is a balanced search tree where

* 2-3 Tree → It can be a leaf node ○

→ 2 Node: (a) with children X, Y — 2 children, 1 data/root element

Properties: Node possible.

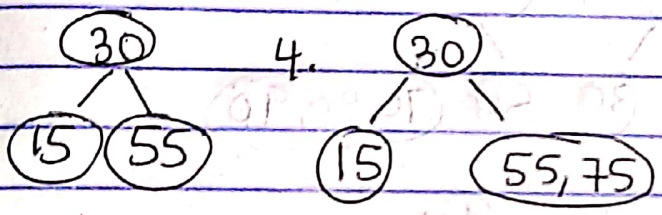
→ 3 Node: (a,b) with children X, Y, Z — 3 children, 2 data/root element

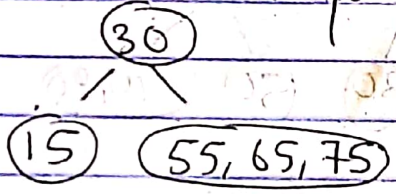
→ Data is always stored in sorted manner.
→ Balanced Tree.
→ All leaf-node are at same level
→ Each node can be leaf, 2node, 3node.
→ Insertion is done on leaf node.

Insertion/Creation: 55, 30, 15, 75, 65, 45, 5

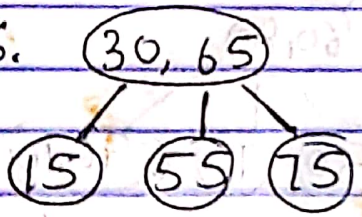
1. (55) 2. (30,55) 3. (15,30,55)

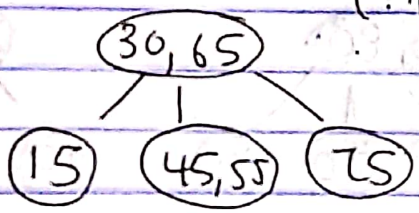
↳ split (push middle element) to parent

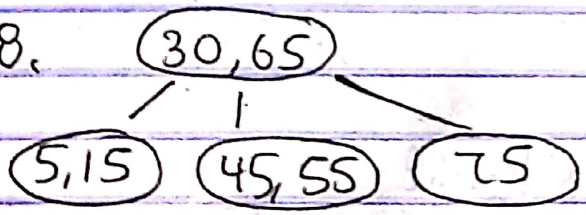
4. 

5. 

Split (push middle element) (parent)

