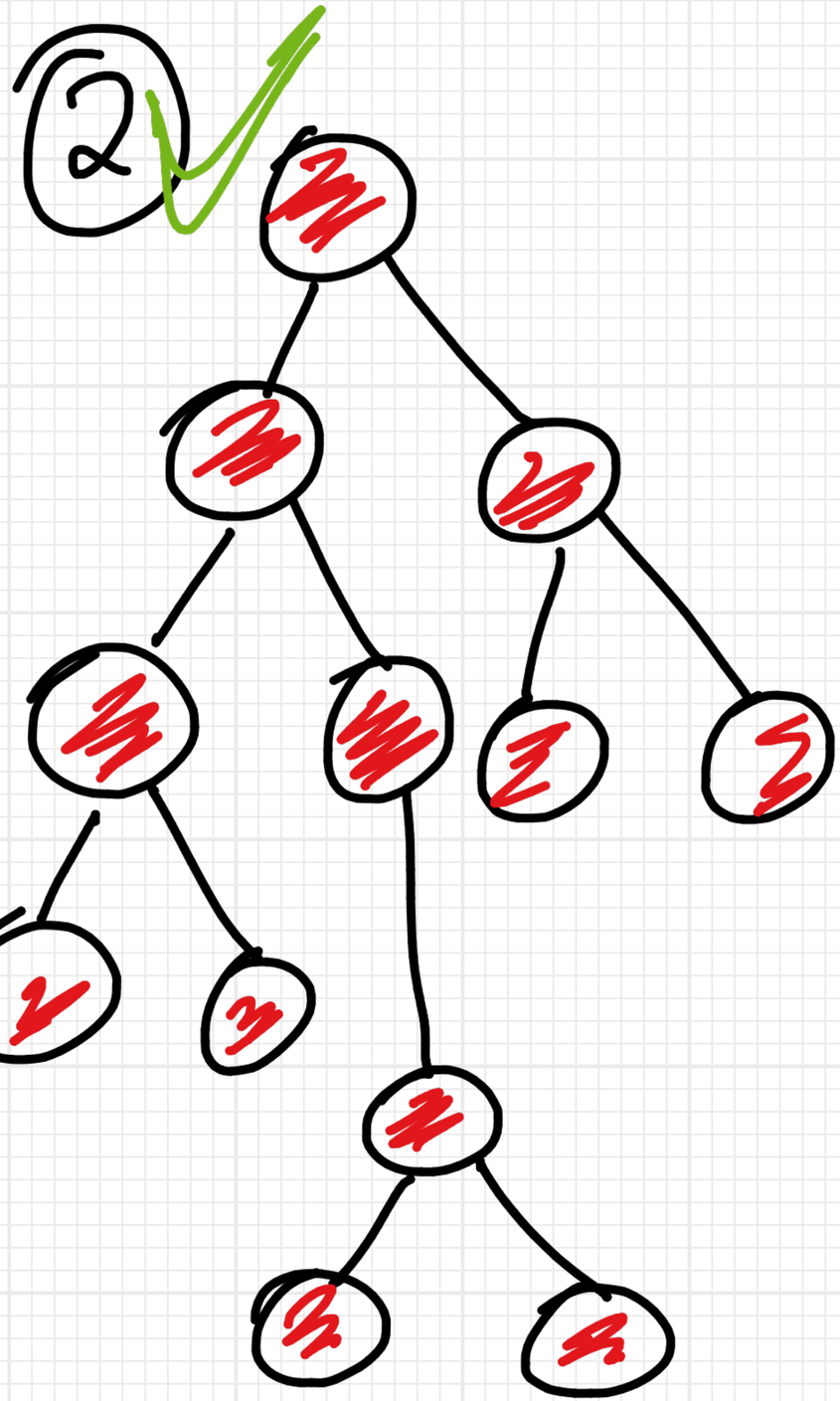
min 0, max 2
=> 0, 1, 2

