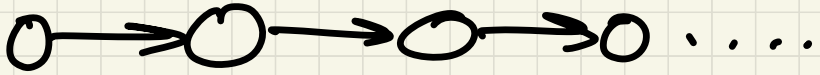
Trees

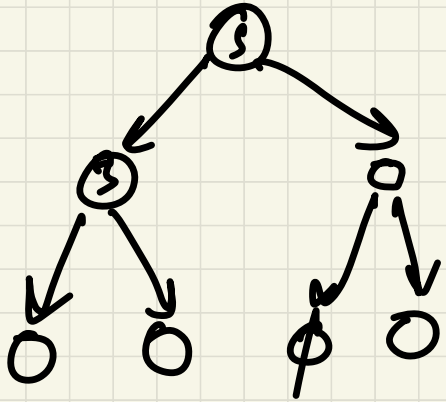
Pre-req:

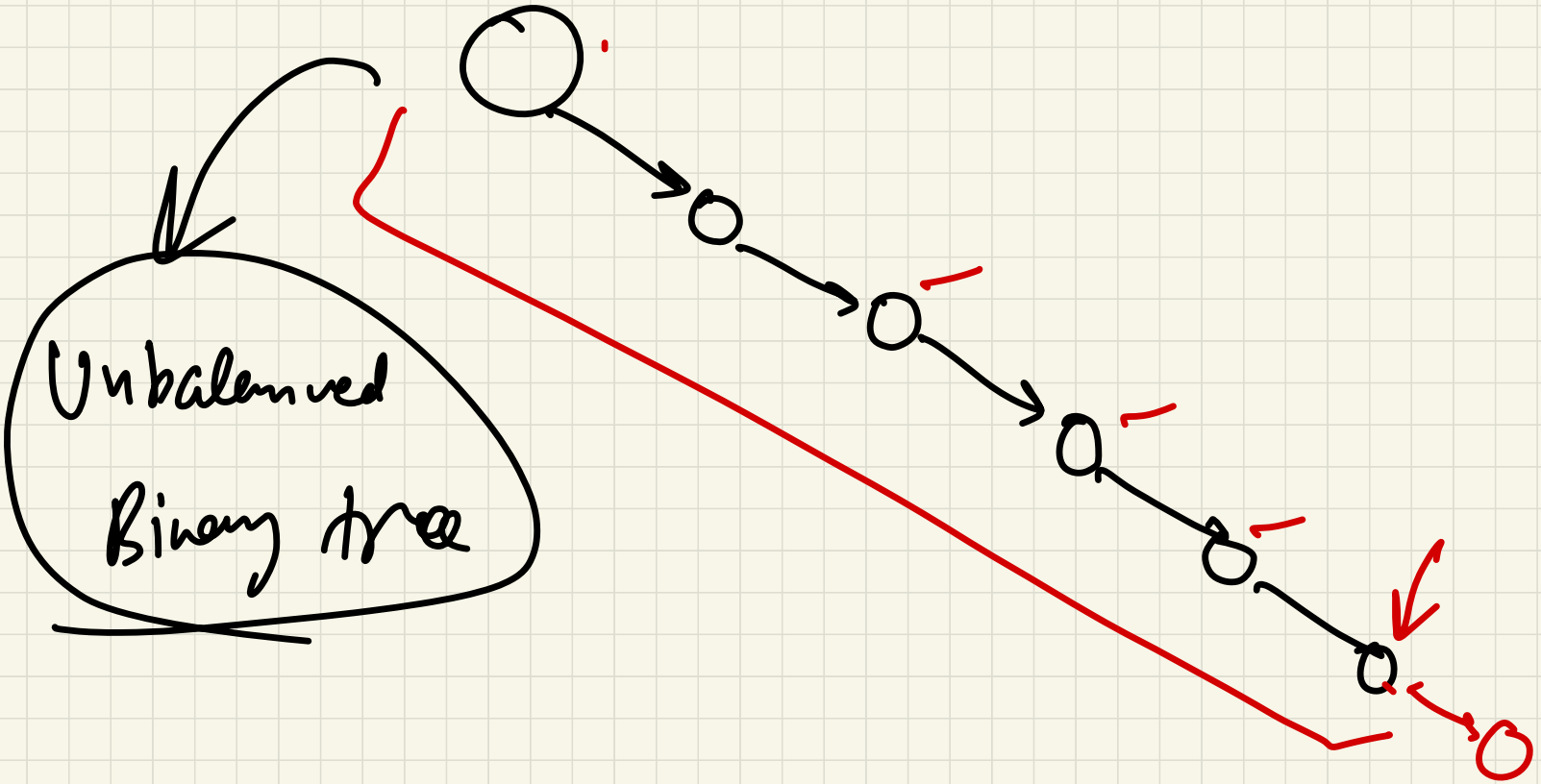
1. Recursion ✓
2. OOP ✓

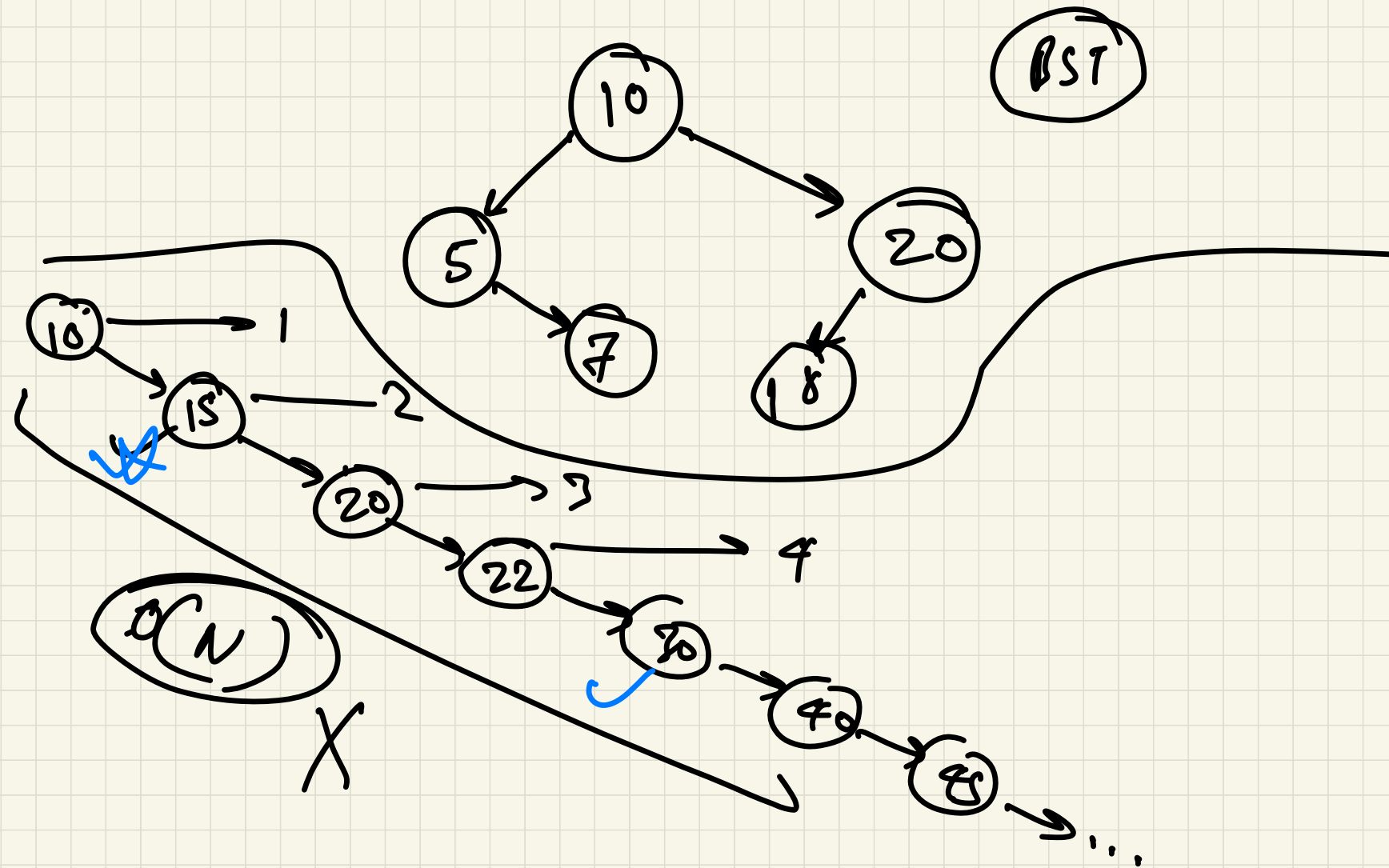
Why ?

- $O(\log N)$
- Ordered storage
- Cost efficient






Unbalanced
Binary tree


BST
10
5
20
7
8
10
1
15
2
20
3
22
4
O(N)
X
30
40
45
...

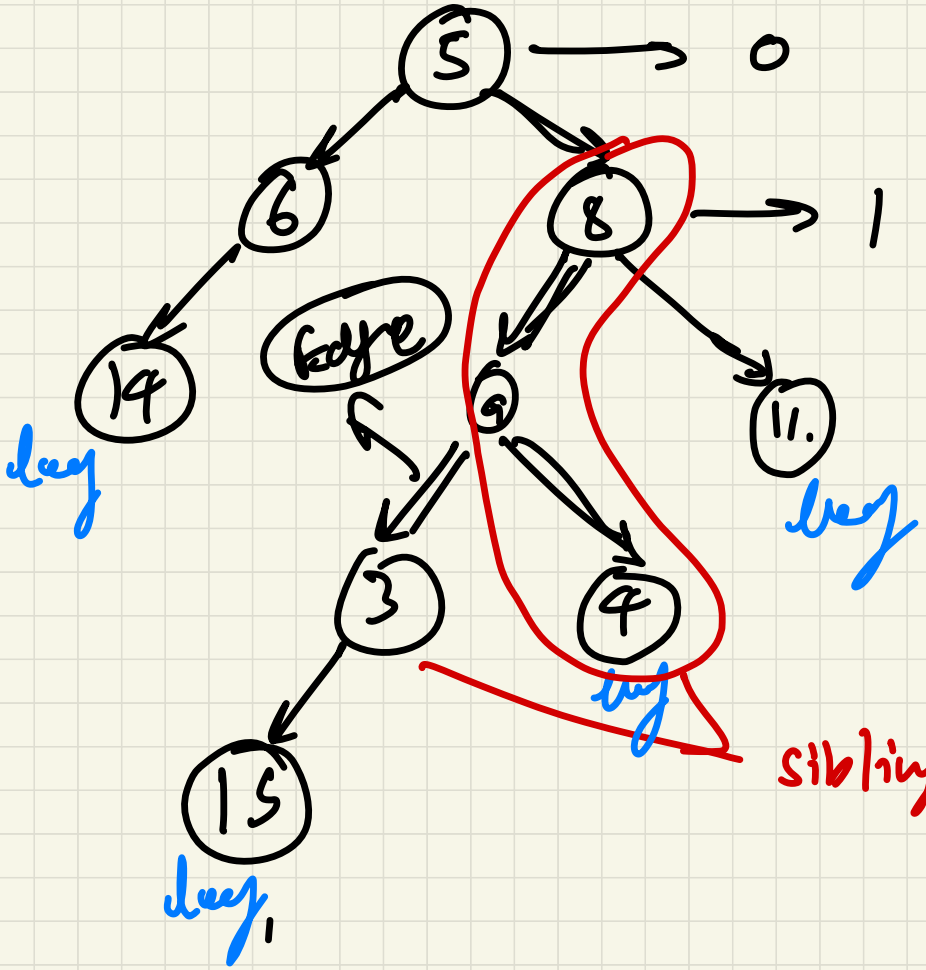
How to solve?

Self balancing BT.

Future concept

Where is it used?

1. File systems
2. Databases
3. Algorithms/ networking
4. maths
5. DTs → ML
6. Compression of files
7. Future DS's




```
Node :
    int value
    Node next
```

BT:

```
Node:
    int value
    Node left
    Node right
```

max(2, 4, 2, 3)

Properties:

1. Size = Total number of Nodes
2. Child & Parent
3. Siblings
4. Edge
5. Height → Max no. of edges from the node to leaf node.
6. Leaf nodes
7. Level → Sub height of root − Height of node.
   root level = 0

