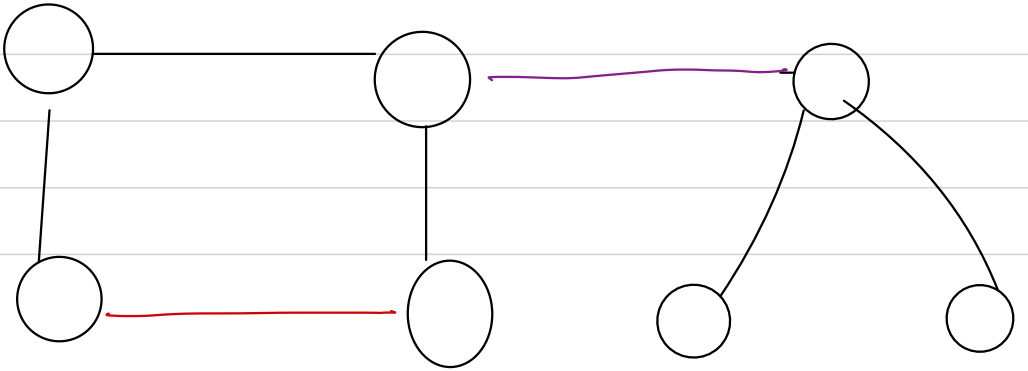
Graph → graph is a non-linear data structure, with two major elements, vertices and edges where vertices represent some real life entity and edges represent the relationship between them.

graphs can have cycle



Trees → Trees are non-linear data structure which represent hierarchy based information. Also trees are those undirected graphs which denot have a cycle.

→ acyclic
→ undirected
→ connected

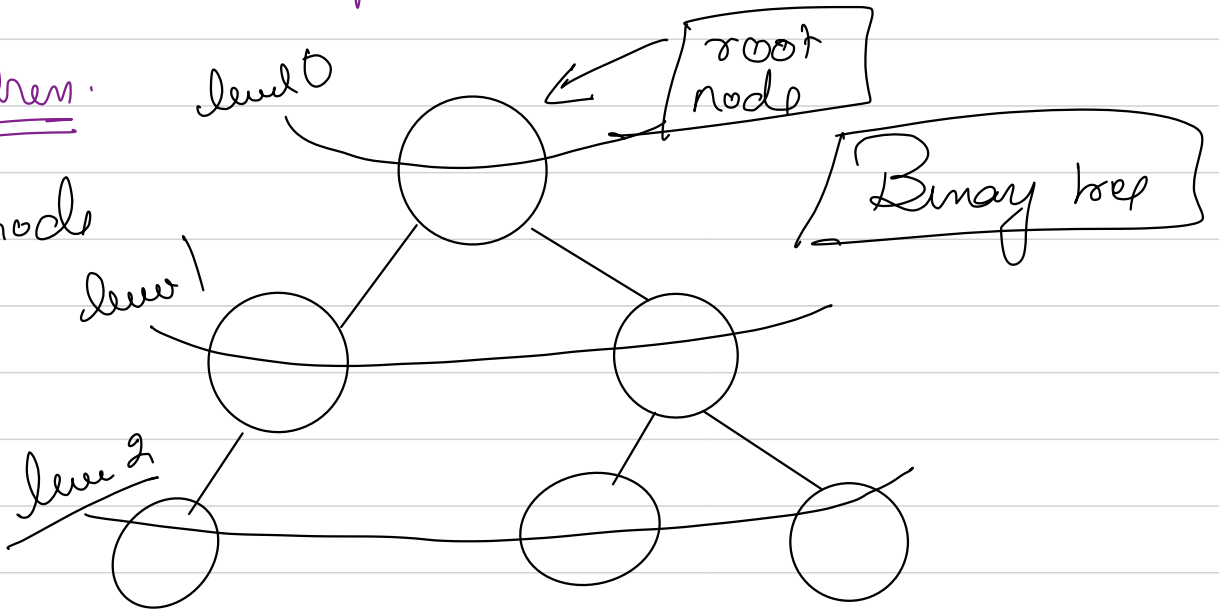
↳ trees represent, parent - child relationship

↳ trees are recursive data structure

→ Based on number of children

- → Binary trees
- → Ternary trees
- → quadnary tree
- ⋮
- → n-ary tree (generic tree)

↳ Binary Tree → these are defined as those trees where a parent can have atmost 2 children.