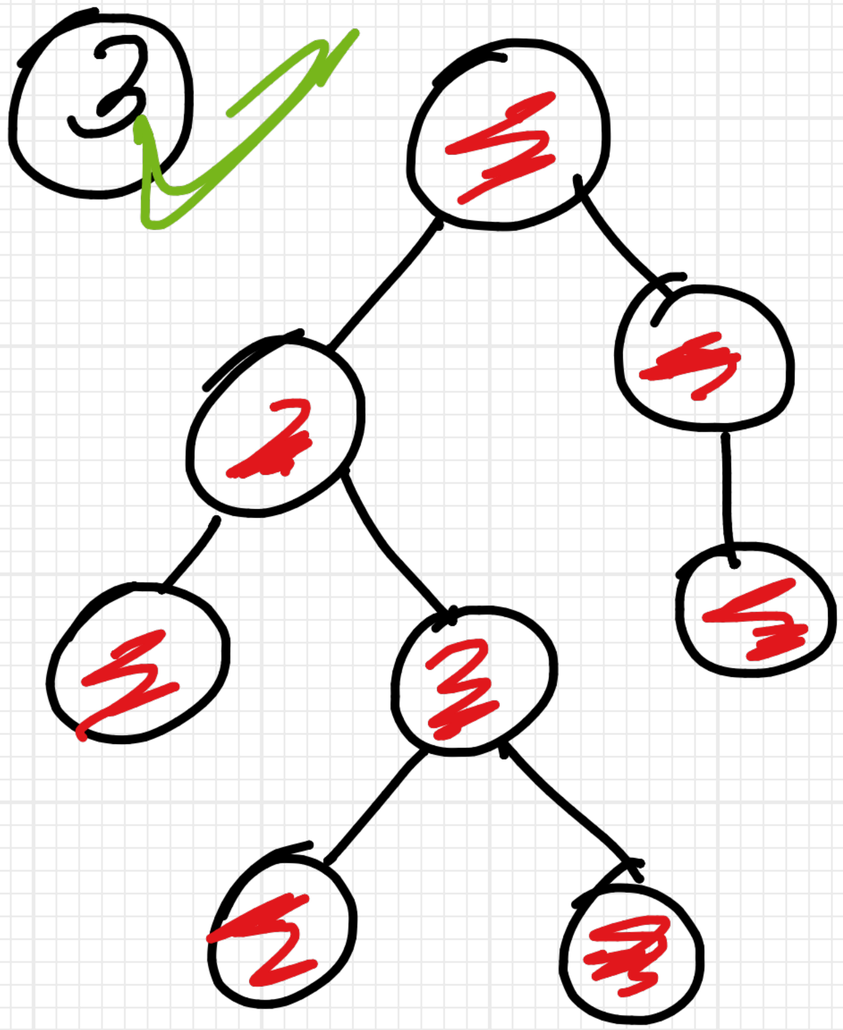
Each node must have max 2 children policy

