Yusrat Sadia Nailat

Binary Tree - Theory

Tree is a non-linear data structure. If each node of the tree has **at most two children,** then the tree is called a **Binary Tree.**

Tree terminologies:

- **Root:** the topmost node of the tree is called the root node
- A node may have
    - No child (**Leaf Node)**
    - One child
    - Two children

A node with **at least one child** is called the **internal node.**
A node that has **no child** is called is called the **Leaf Node/ External Node**

- **Degree:** the number of children to a node is called the degree of a node
- **Depth:** the length of the path from a Node A to the root node is called the Depth of the node. **The level of a Node is the Depth of that Node**
- **Height:** The longest path from the root node to one of the leaf nodes is called the Height of BT
- **Subtree:** A subtree is a tree that is the child of a node

Characteristics of a Tree

Two characteristics of a tree are the following:

1. A tree must be continuous and not disjoint. That means every node must be traversable starting from the root node.
2. A tree cannot have cycles. That means that the

   number of edges of a tree is = number of nodes - 1

Binary Tree

A Binary tree can have at most two children nodes. Given an array representation, the binary tree can drawn:

Usually in BT,

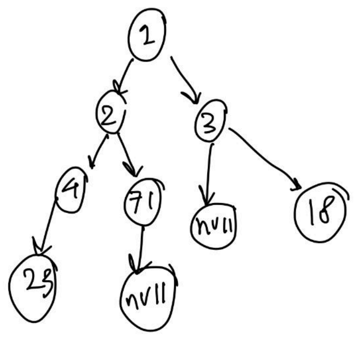
- Index i has children at indices 2*i+1 and 2*i + 2 (if 0th place is not NULL)

- Null means no node at that position

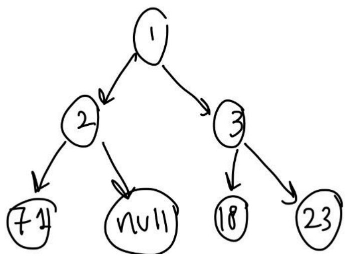
The elements are indexed level wise

Example: {null, 1,2 3,4, 71, null, 18, 23, null} convert this to BT

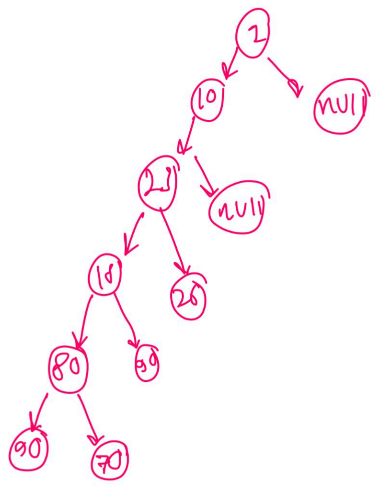
arr = {null, 1, 2, 3, 4, 71, null, 18, 23, null}



if arr = {null, 1, 2, 3, 71, null, 18, 23, null}



arr = {null, 1, 10, null, 25, null, 18, 26, 80, 90, 70, null}



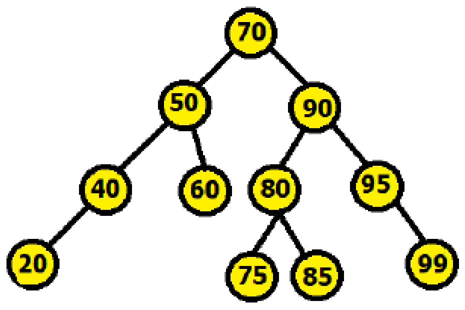
Let the number of internal nodes be N. So,

- The number of leaf nodes: N + 1
- The total number of edges: 2N
- The number of internal edges: N - 1
- The number of external edges: N + 1

Maximum number of nodes in a BT of Height H is = $2^H + 1$

Maximum number of nodes at level i **is given by** = 2^i

Binary Tree Traversal:



Pre-Order: 70, 50, 40, 20, 60, 90, 80, 75, 85, 95, 99

In-Order: 20, 40, 50, 60, 70, 75, 80, 85, 90, 95, 99

Post-Order: 20, 40, 60, 50, 75, 85, 80, 99, 95, 90, 70

Pre order traversal: Root -> Left -> Right (The node is printed when traversed for the first time)