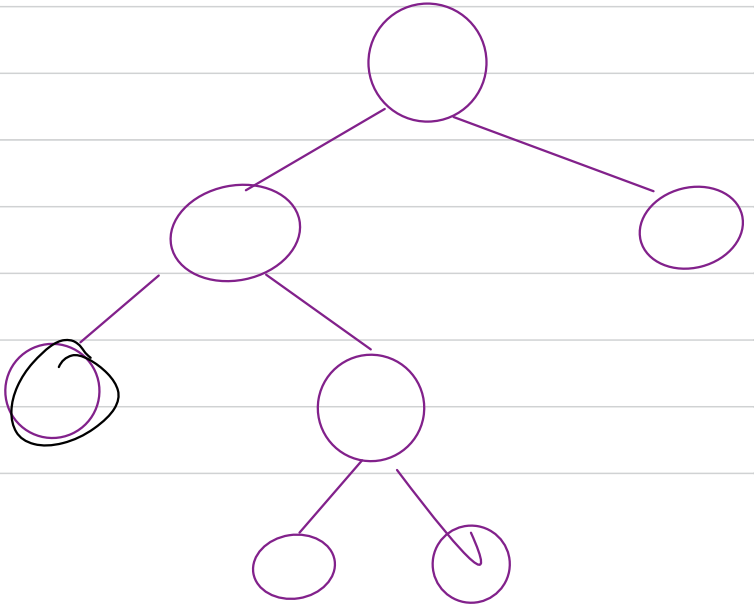


root → a node without parent

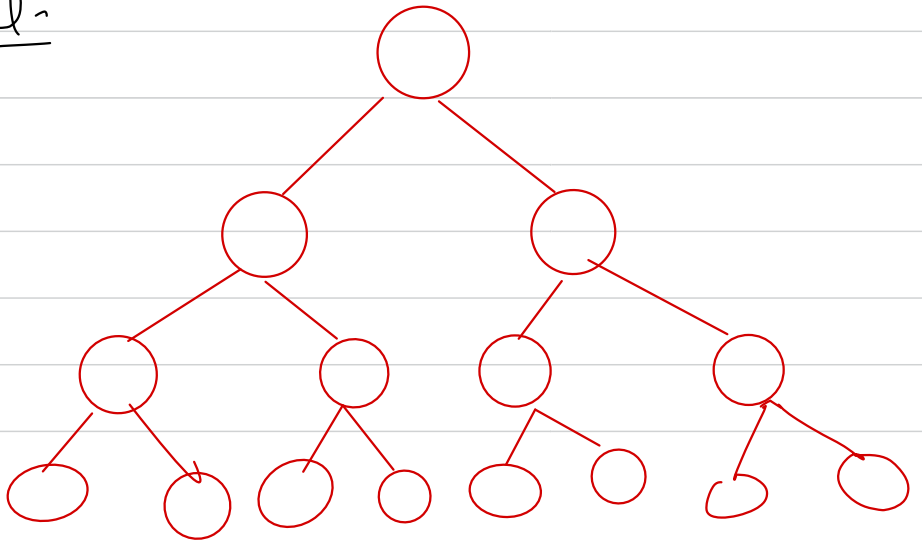
leaf node → a node with 0 child

Different Types of binary trees-

1. **full binary tree** → In a full binary tree every node has either 0 or 2 children.

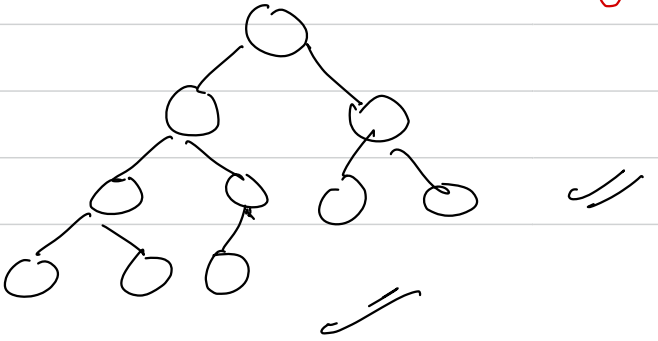


(2) Perfect Binary tree → A binary tree is a called perfect if all the internal nodes have 2 children and all the leaf nodes are at same level.