How many min? => 0

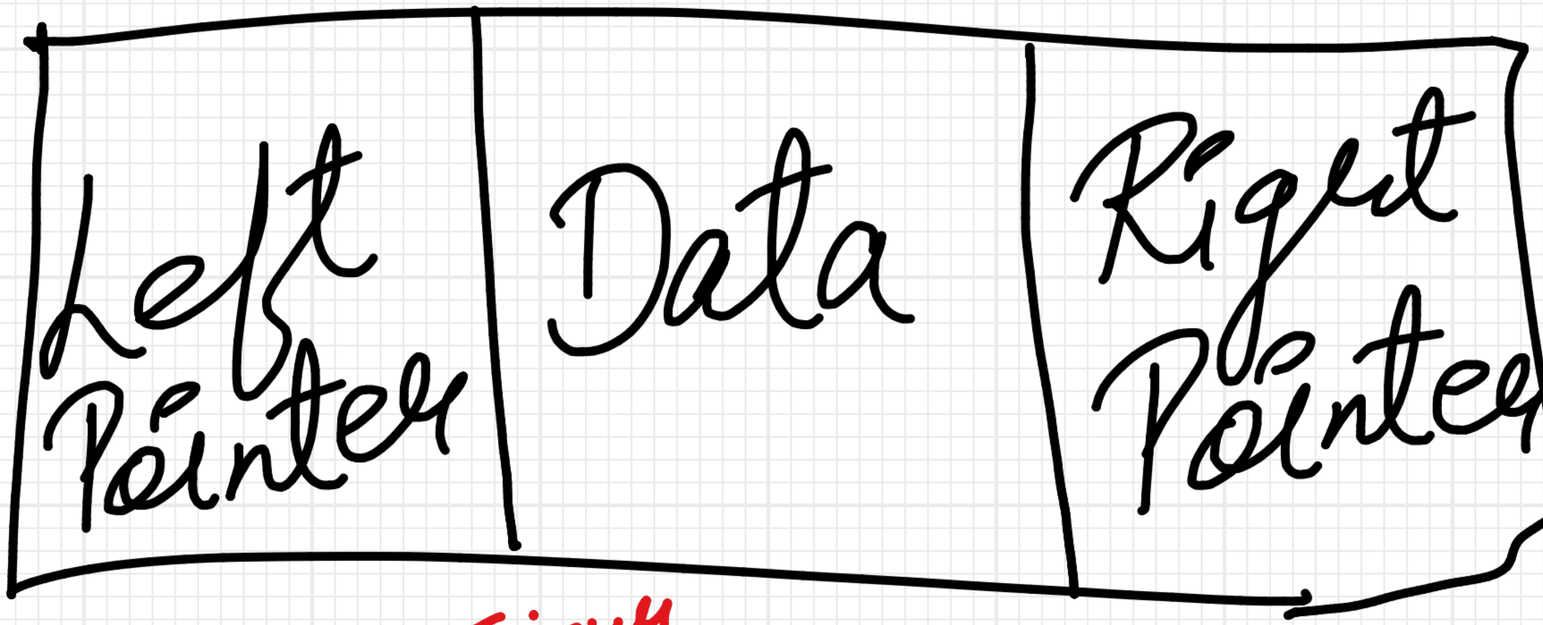


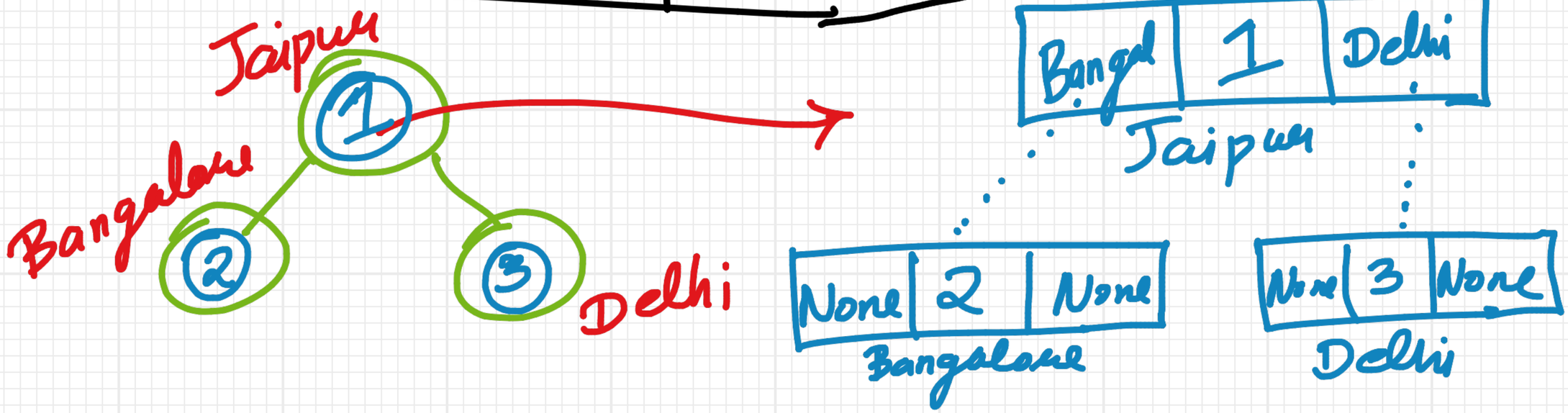

1
3


2


3

Node


Left Pointer
Data
Right Pointer


Jaipur
Bangalore
Delhi
1
2
3
Bangal
1
Delhi
Jaipur
None
2
None
Bangalore
None
3
None
Delhi

Binary Search Tree (B.S.T)