70, 50, 40, 20, 60, 80, 80, 85, 95, 99

In order traversal: Left -> Root -> Right (The node is printed when traversed for the second time)

20, 40, 50, 60, 70, 75, 80, 85, 90, 95, 99

Post Order traversal: Left -> Right -> Root (The node is printed when traversed for the third time)

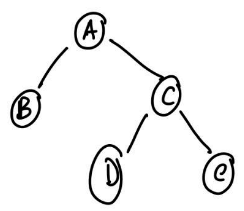
Yusrat Sadia Nailat

20, 40, 50, 60, 75, 80, 85, 99, 95, 90, 70

Types of Binary Tree

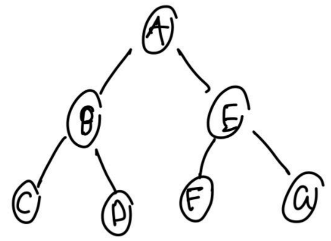
Full Binary Tree:

- All internal nodes (they are not leaf nodes) have **Exactly two children**



Internal Nodes are: A and C

Both Nodes have two children.

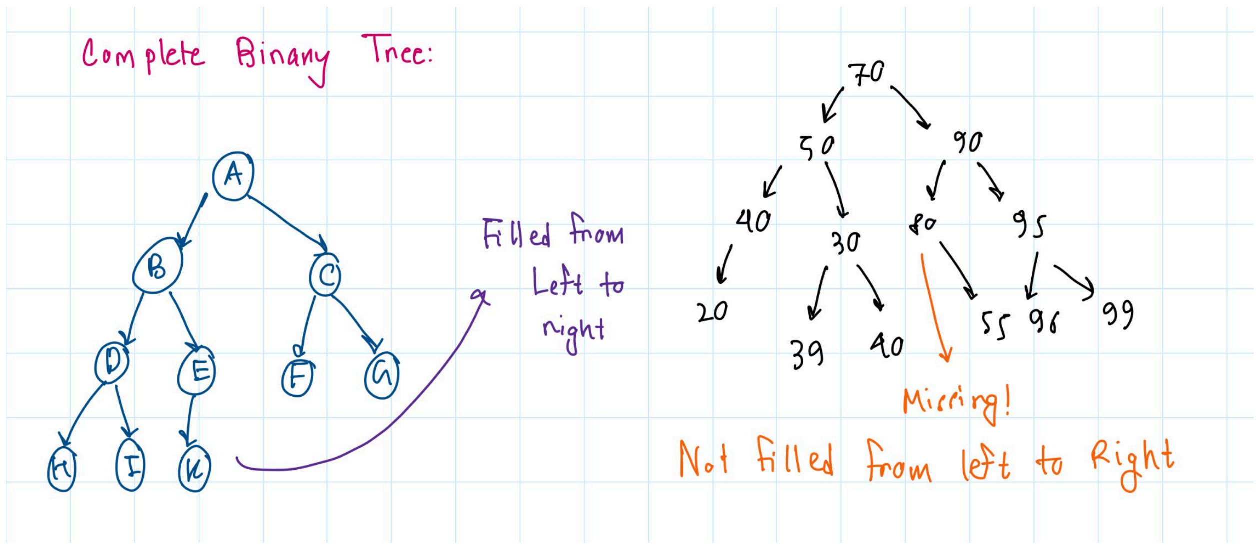


Similarly this binary tree is also full since every internal nodes have strictly two children nodes.

