DSA through C++

Binary Tree

Saurabh Shukla (MySirG)

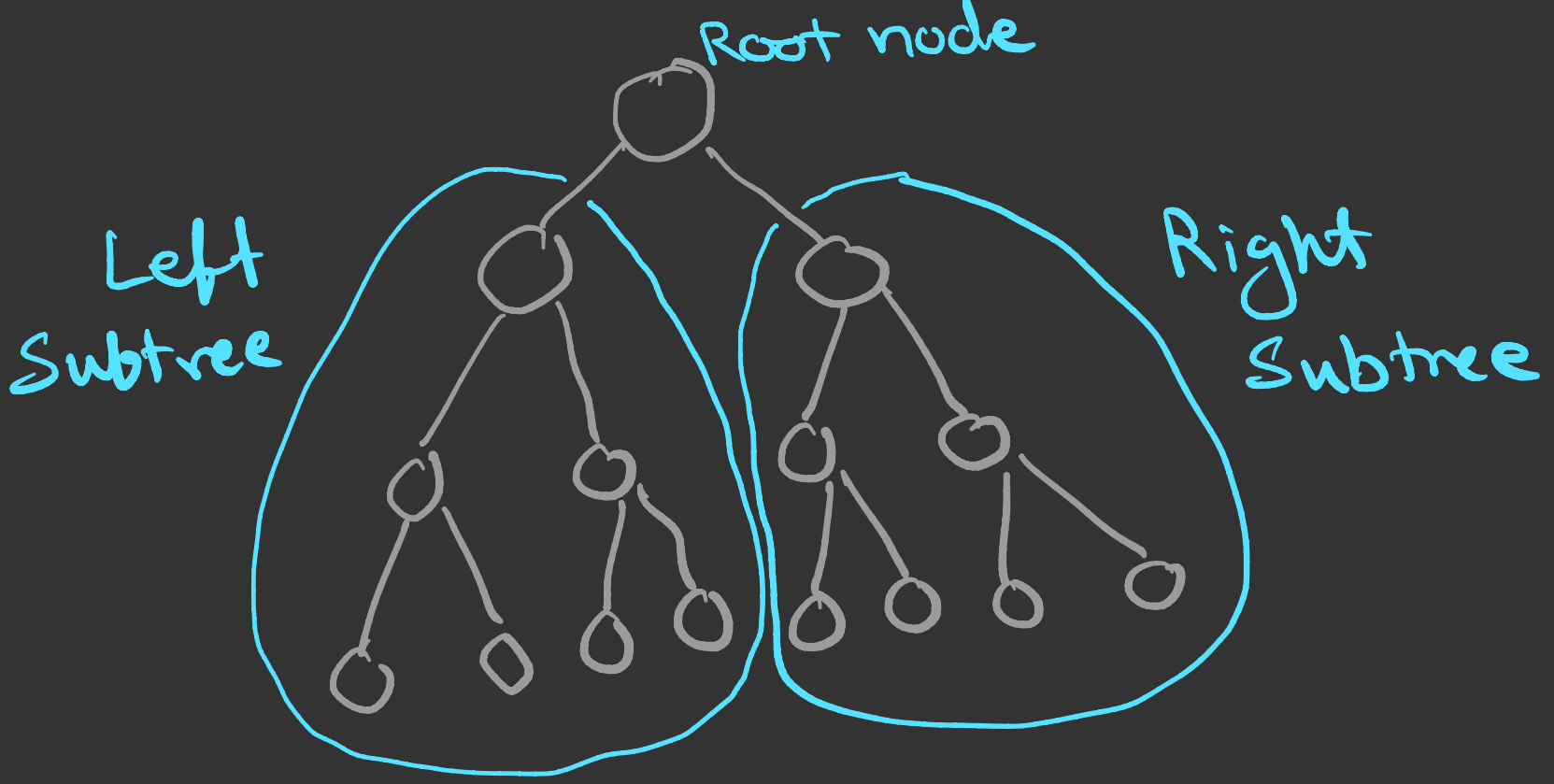
Agenda

1. Binary Tree
2. Complete Binary Tree
3. Almost Complete Binary tree
4. Strict Binary Tree
5. Representation of Binary tree