8. Degree
   0, 1, 2

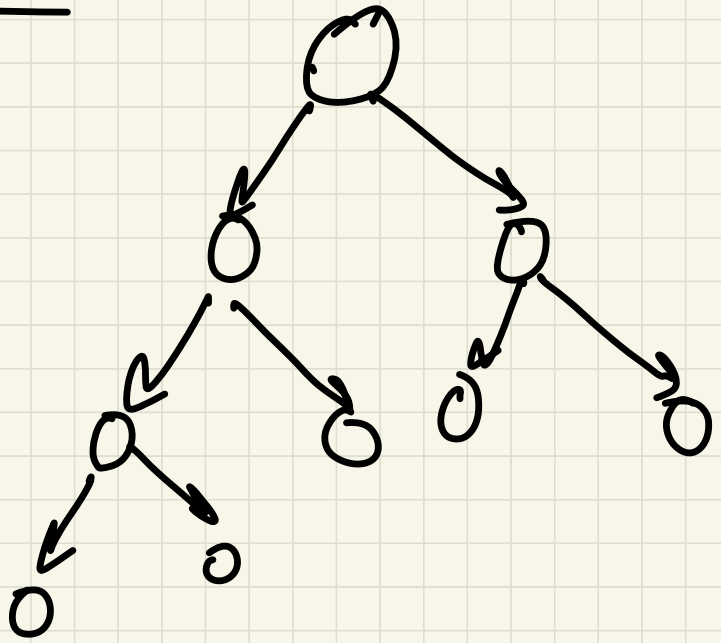
⑧ Ancestor & descendant

Types of binary tree:

① Complete binary tree

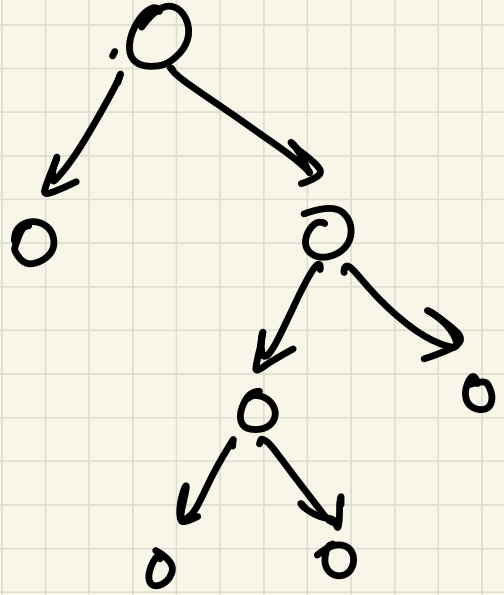
→ All level full

→ Last full from L to R →



② Full bt / strict bt:

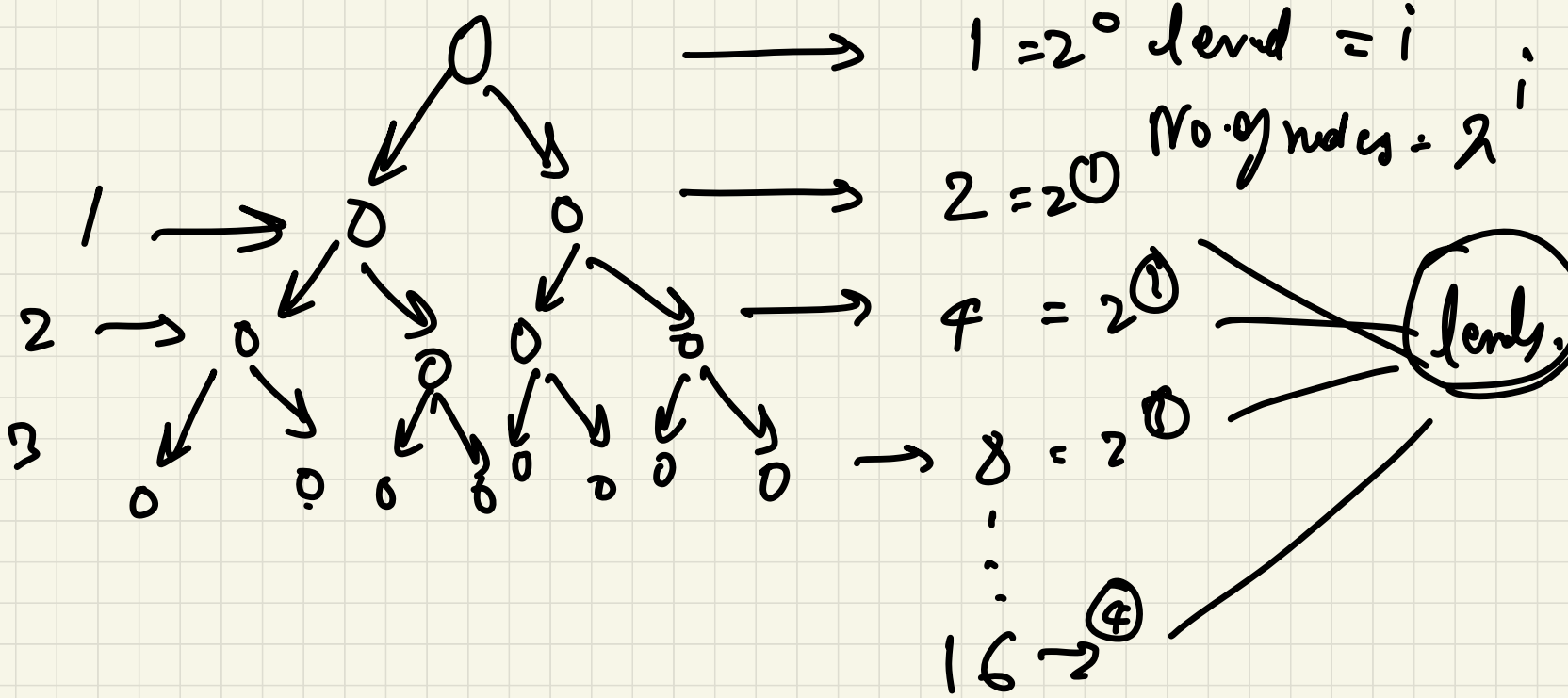
— Either 0 or 2 children



Use case: compression / Segment trees.

③ Perfect BT

→ All levels are full

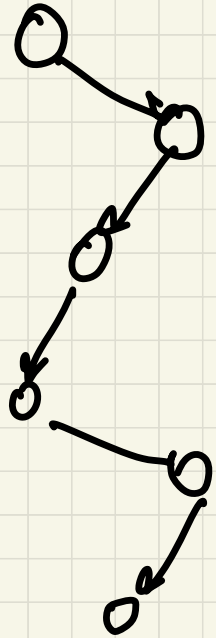


④ Height Balanced → Avg height $O(\log n)$

⑤ Skewed BT

Height $(O(N))$

like linked list



⑥ Ordered BT

Every node has some property.