No of leaf Nodes = No of internal nodes + 1

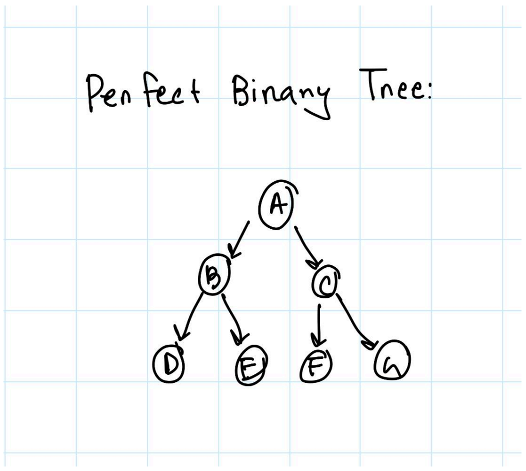
Complete Binary Tree:

- In a complete BT, all levels must be filled
- Except the last level, which should be filled from **left to right**



Perfect Binary Tree

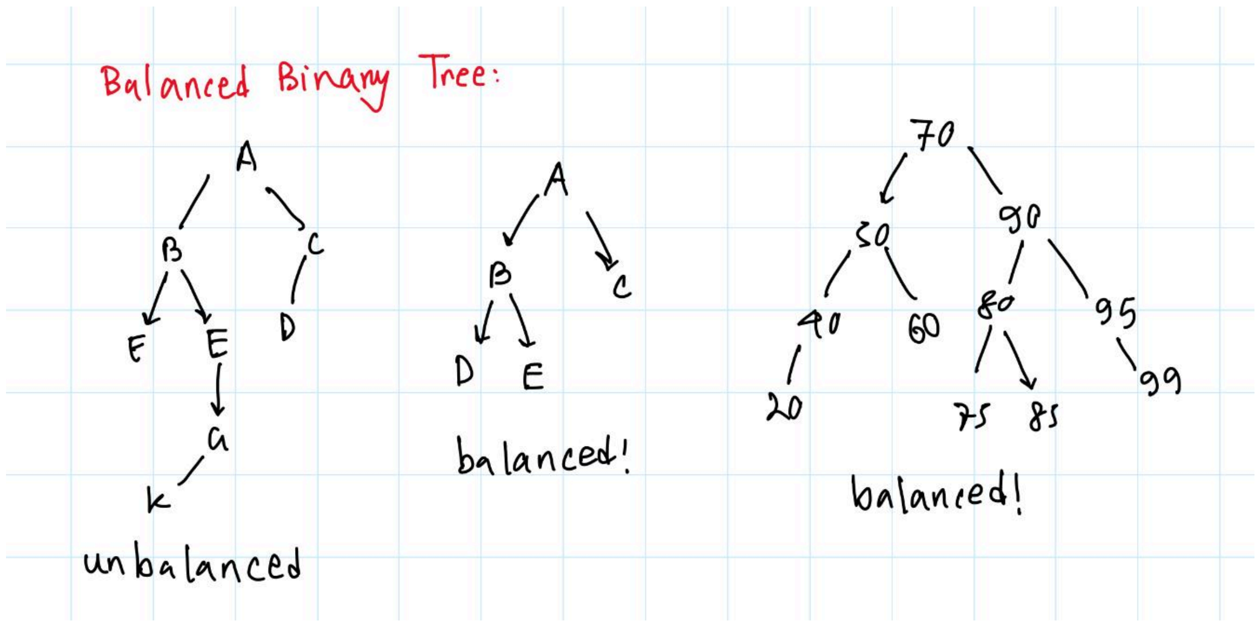
- **In a perfect binary tree, all internal nodes have exactly the two children nodes and the leaf nodes are at the same level**



Balanced Binary Tree:

In a balanced binary tree, the height of the left subtree and right subtree of **each node** is at **most one.**

Yusrat Sadia Nailat

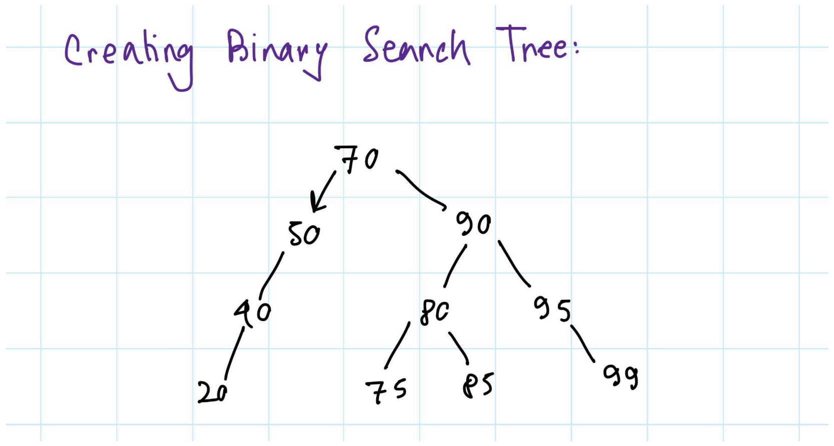


Binary Search Tree

Creating a BST, Insertion, and deletation:

Question: draw the BST by inserting the following number from left to right (when said "left to right", don't sort the arrangement! Directly use BST rule and draw the binary tree)

70, 50, 40, 90, 60, 20, 95, 99, 80, 85, 75

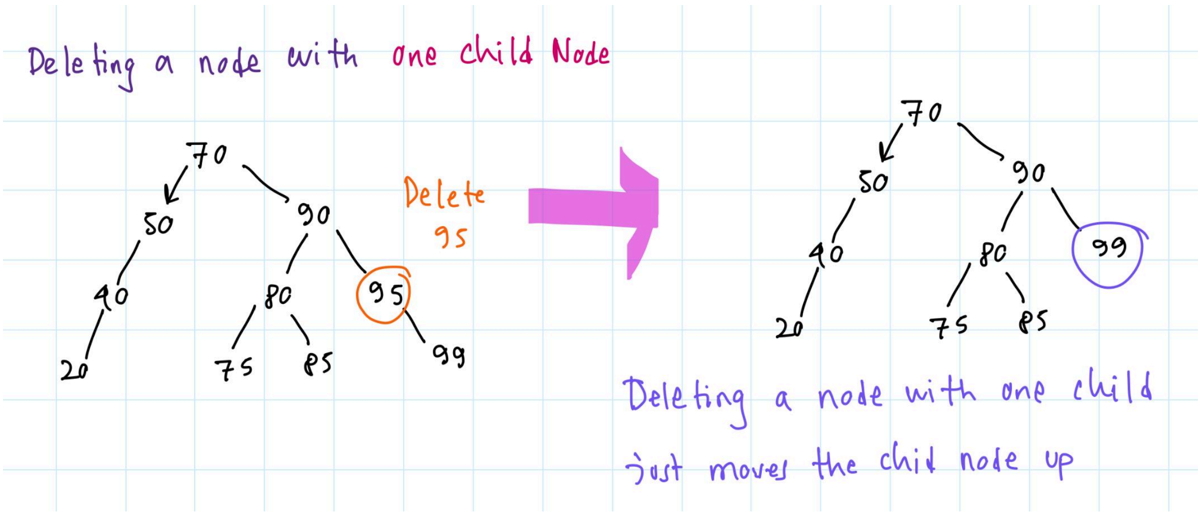


Insertion: same as creating a BST. just maintain the rules of BST and plug in the node in the right place

Yusrat Sadia Nailat

Delete: we can have three cases:

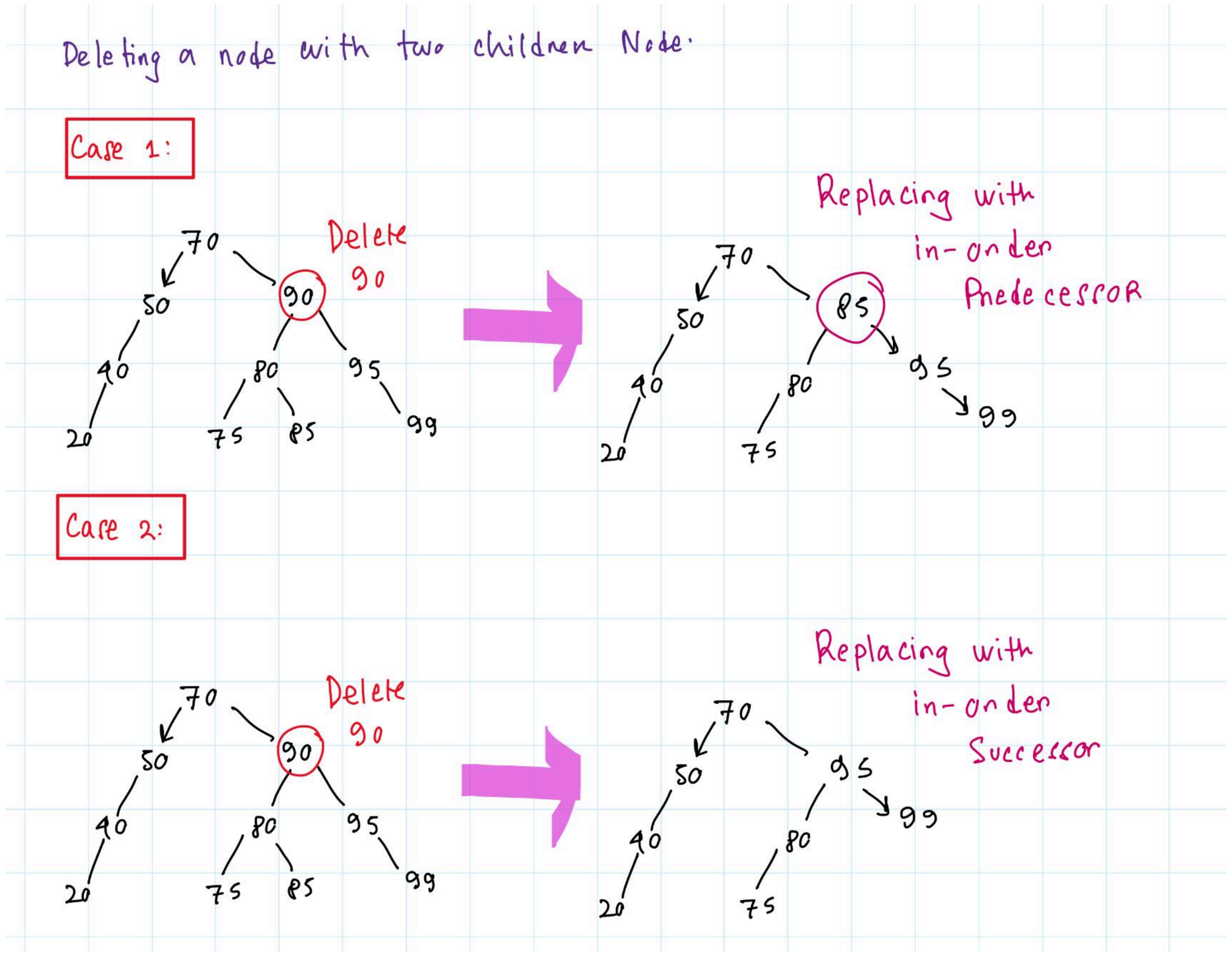
1. For deleting a node with **no child,** it is just removing the node
2. For deleting a node with **one child, we use move the child node to its place**


Deleting a node with one child Node
Delete 95
70
50
90
40
80
95
20
75
85
99
Deleting a node with one child just moves the chid node up

3. For deleting a node with **two children, we can have two cases:**

1. Find either:
   - The inorder successor → the smallest (leftmost) node in the right subtree OR
   - The inorder predecessor → the largest (rightmost) node in the left subtree
2. Replace the value of the node-to-delete with that value.

3. Delete the inorder successor (or predecessor) node, which will now fall into Case 1 (leaf) or Case 2 (one child)


Deleting a node with two children Node.
Case 1:
Delete 90
Replacing with in-order Predecessor
Case 2:
Delete 90
Replacing with in-order Successor
70
50
90
40
80
95
20
75
85
99

Example Type Question:

1. **The preorder traversal sequence of a binary search tree is 30, 20, 10, 15, 25, 23, 39, 35, 42. Which of the following is the postorder traversal sequence of the same tree?**

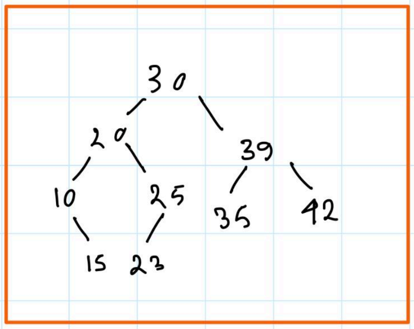
   A. 10, 20, 15, 23, 25, 35, 42, 39, 30
   B. 15, 10, 25, 23, 20, 42, 35, 39, 30
   C. 15, 20, 10, 23, 25, 42, 35, 39, 30
   D. 15, 10, 23, 25, 20, 35, 42, 39, 30

Yusrat Sadia Nailat

Pre-order Traversal: 30, 20, 10, 15, 25, 23, 39, 35, 42

root → left → right

Also in BST, the nodes of left subTree has smaller values than the root and the right subtree nodes have higher values of nodes.