Binary Tree

A binary tree is defined as a finite set of elements, called nodes, such that

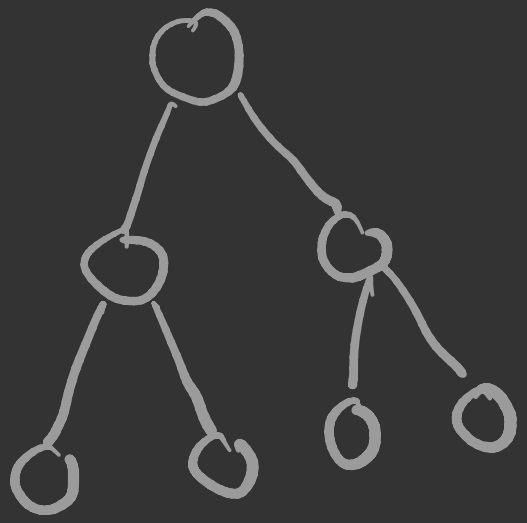
- T is empty (called the Null tree or empty tree), or
- T contains a distinguished node R, called the root of T, and the remaining nodes of T form an ordered pair of disjoint binary trees T_1 and T_2



Any node in the binary tree has either 0, 1 or 2 child nodes.

Complete Binary Tree

All levels are completely filled.



$L_0 \rightarrow 1$

$L_1 \rightarrow 2$

$L_2 \rightarrow 4$

$L_3 \rightarrow 8$

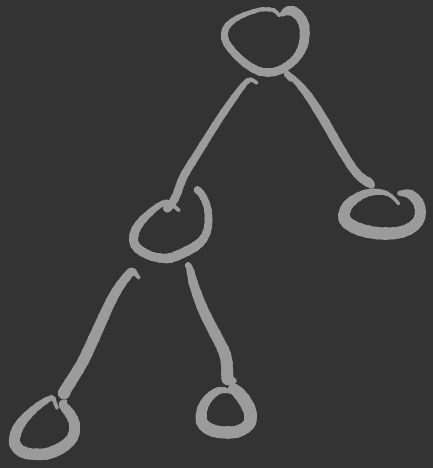
$L_4 \rightarrow 16$

⋮

$L_{10} \rightarrow 1024$

Almost Complete Binary Tree

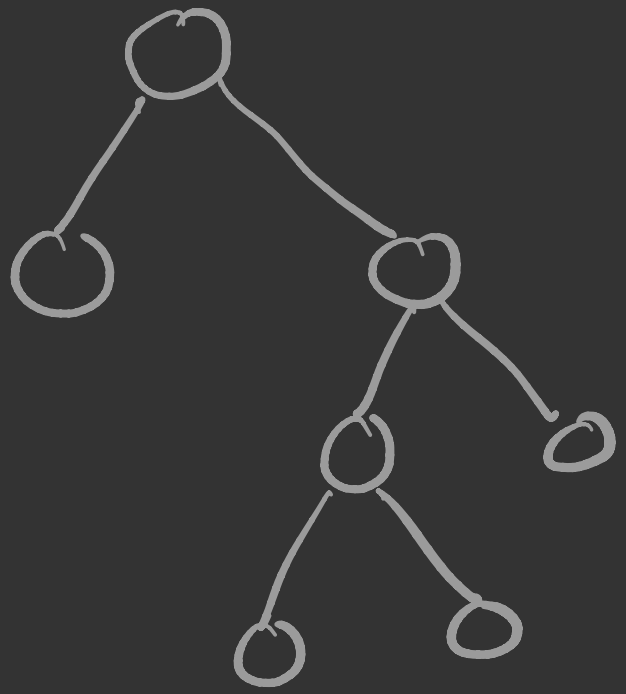
All levels are completely filled, except possibly the last level and nodes in the last level are all left aligned.