Example: BST

Properties that will help you in some questions:

① Perfect BT, height $= h$

Total nodes $= 2^{(h+1)} - 1$

$$2^0 + 2^1 + 2^2 + 2^3 + \cdots + 2^h = \text{GP}$$

$$2^{(h+1)} - 1$$

(2) leaf nodes in Perfect BT $= 2^h$

$$2^{h+1} - 1 - 2^h = 2^h - 1$$

(3) N = no. of leaves

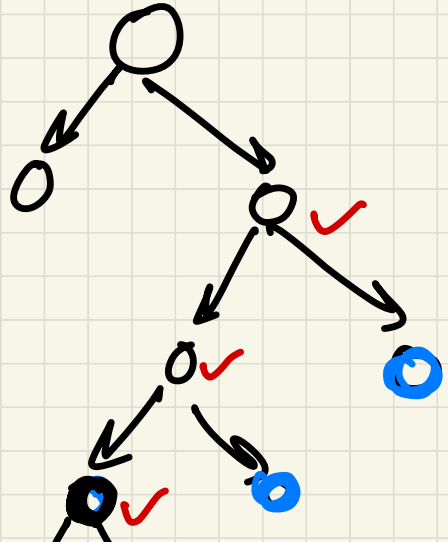
$\log N + 1$ levels atleast

N nodes $\rightarrow$ $\log(N+1)$ min levels.

④ strict BT, N → leaf nodes

N - 1 ⇒ internal nodes.

No. of leaf nodes = No. of internal nodes + 1



Internal = 4 (+1)

Leaf = 3 - 1 + 2

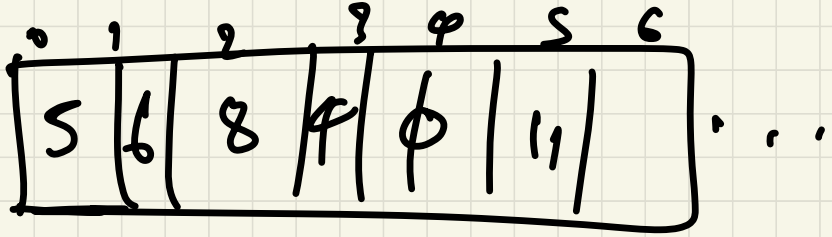
= 3 (+1)

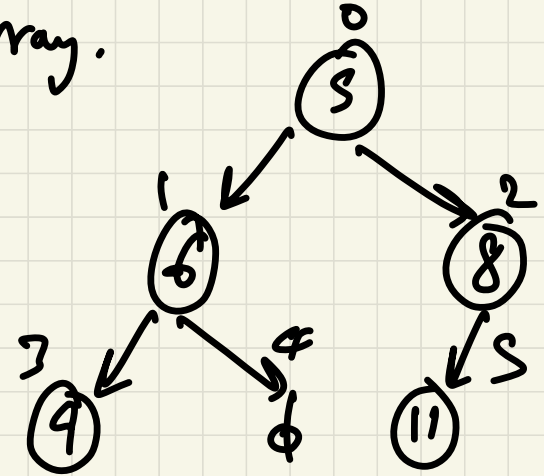
(5) No. of leaf nodes = 1 + No of internal nodes with 2 children (not including root)

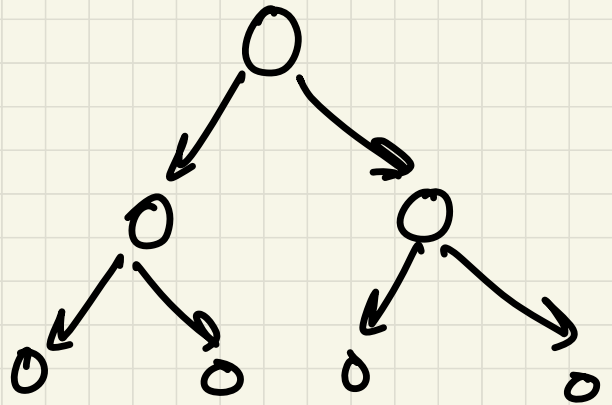
$4 = 1 + 3$ ✓

Implementation:

1. Linked representation
2. Sequential → using array.







1

2

2^2

2^3

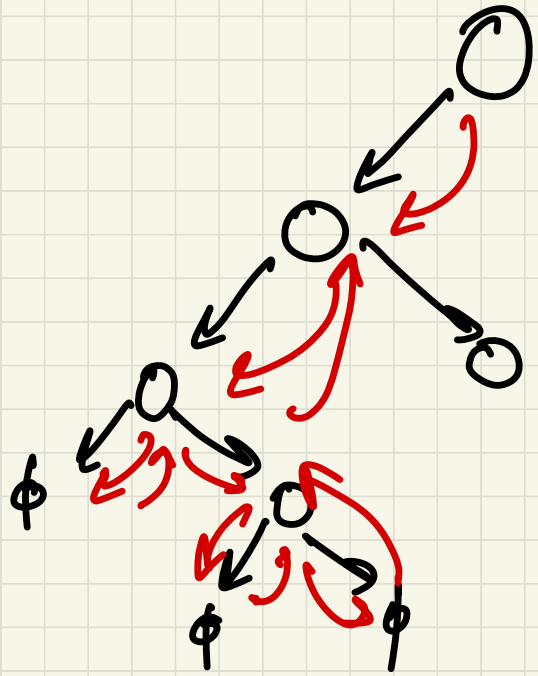
2^4

2^h

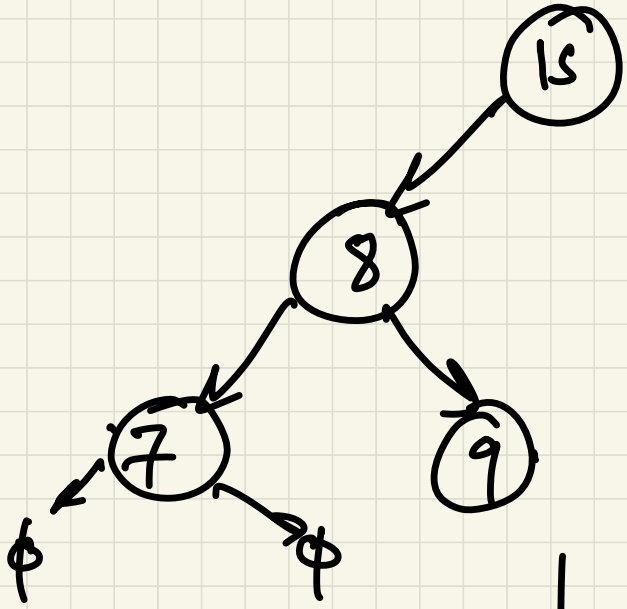
$N = 2^h$

$\log N = h \log 2$

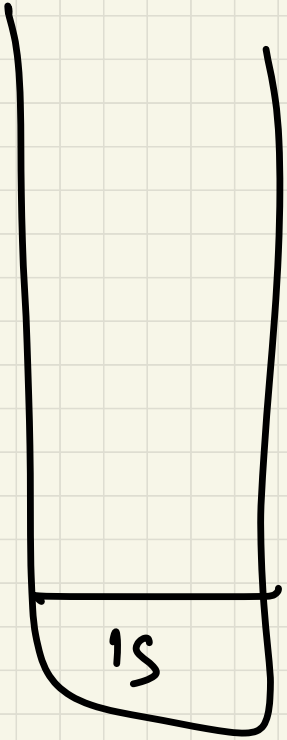
$h = \log_2(N)$

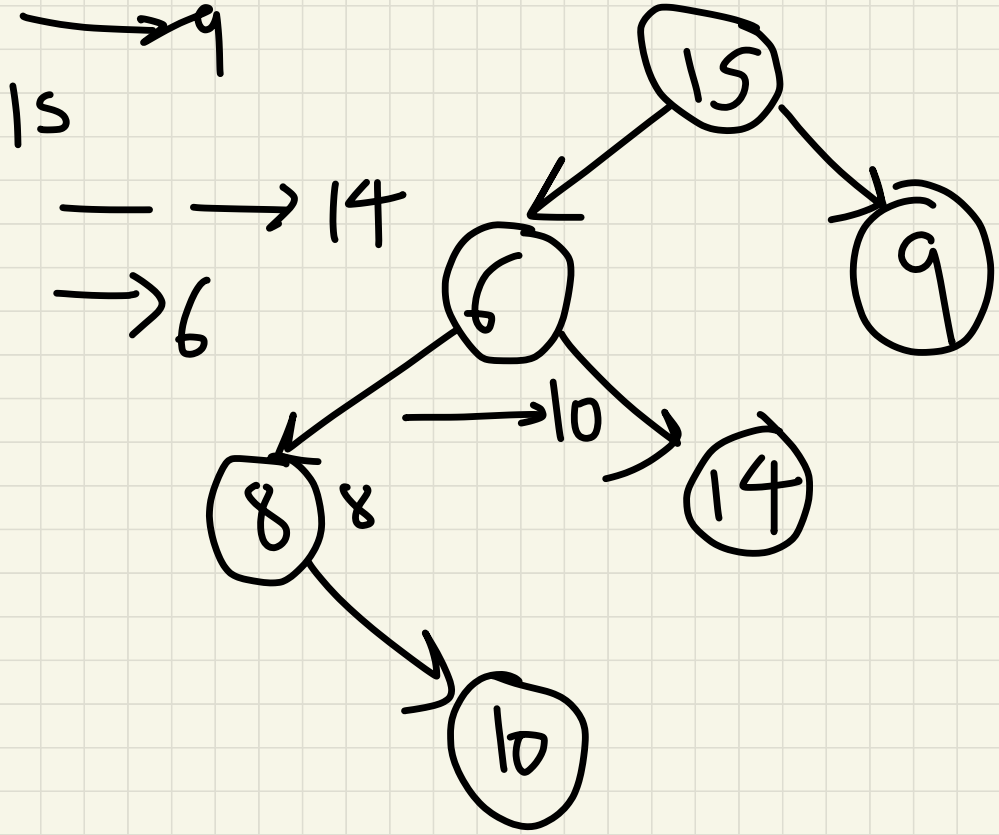


insert (
    insert (left)
    insert (right)