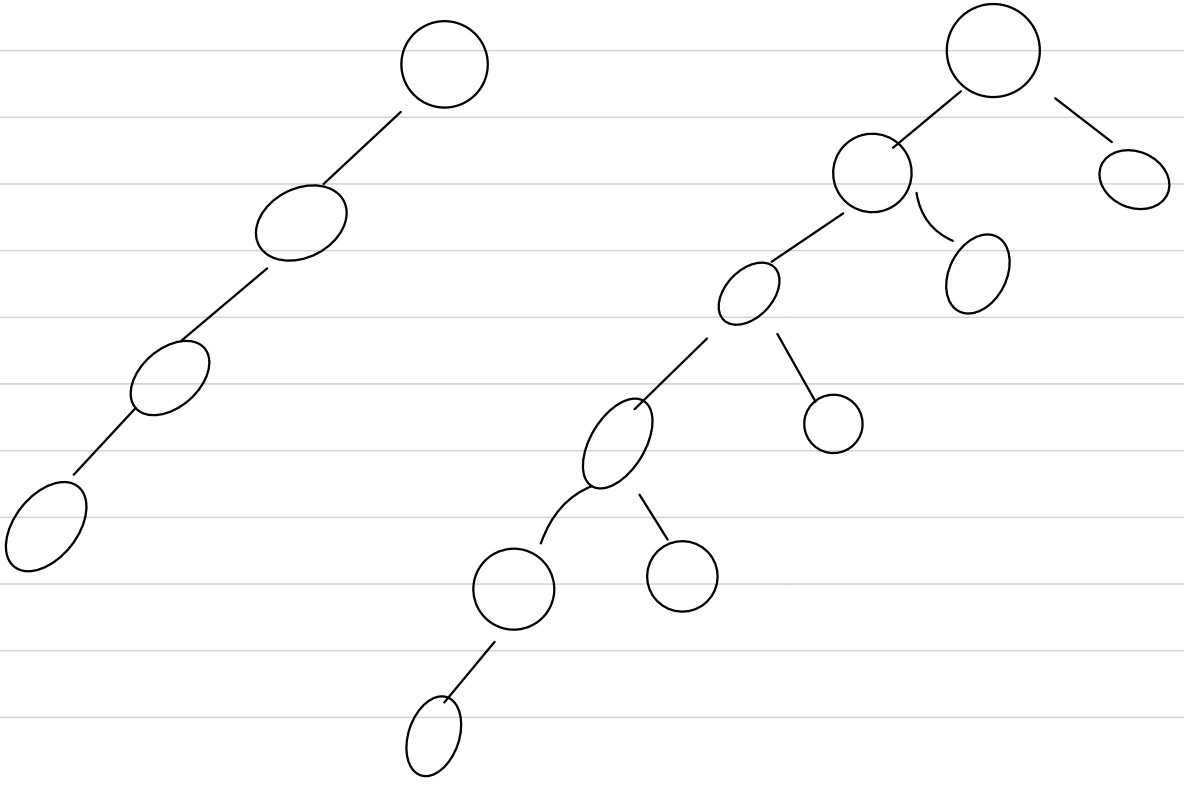
(3) Complete Binary Tree → A complete BT is a BT if all levels are completely filled except the last level, and the last level is filled from left to right without leaving any node.

(all nodes of last level is as left as possible)