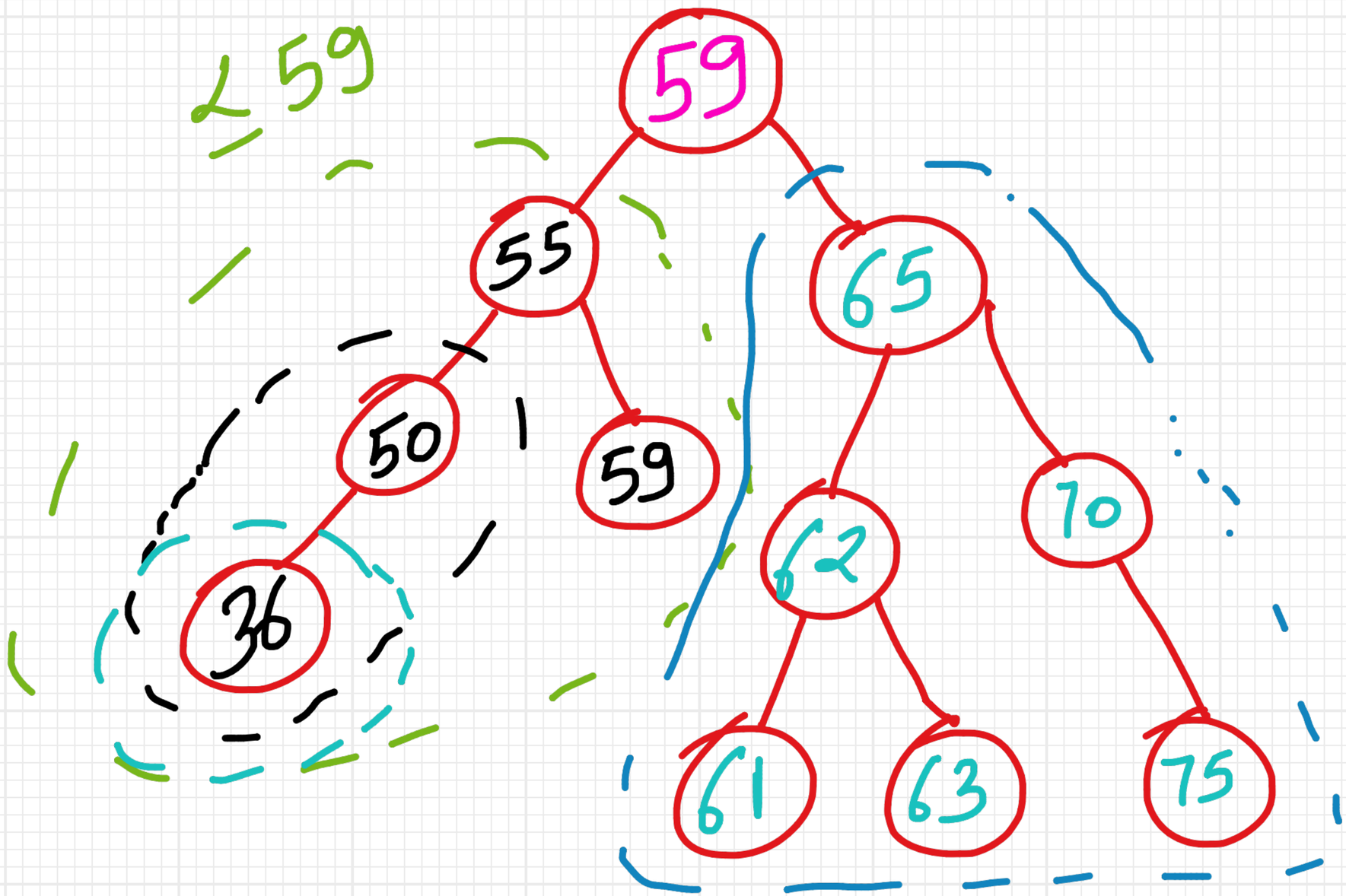
A binary search Tree is a tree in which for each node, value/data of all the nodes in left subtree is lesser/equal. And value/data of each node in Right subtree is greater.