Post-order: left → right → root

15, 10, 23, 25, 20, 35, 42, 39, 30

Root

2. **The postorder traversal of a BST is given as: 5, 12, 10, 20, 25, 22, 30, 28, 26, 24, 15. What is its preorder traversal?**

```
                15
              /   \
           10      24
          / \    /  \
        5   12  22    26
                /    / \
             20   25    28
                          \
                          30
```

So, post order traversal: **15, 10, 5, 12, 24, 22, 20, 26, 25, 28, 30**

3. Suppose that we have numbers between 1 and 100 in a binary search tree and want to search for the number 55. Which of the following sequences CANNOT be the sequence of the nodes examined?
   A. {10, 75, 64, 43, 60, 57, 55}
   B. {90, 12, 68, 34, 62, 45, 55}
   C. {9, 85, 47, 68, 43, 57, 55}
   D. {79, 14, 72, 56, 16, 53, 55}

Answer: So the sequence of nodes examined must be consistent:

Option A: {10, 75, 64, 43, 60, 57, 55}

- Start: (1,100)
- 10 < 55 → go right → range (11,100)
- 75 > 55 → go left → range (11,74)
- 64 > 55 → go left → range (11,63)
- 43 < 55 → go right → range (44,63)
- 60 > 55 → go left → range (44,59)
- 57 > 55 → go left → range (44,56)
- 55 inside (44,56) ✅ Works.

Option B: {90, 12, 68, 34, 62, 45, 55}

- Start: (1,100)
- 90 > 55 → go left → range (1,89)
- 12 < 55 → go right → range (13,89)
- 68 > 55 → go left → range (13,67)
- 34 < 55 → go right → range (35,67)
- 62 > 55 → go left → range (35,61)
- 45 < 55 → go right → range (46,61)

- 55 inside (46,61) ✅ Works.

Option C: {9, 85, 47, 68, 43, 57, 55}

- Start: (1,100)
- 9 < 55 → go right → range (10,100)
- 85 > 55 → go left → range (10,84)
- 47 < 55 → go right → range (48,84)
- 68 ??? But 68 > 55, must go left → new range (48,67)
- 43 ??? 43 < 55 but **outside (48,67)** ❌ Contradiction!

So this sequence is **impossible**.

Option D: {79, 14, 72, 56, 16, 53, 55}

- Start: (1,100)
- 79 > 55 → go left → range (1,78)
- 14 < 55 → go right → range (15,78)
- 72 > 55 → go left → range (15,71)
- 56 > 55 → go left → range (15,55)
- 16 < 55 → go right → range (17,55)
- 53 < 55 → go right → range (54,55)
- 55 inside (54,55) ✅ Works.

🎯 Final Answer

The sequence that **cannot** happen is: **C. {9, 85, 47, 68, 43, 57, 55}**

The sorted array: {30,40,41,42,45,49,50,55,100}

```
          50
        /     \
      40        100
     / \          /
   30   42     55
        / \
      41   49
           /
         45  ← inserted here
```

After Binary Searching to create a new balanced binary tree.

```
        45
       /   \
     41     55
    / \   / \
  40   42 50  100
  /       /
30       49
```

ix) Suppose, you are given the array representation of a binary tree:
[None, 6, 8, 4, 3, None, 11, 20, None, 2, None, None, 7, 5] Which of the following statements is NOT correct?

a) Node 20 is the parent of Node 7.
b) Node 11 is the parent of Node 5.
c) This binary tree is not balanced.
d) This binary tree is not full/strict.

```
        6
      /    \
    8        4
   /        /  \
  3        11    20
   \             / \
    2.          7.   5
```

So, (b) is the wrong answer. **Notice that two None None indicates null places on the right side of the left subtree.**