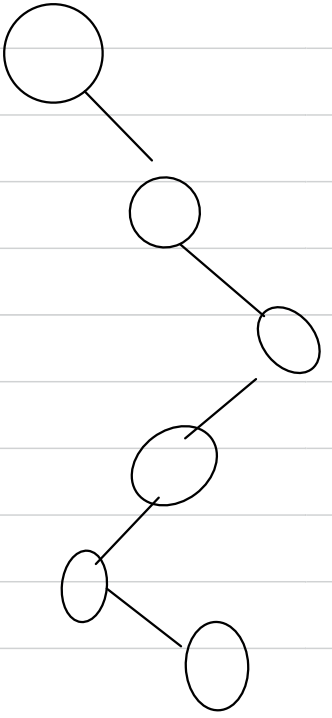


full capacity fully

4 Skewed Binary tree → tilted in one direction

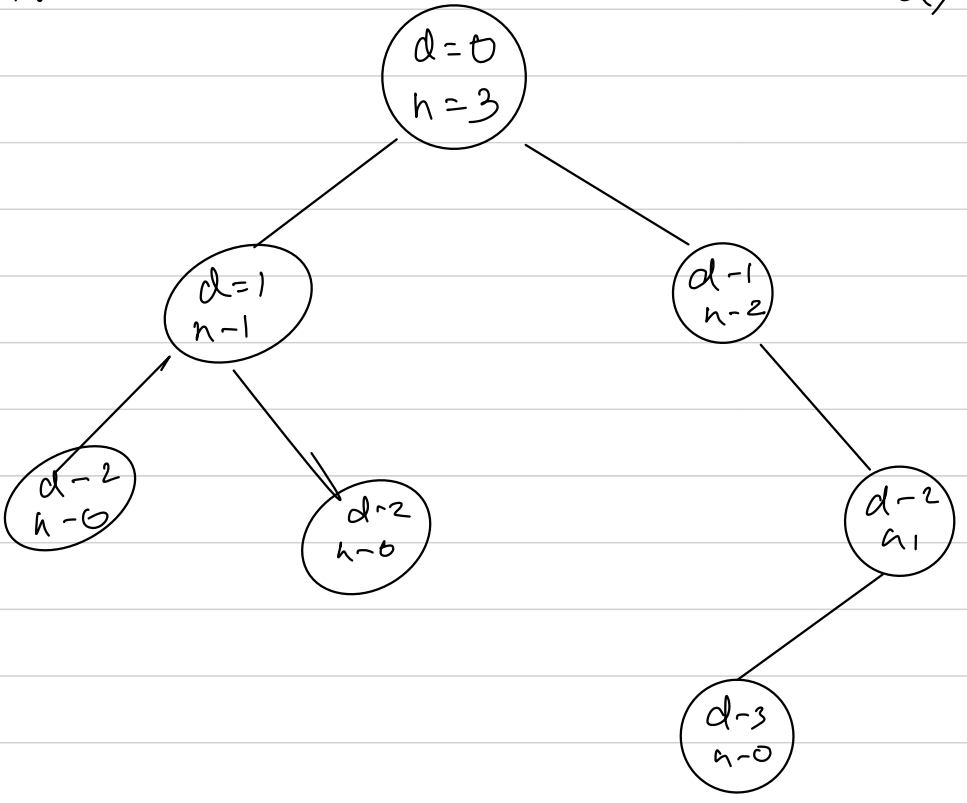


(5) Degenerate Tree → Every node except leaf has one child

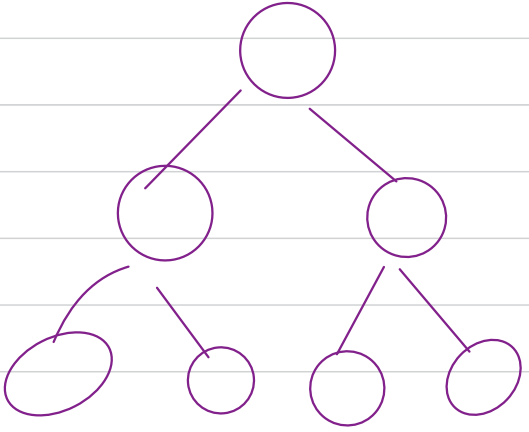


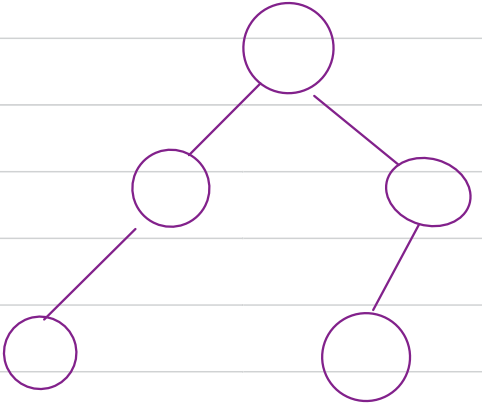
1) Depth

2) Height



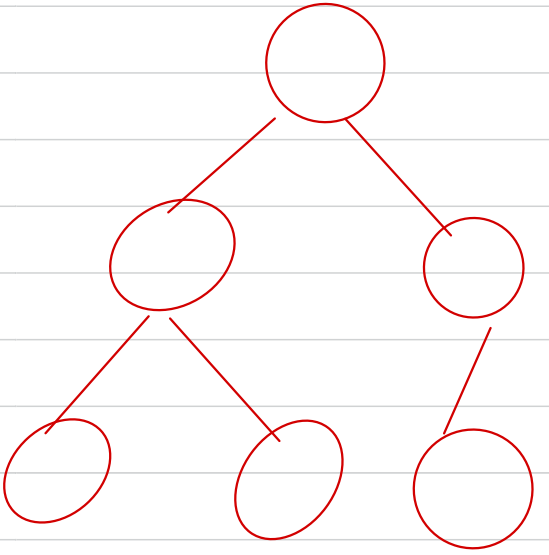
6) Balanced BT → from the root node, the absolute difference of height of left subtree & right subtree is atmost 1. And this should be true recursively for left & right ST also





How to read a Tree??

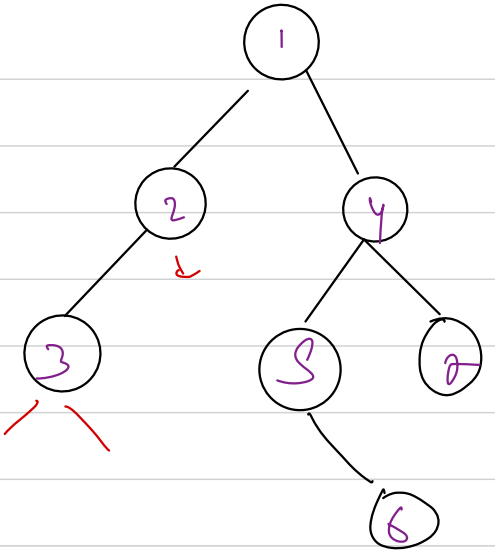
- Preorder traversal
- Inorder traversal
- Postorder traversal



Preorder → 1) Read the node