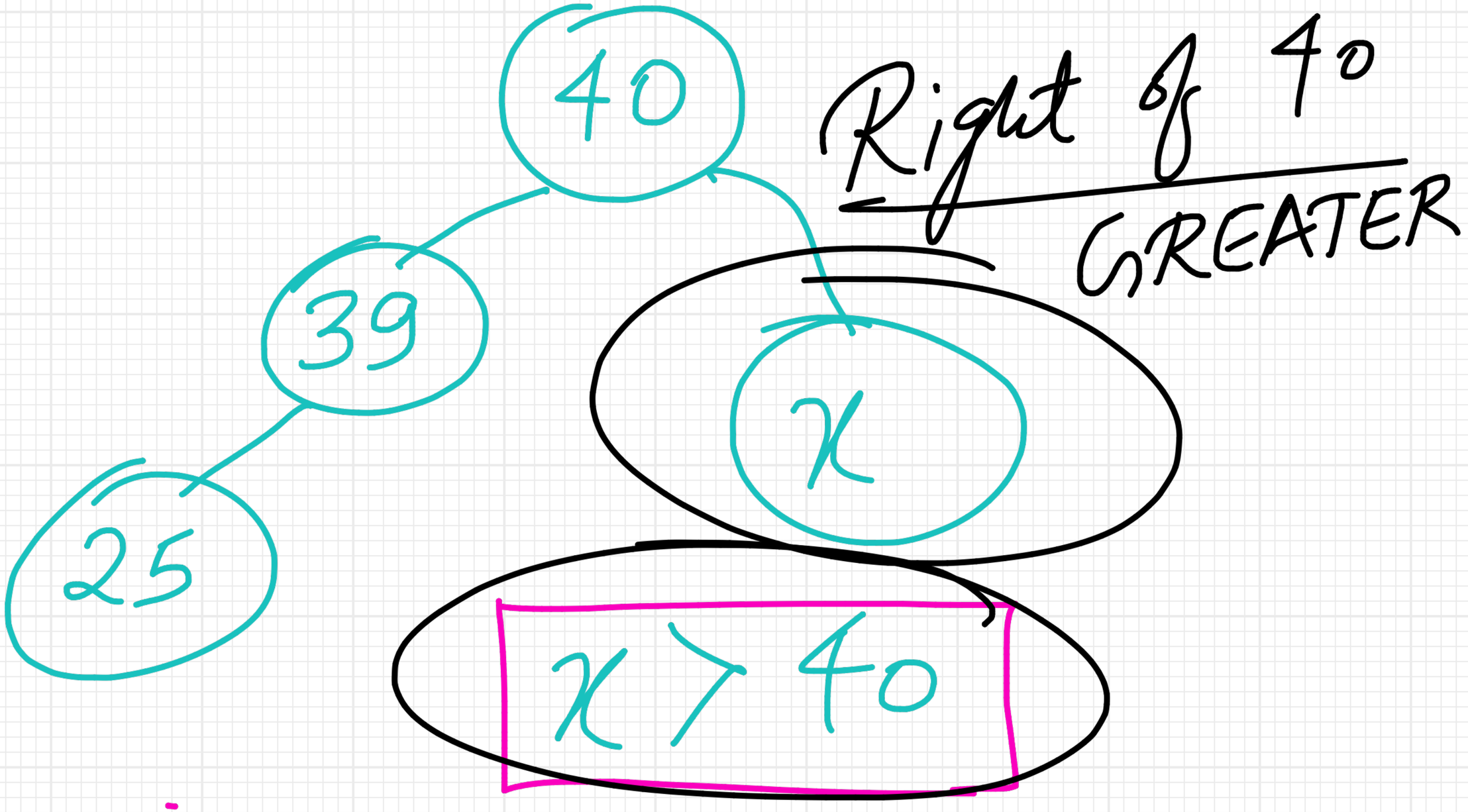
B.S.T is a tree which for each node :-

① Value of all the nodes in LEFT Subtree is LESSER/EQUAL

② Value of all the nodes in RIGHT Subtree is GREATER

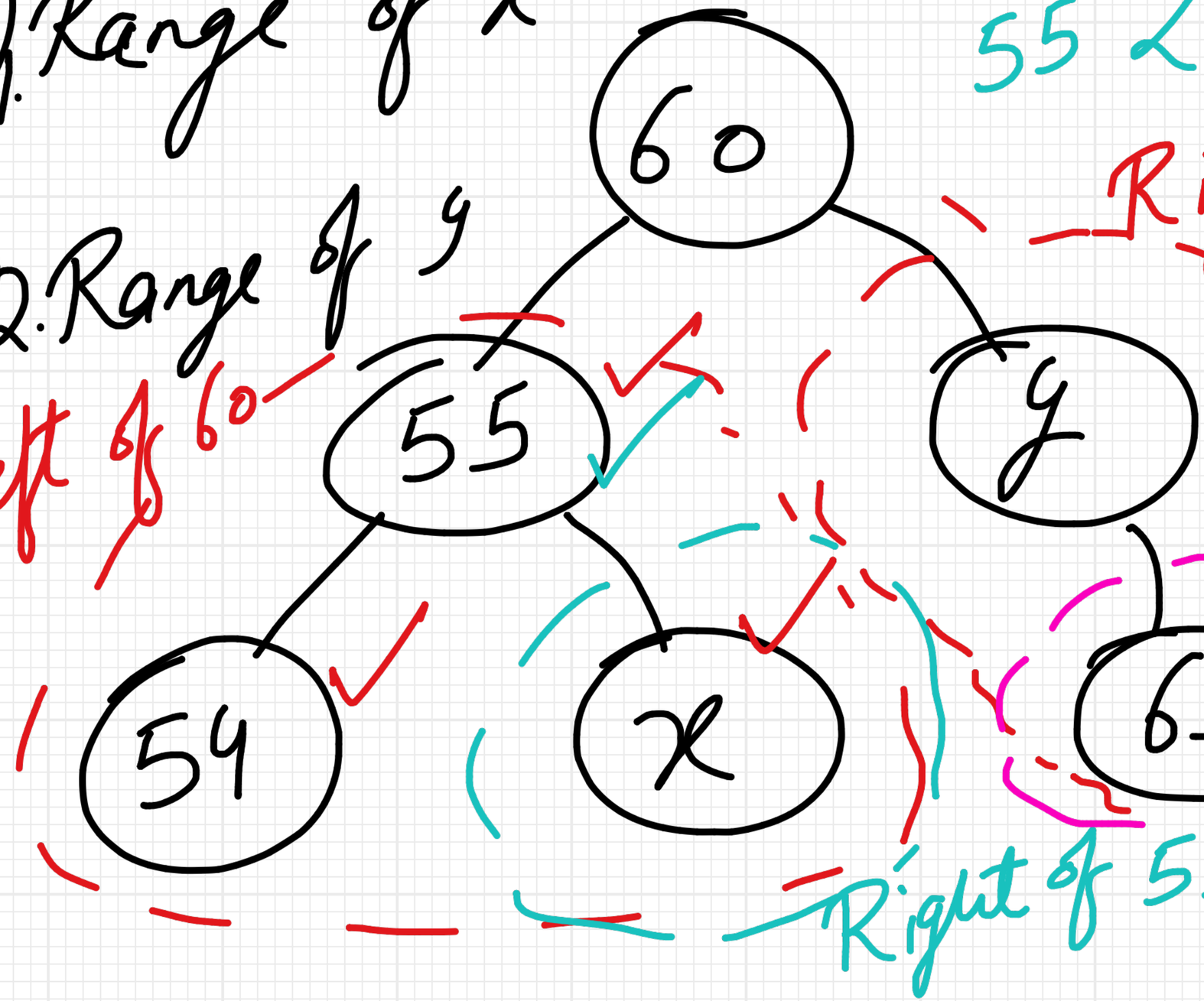
