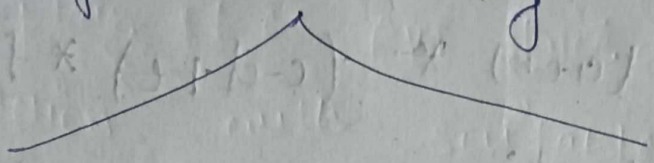
Tutorial-7

Q1:- what are the types of threaded binary tree?

Types of Threaded Binary tree:



single-threaded	Double Threaded
⇒ only one child (either left or right) is threaded.	⇒ Both left and right children are threaded

Q2. what is AVL Tree?

⇒ A self-balancing binary search tree where the difference between the heights of left and right subtree (balance factor) is at most 1.

Q.3. List out the steps involved in deleting a node from a binary search tree

⇒ Step to Delete a Node from a BST:

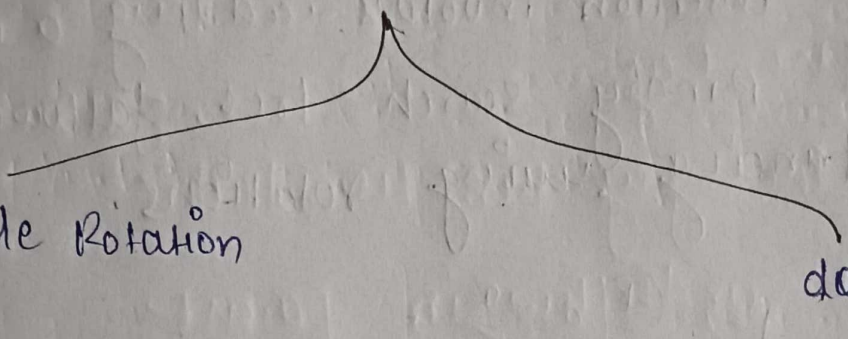
1) Find the node to delete

2) if the node has no children, remove it.

3) if the node has one child, replace it with the child

4) if the node has two children, replace it with its in-order successor or predecessor and adjust the tree.

Q4 Describe the algorithms used to perform single and double rotation on AVL Tree



- Single Rotation
  - ⇒ For LL or RR imbalance, perform a single rotation (Right or left, respectively)
- double Rotation
  - ⇒ For LR or RL imbalance,
  - ⇒ Perform two Rotation
  - e.g left-Right or Right-left

Q5 What are the threaded binary tree? Write an algorithm for inserting a node in a threaded Binary tree

→ A binary tree where NULL pointer in leaves are replaced with thread pointing to the in-order predecessor or successor

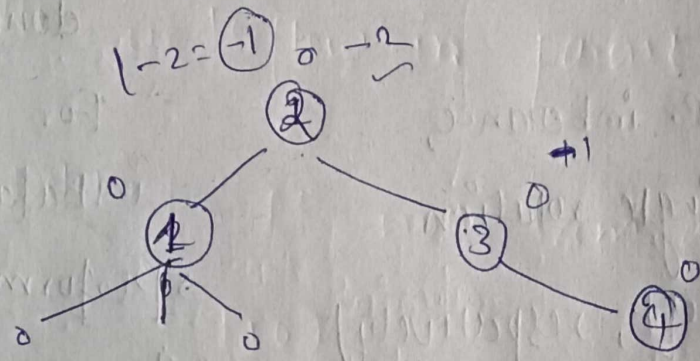
Algorithms:-

⇒ Depends on traversing the tree to find the correct positions and updating threads