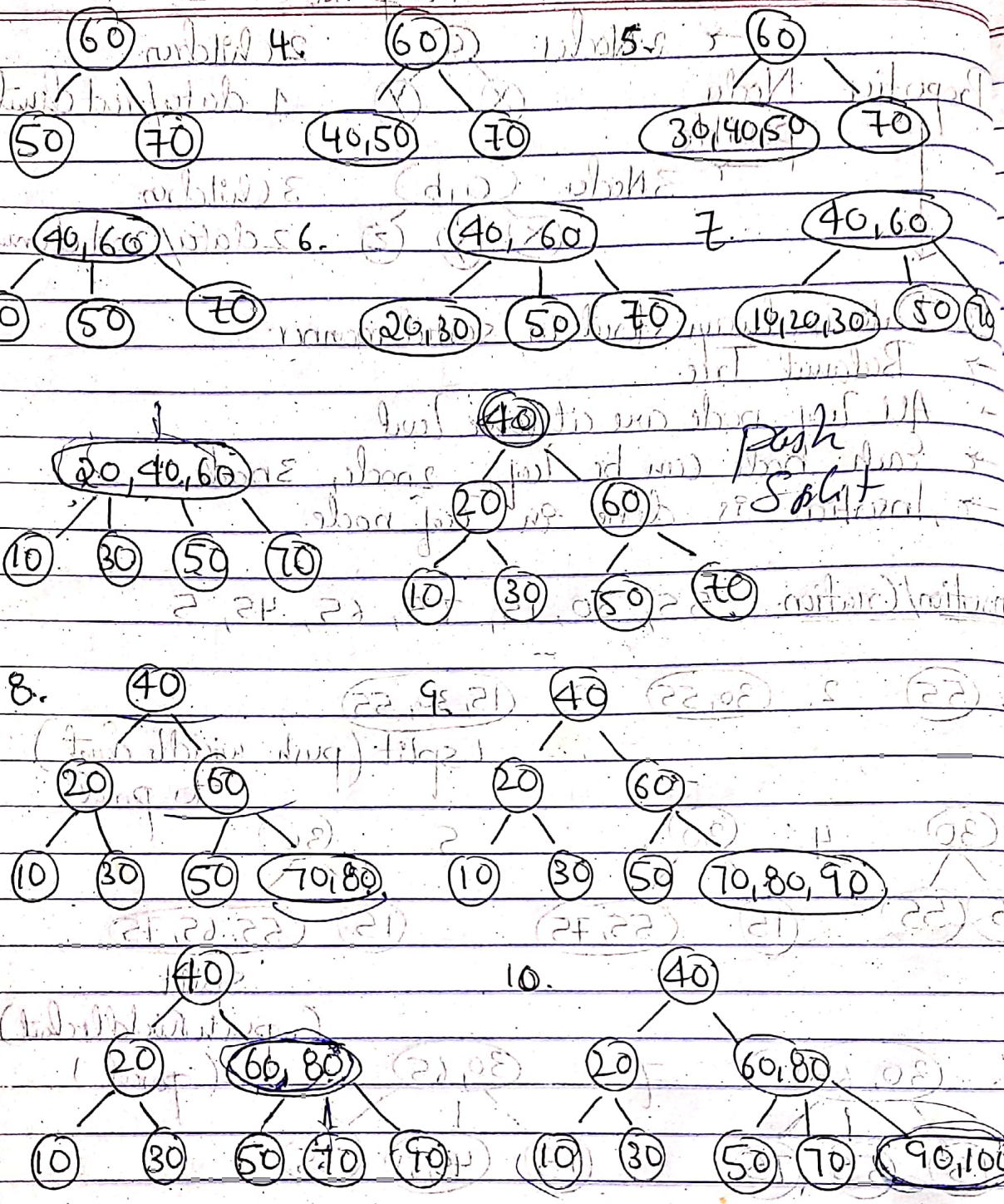
6. 

7. 

8. 

Eg 2: 50, 60, 70, 40, 30, 20, 10, 80, 90, 100

→ 1. 50 2. 50,60 3. 50,60,70

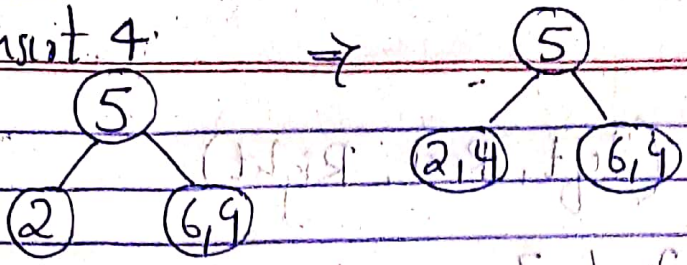


→ Why 2-3 Tree.

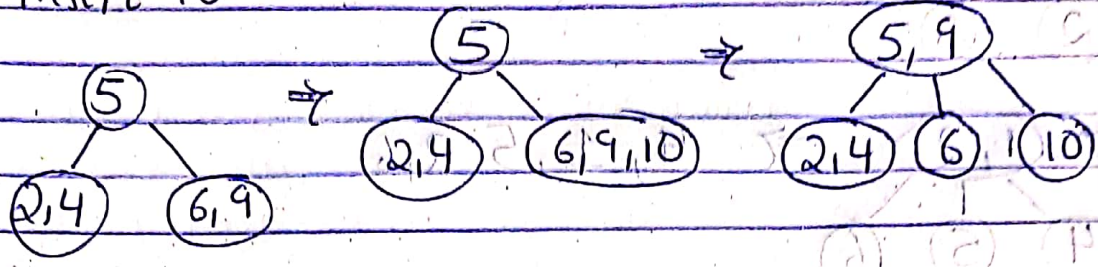
1. Easier to balance the height
2. Easier Access than Binary Search Tree.
3. Used in file system & database.