≤ 59




40
39
25
Right of 40
GREATER
x
x > 40

Q1. Range of x

Q2. Range of y



$55 < x \leq 60$

Right of 60

Left of 60

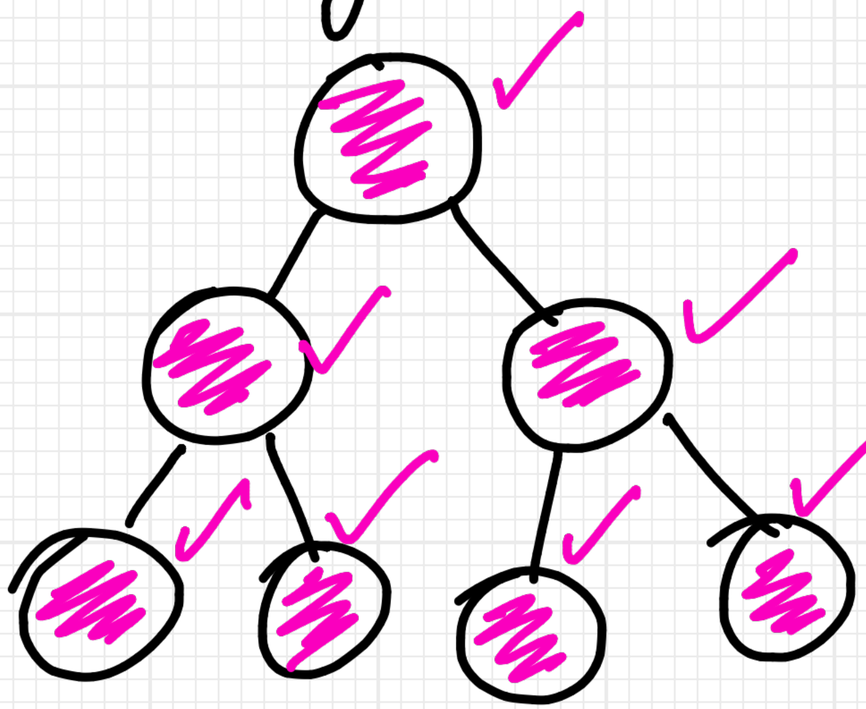
$60 < y \leq 69$

69 is right of y

Right of 55