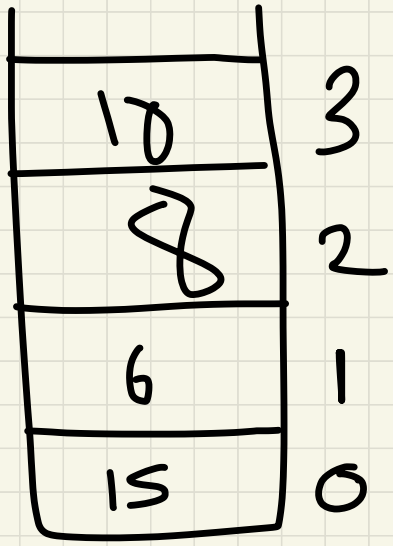


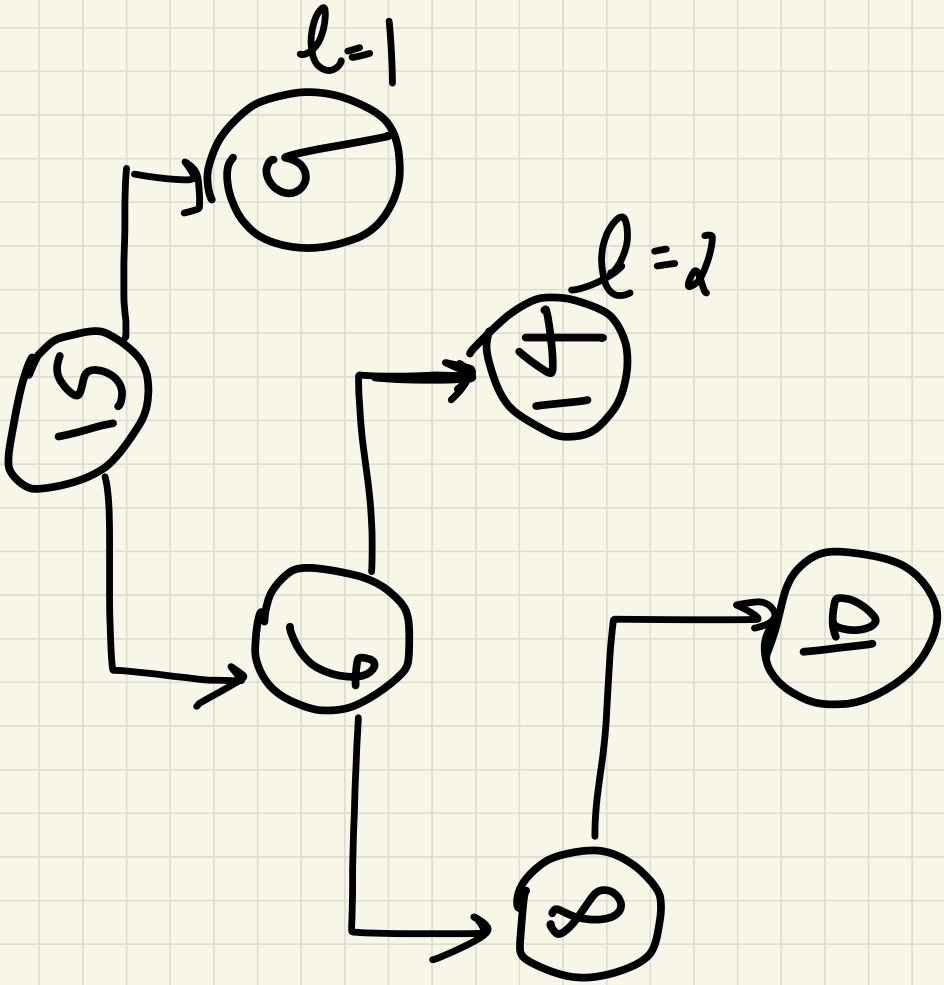
15
6
8
10
14



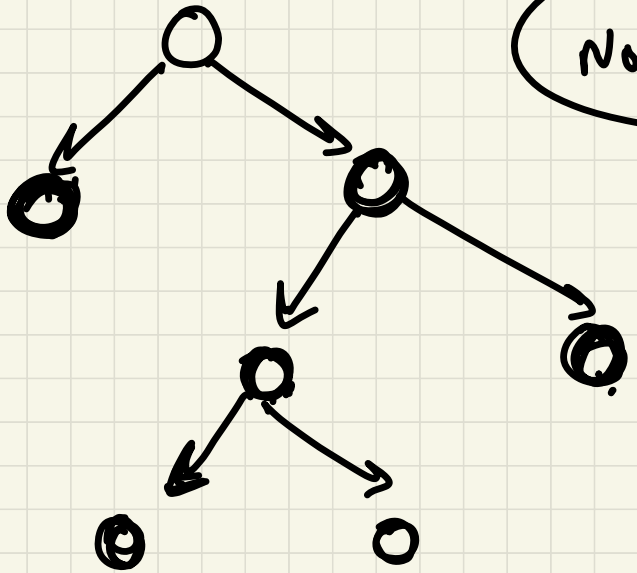


node = 15

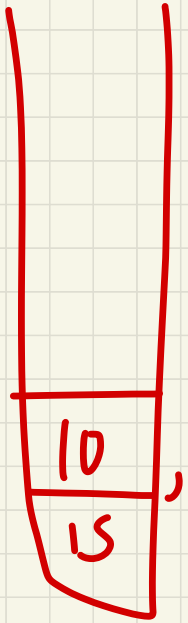


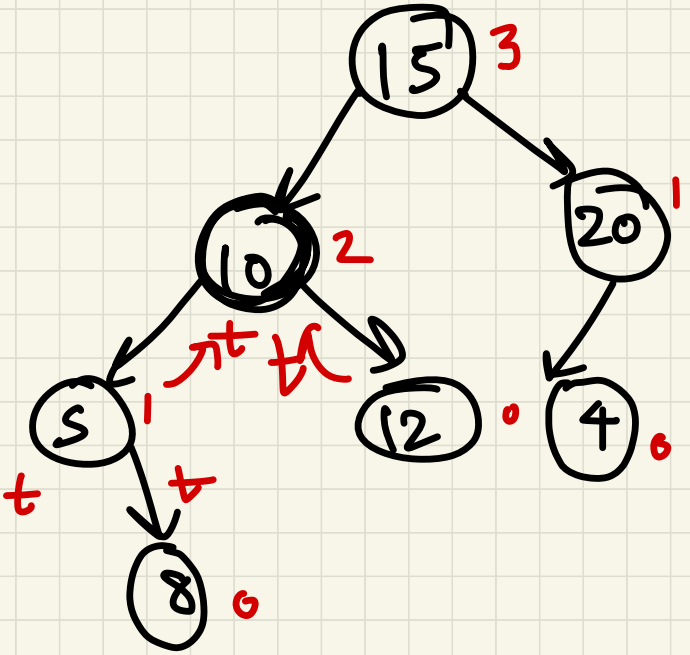

l=1
l=2

Binary Search Tree



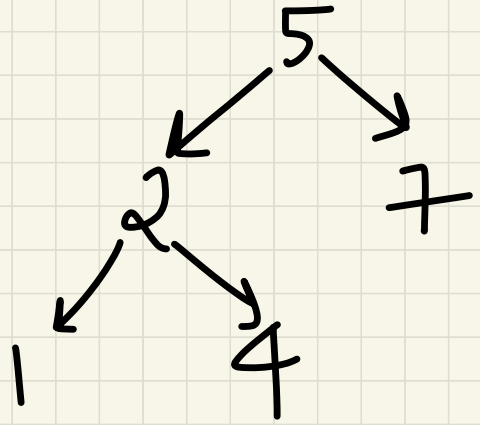
Not balanced

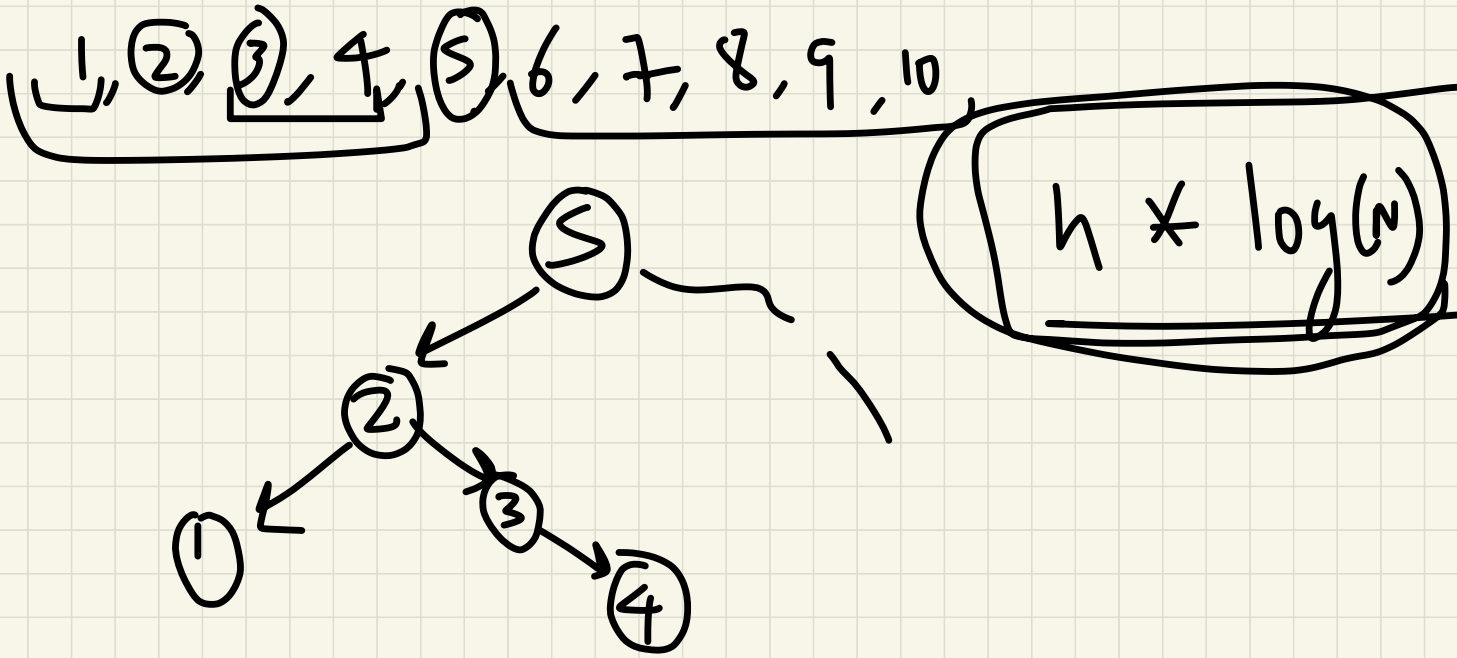




* Start from root.

```
Node {
    int value,