Strict Binary Tree

Each node of a strict Binary Tree will have either 0 or 2 children.

Full Binary Tree



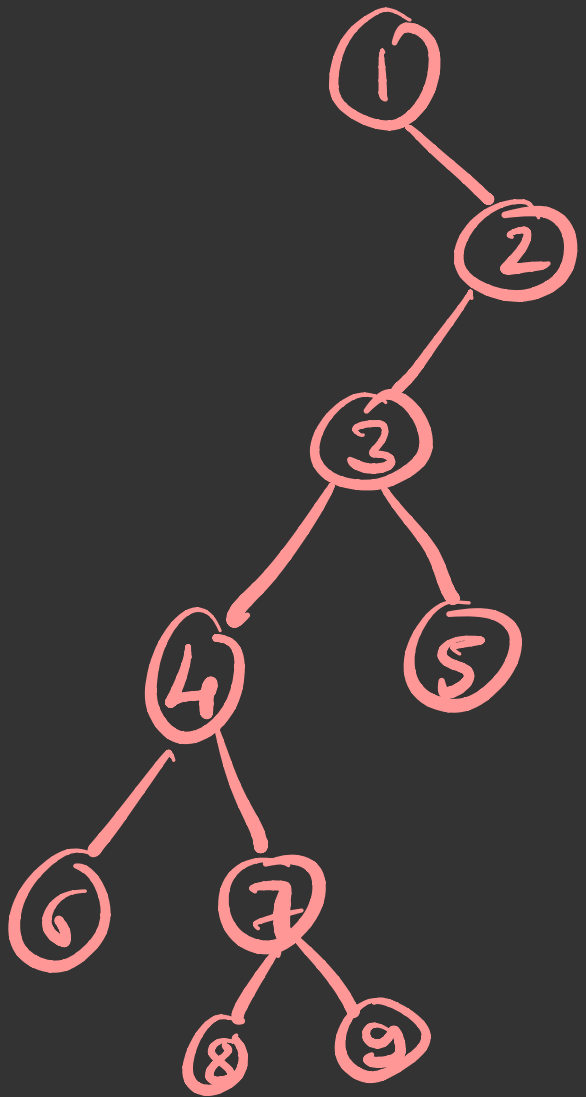
Representation of Binary Tree

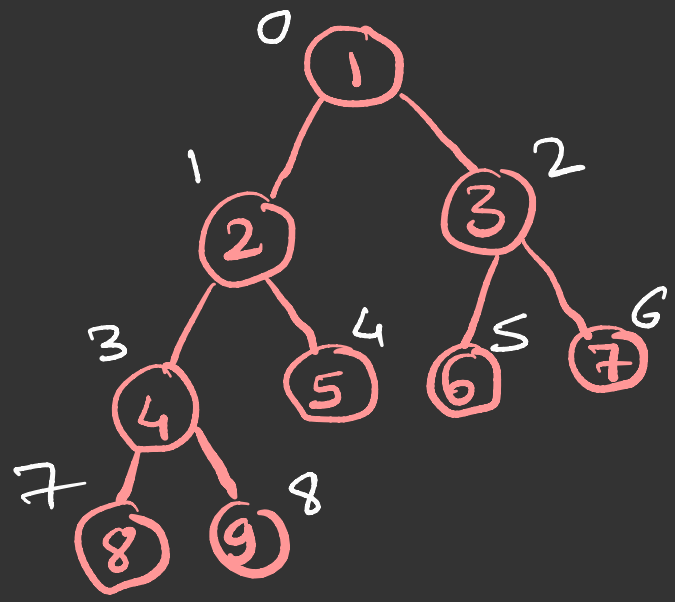
There are two possible representations of binary tree