✱ Traversal in B.S.T

Reading all nodes 1 by 1



Traversal

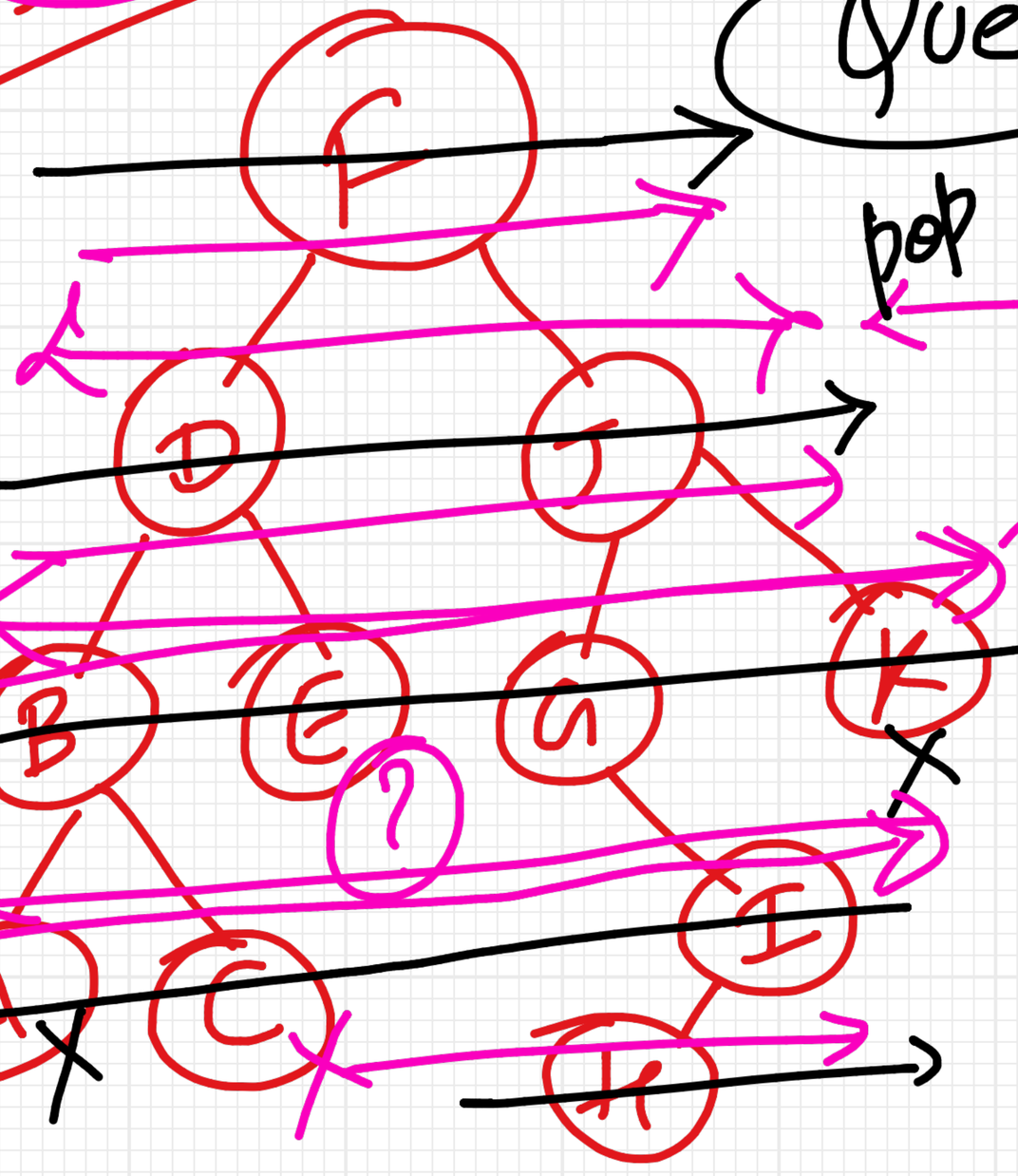
- BF
  Breadth First
- DF
  Depth First

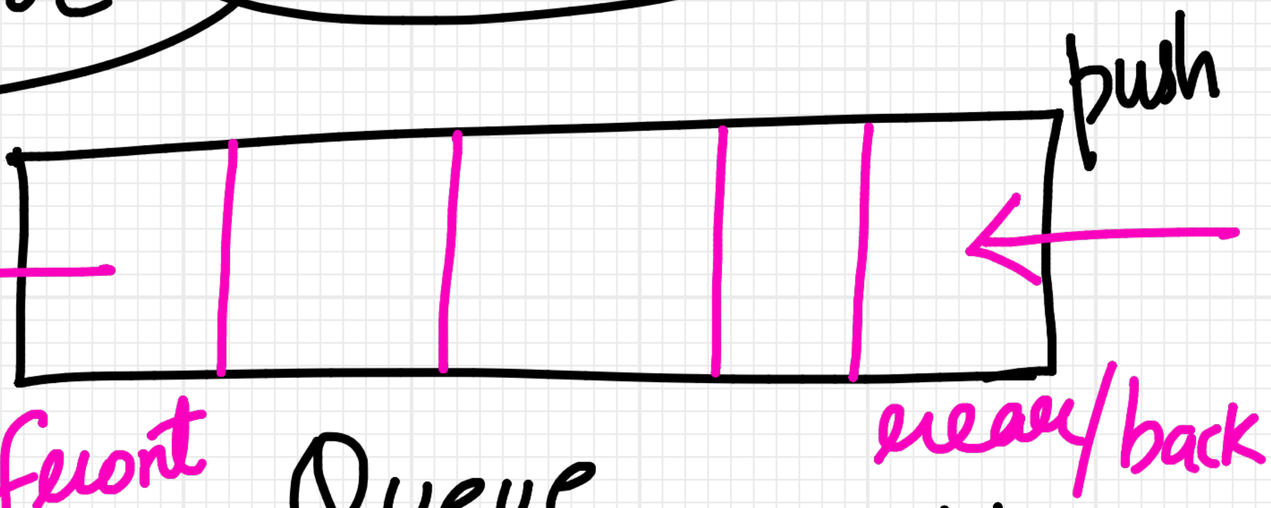
BFS

F D J B E G K A C I H

Queue

Use Queue ✓





Queue

✓ Since Q is empty now we can say we traversed the tree completely.

① Pop

② Push it's children

x = insert_in_bst(None, 50)

insert_in_bst(x, 33)

50

data

x.data?
= 50

x.left = insert_in_bst(x.left, 33)

None