2) Read the LST
3) Read the RST

Inorder → LST
Node
RST

Postorder → LST
RST
No



level order → 1, 2, 4, 3, 5, 7, 6

[1, 2, 3, -1, -1, -1, 4, 5, -1, 6, -1, -1, 7, -1, -1]

Preorder → 1, 2, 3, 4, 5, 6, 7

Inorder → 3, 2, 1, 5, 6, 4, 7

Postorder → 3, 2, 6, 5, 7, 4, 1

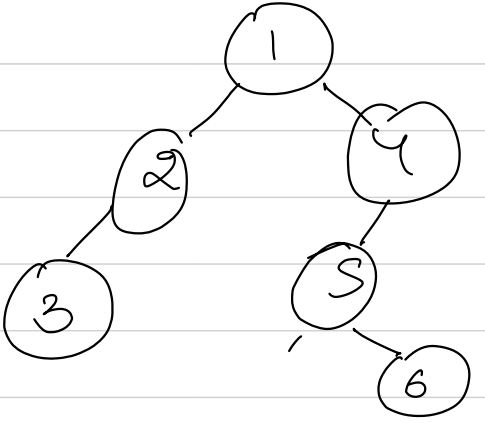
BT

data
leftchild
rightchild

Recursively

↳ We will build a tree with special preorder

1, 2, 3, -1, -1, -1, 4,
5, -1, 6, -1, -1, 7, -1, -1

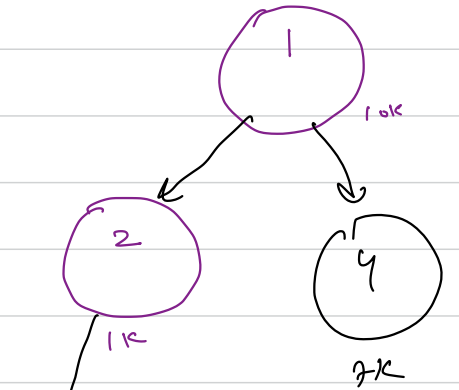


```python
def buildBTRec():
    d = int(input())
    if d == -1:
        return None

    root = Node(d)
    root.left = buildBTRec()
    root.right = buildBTRec()
    return root
```

root = buildBTRec()