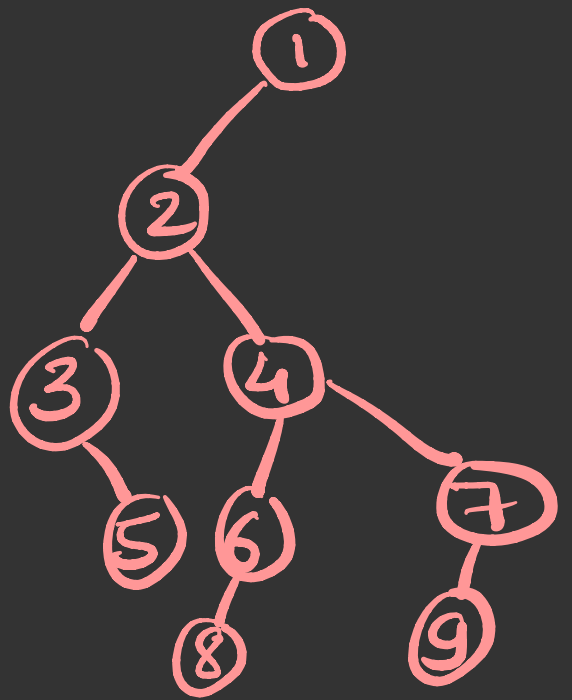
1. Array Representation
2. Linked Representation (by default)