    Node left,
    Node right
    int height
}
```

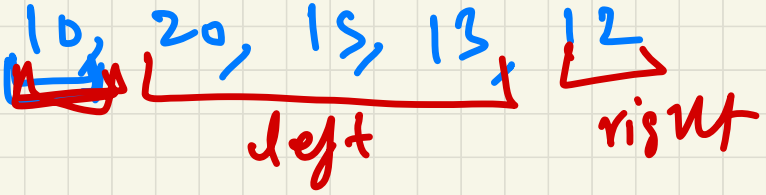

5
2
7
1
4


1, 2, 3, 4, 5, 6, 7, 8, 9, 10
h * log(N)
5
2
1
3
4

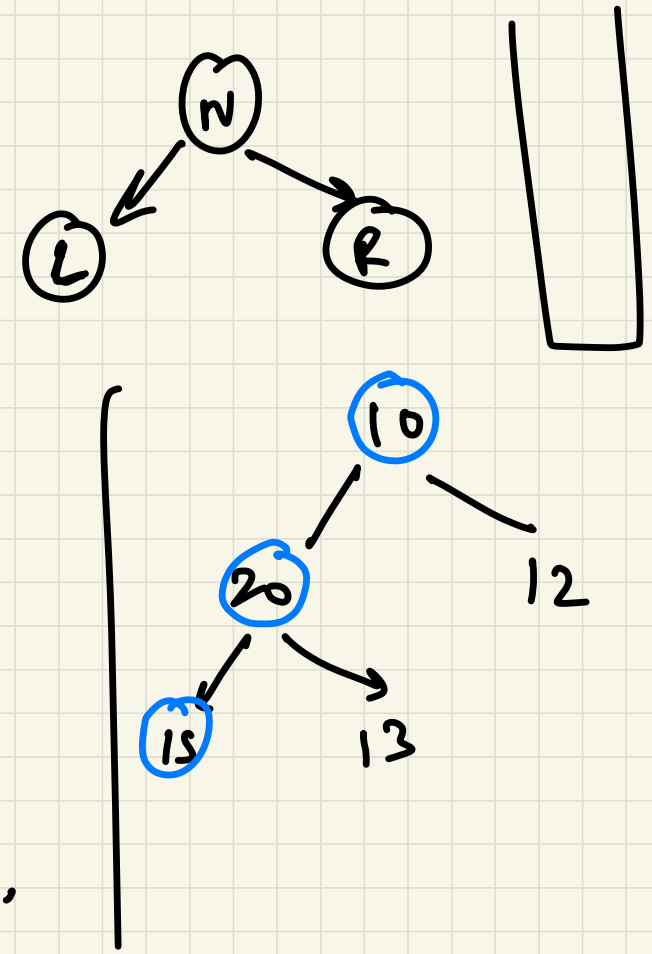
Traversal methods:

① Pre - order

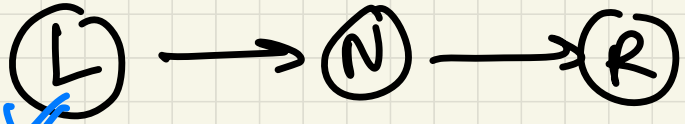
N → L → R

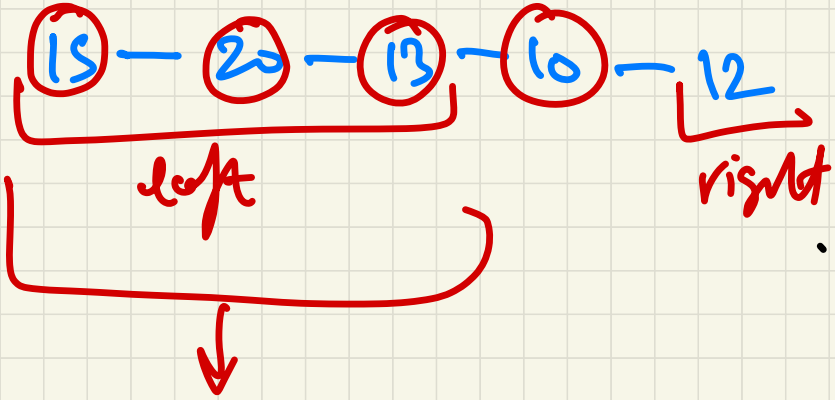


Used for evaluating math expression or making a copy.
Serialisation from string / array.