Insertion

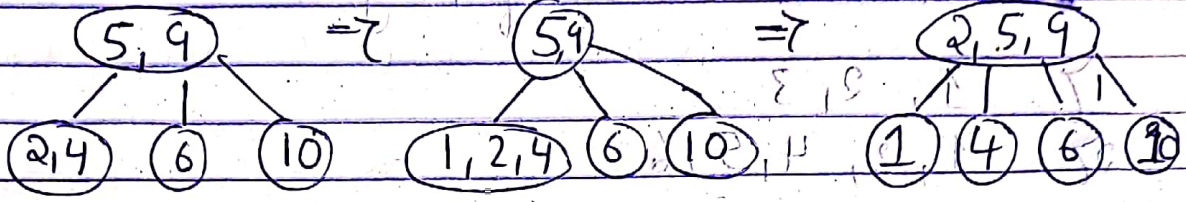
1) Insert 4.

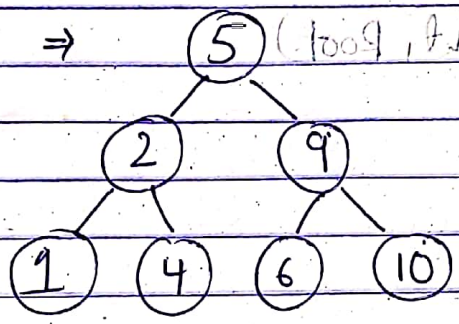


2) Insert 10.



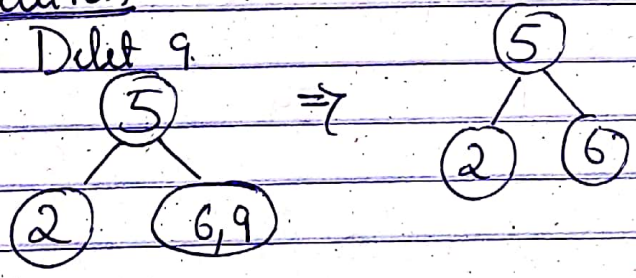
3) Insert 1.



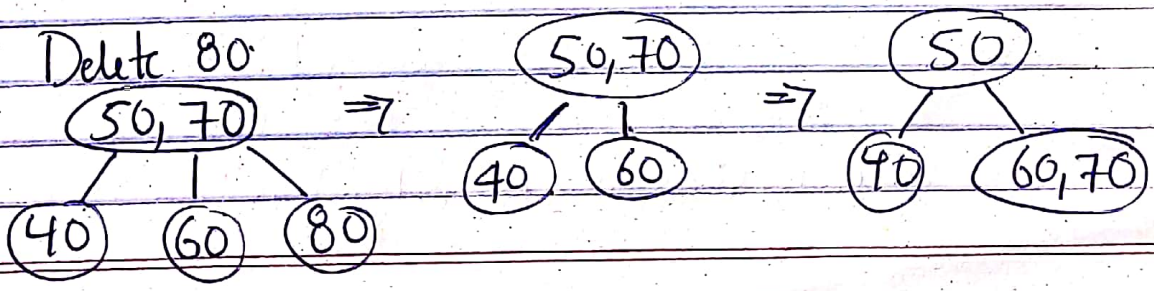


Delution

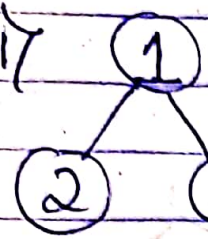
1) Delet 9.

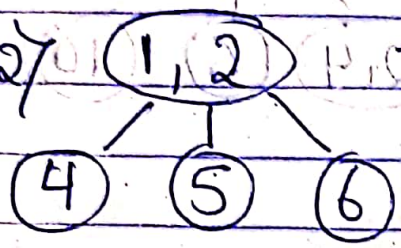


2) Delete 80



Inorder Traversal (Left, Root, Right)

1)  ⇒ 2, 1, 3

2)  ⇒ 4, 1, 5, 2, 6

Pre order Traversal (Root, Left, Right)

1) 1, 2, 3

2) 1, 2, 4, 5, 6

Post order Traversal (Left, Right, Root)

1) 2, 3, 1

2) 4, 5, 6, 1, 2.