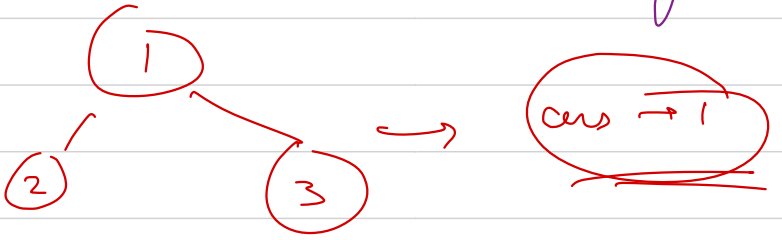
1, 2, 3, -1, -1, -1, 4,
5, -1, 6, -1, -1, 7, -1, -1

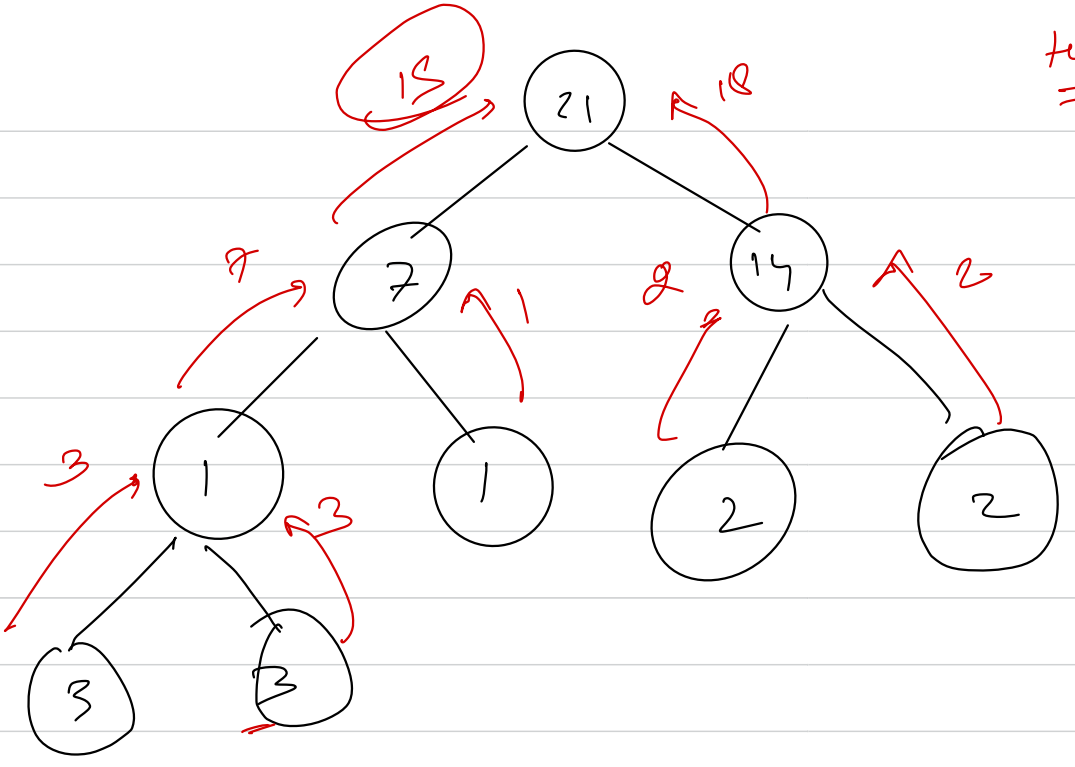


Write a function to calculate height of a BT.

Tilt of a Binary Tree → It is the absolute difference btw sum of all left subtree nodes & all rest nodes. if a node doesn't have a left child the sum of LST is considered 0, & same for right.

Given a BT, return the sum of every node's tilt.





diff = 6+3

9

9