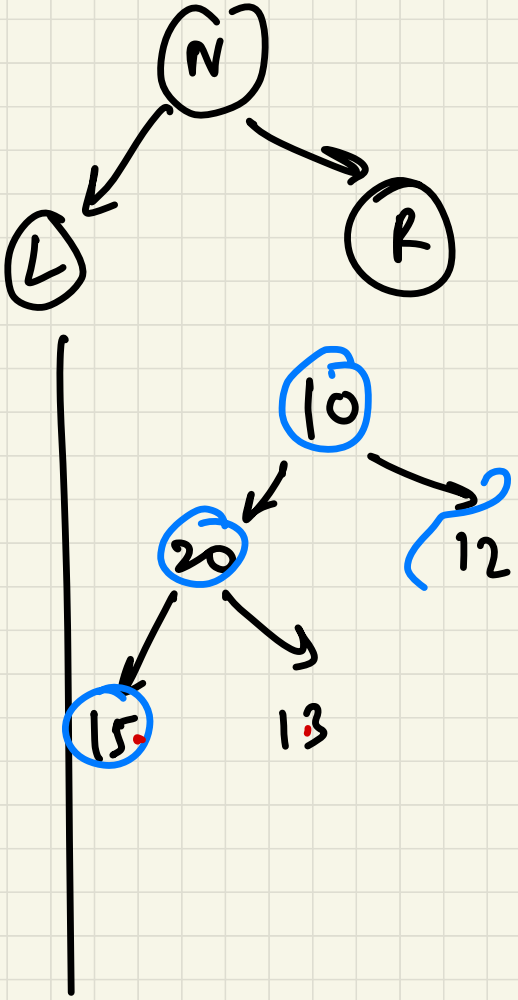


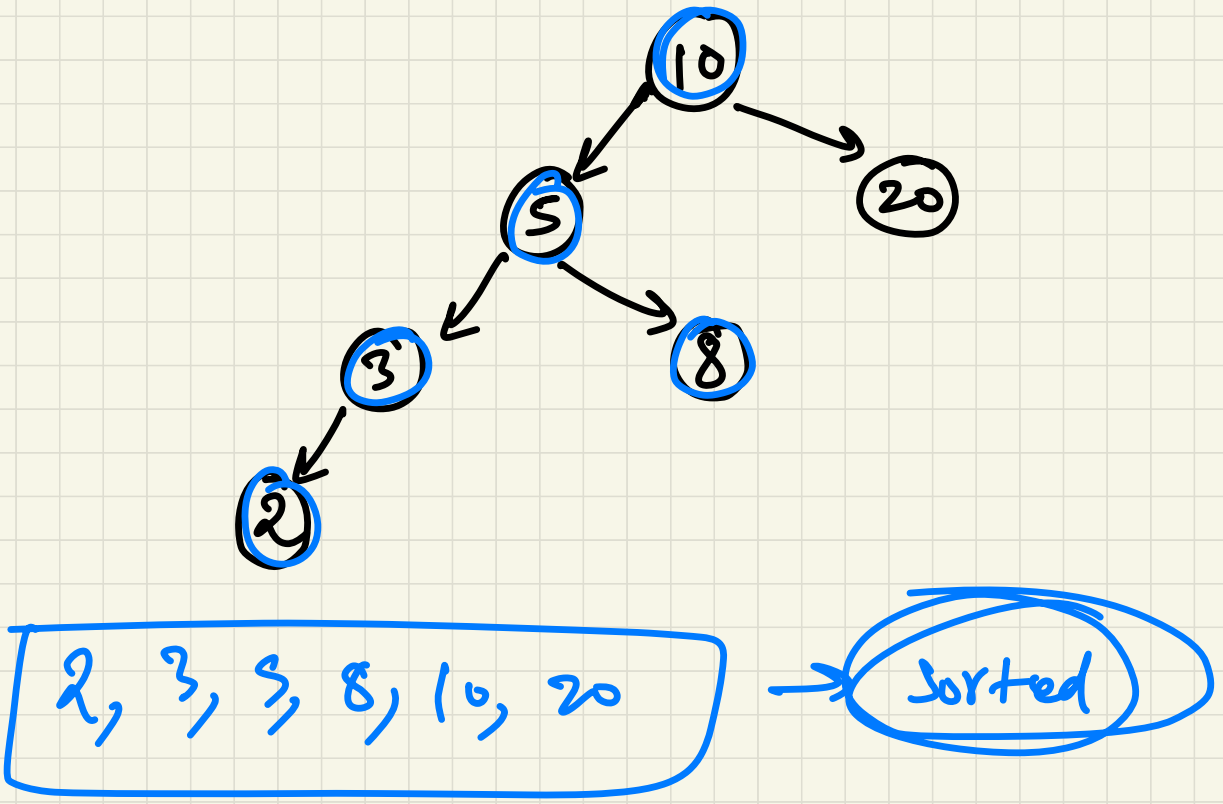
(a) Inorder



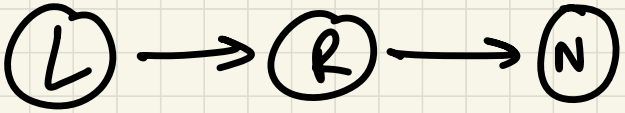


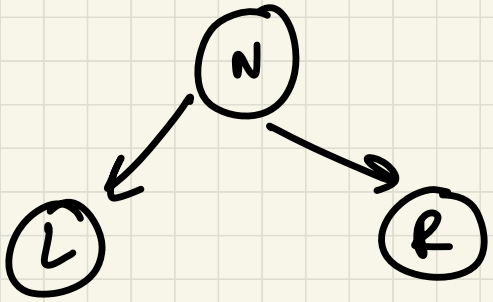
In BST, visit node in sorted manner.




10
5
20
3
8
2
2, 3, 5, 8, 10, 20
sorted

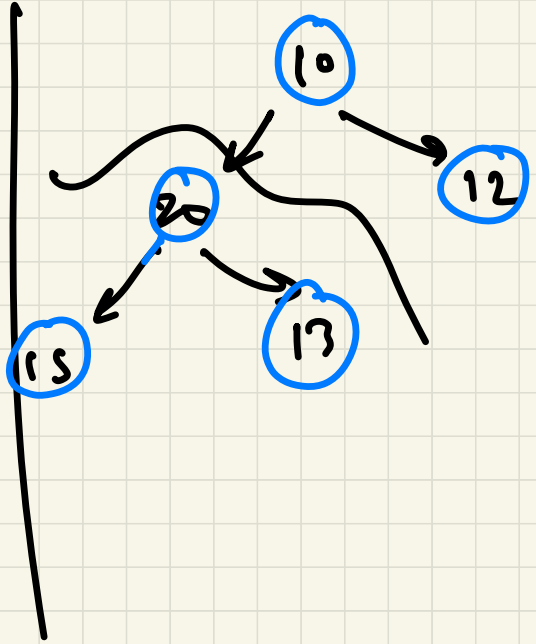
(3) Post order :





15, 13, 20, 12, 10

- delete binary tree
- bottom - up calculation



BFS & DFS or BFT or DFT