Array Representation

0	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8	9



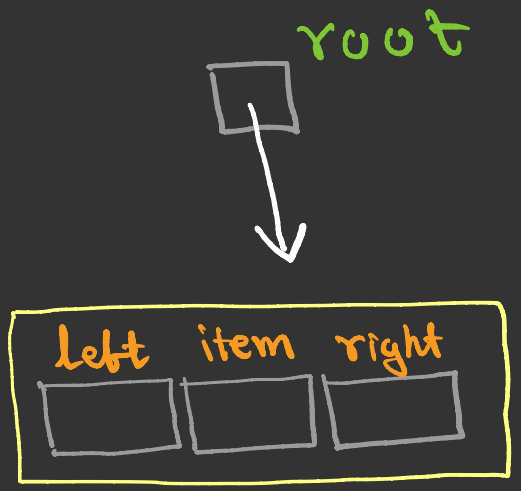




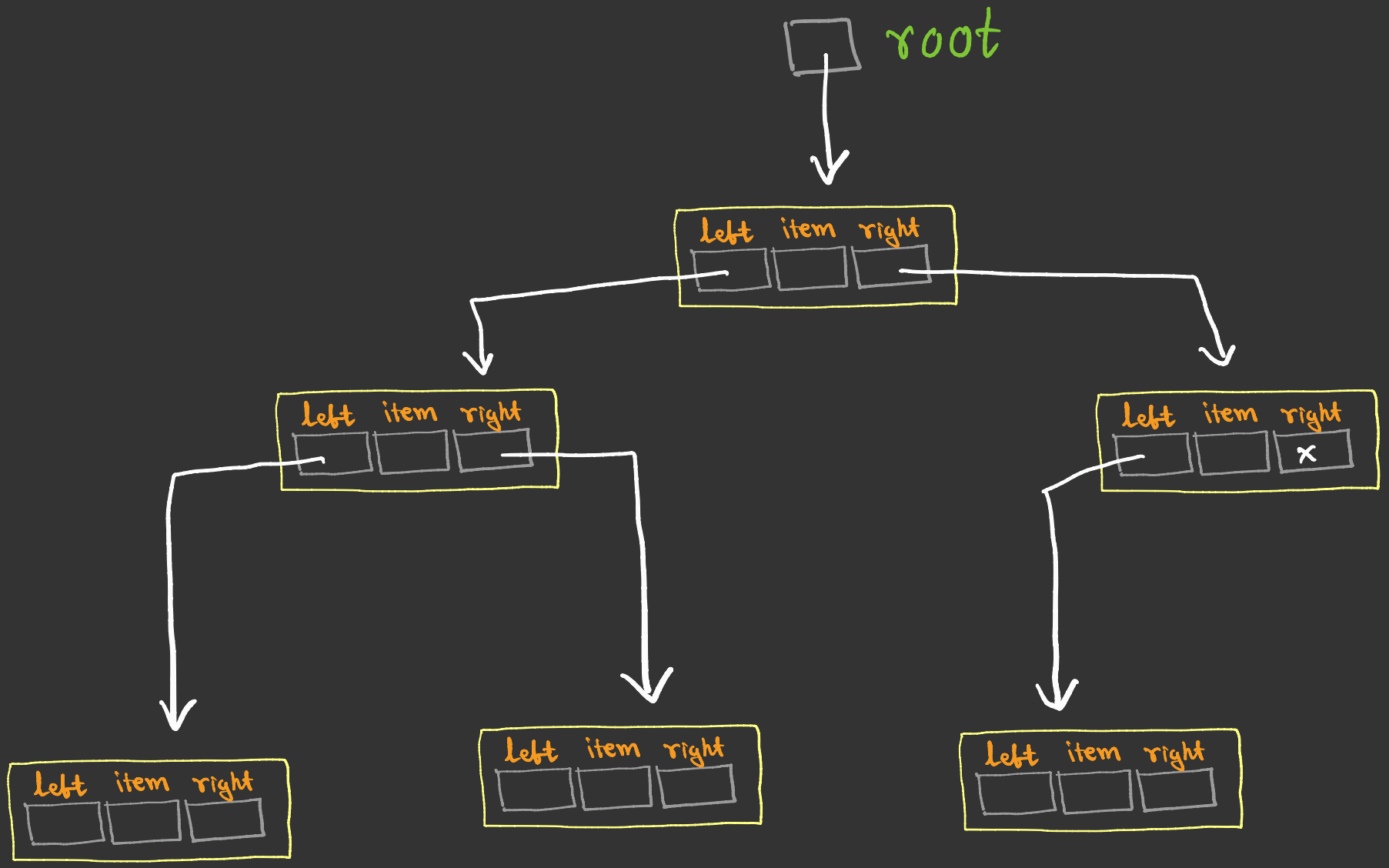
Linked Representation

node

Root

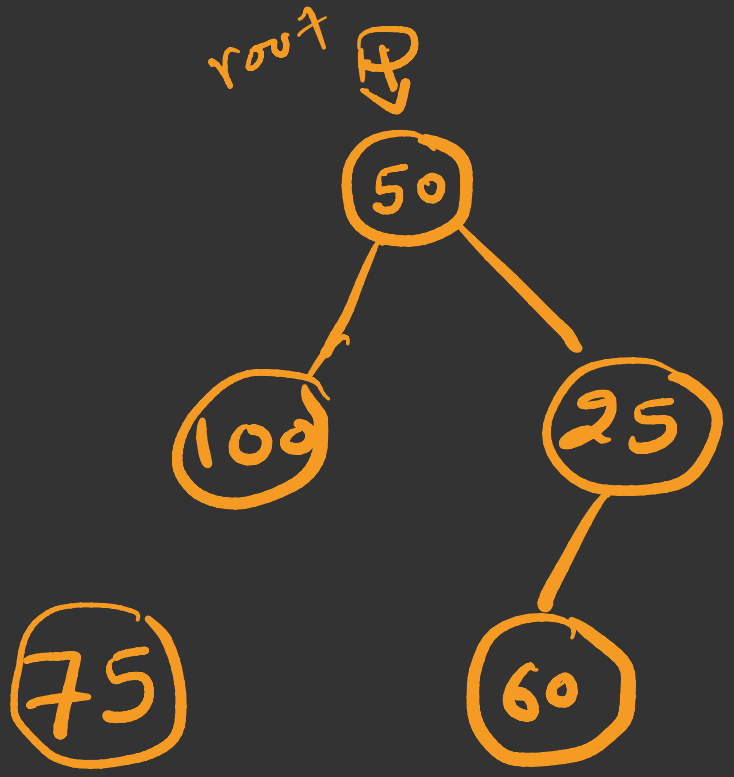


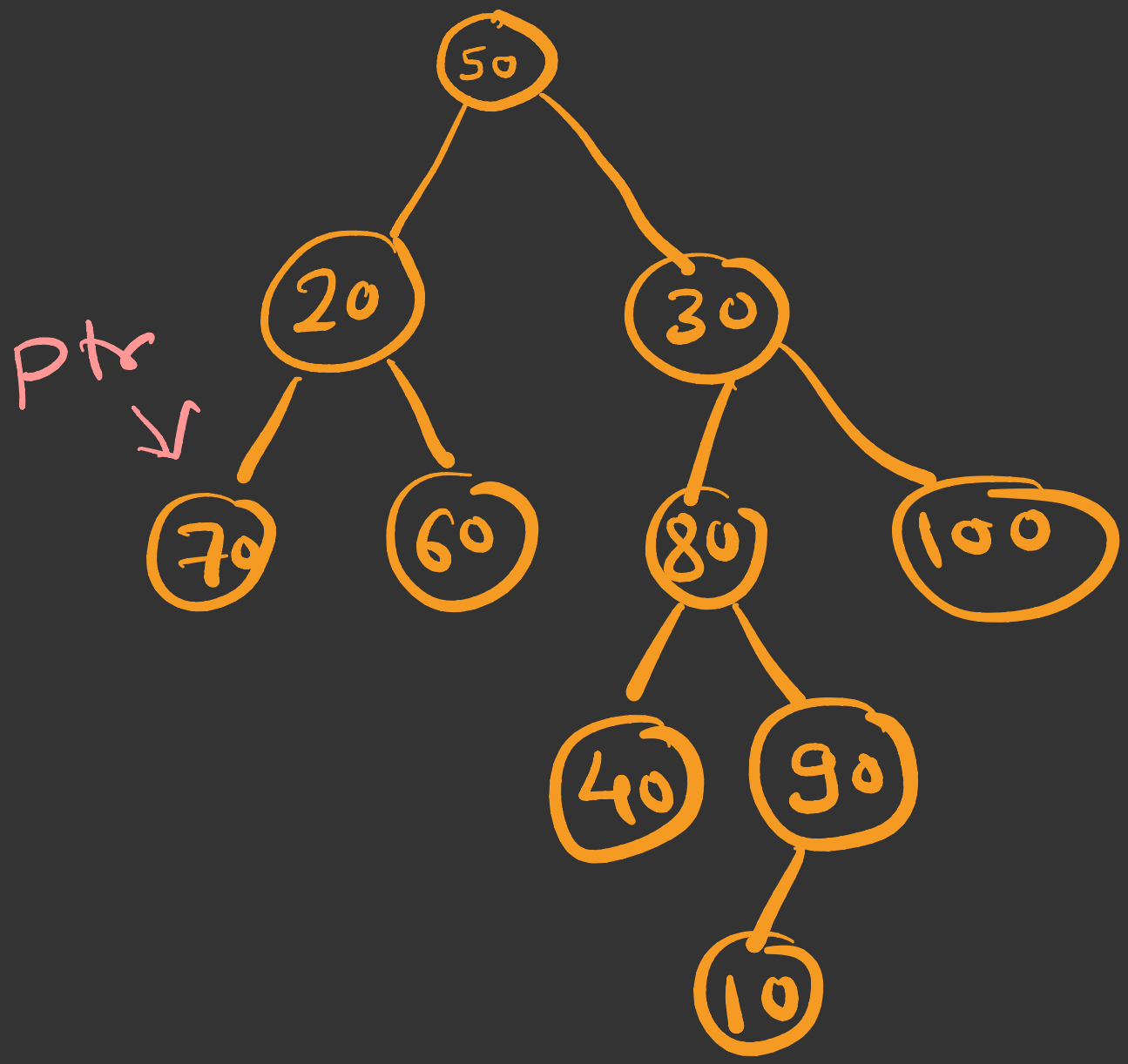
- root is a node pointer
- when root contains NULL, tree is empty.


root
left item right
left item right
left item right
x
left item right
left item right
left item right

Discuss

- How to insert an item in a BT?
- How to traverse a BT?






```
cout<<ptr->item; 50
ptr = ptr->left;
cout<<ptr->item; 20
ptr = ptr->left
cout<<ptr->item; 70
```