Q-6 Explain the AVL tree insertion and deletion with suitable example

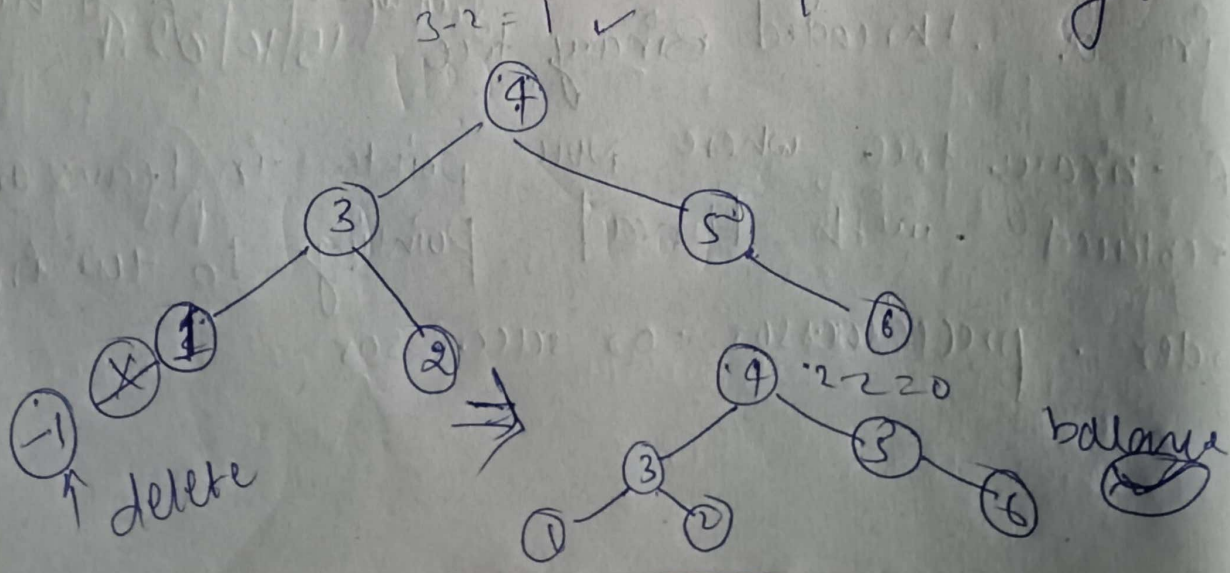
Insertion:- insertion involves adding a node like in Binary search tree, followed by rebalancing using rotations

eg



Balance Tree

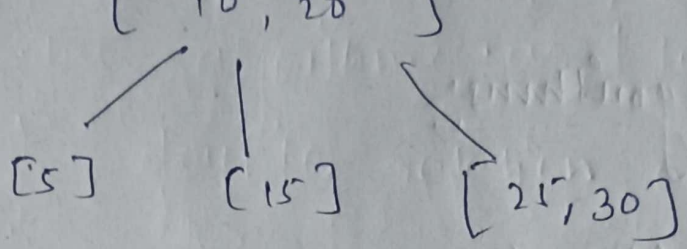
deletion :- Deletion involves removing the node like in BST, followed by rebalancing.



Q.7 Explain about B-Tree with suitable eg

Sol A balanced tree designed for disk storage with multiple key in a node Nodes split when full.

for eg



⇒ all leaf nodes are at the same level
⇒ each node has keys in sorted order
⇒ the tree is balanced, ensuring efficient search insertion, and deletions

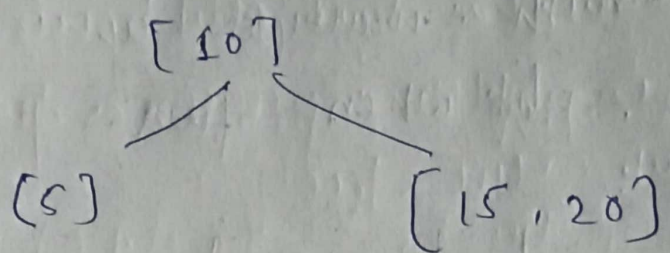
Q.8 Explain about B+ trees with suitable algorithms

⇒ A variant of B-Tree where all values are stored in leaf nodes connected in a linked list, enabling faster range Queries

insertion - Algorithms:-

i) start the root
ii) insert into the leaf
iii) Handle parent split reursively

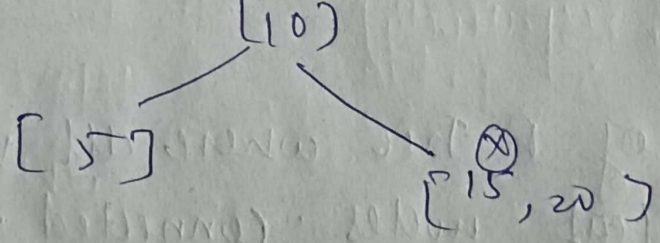
Example:- 10, 20, 5, 15 into a B+ tree of order ③



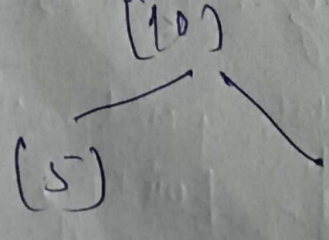
Deletion Algorithms:-

1) Locate the key
2) delete the key from the leaf
3) Handle underflow in parent

for eg Deleting 15 node from above tree



⇒ Remove 15 (Redistribute/Re from sibling)



4, 12:04

(3) search algo:-

1) start the root

2) · Traverse the leaf

3) search within the leaf

Q.9 · why does time complexity of search operation in B-Tree is better than binary search tree (BST)?

Ans · B-Tree vs BST search Time complexity

→ B-Tree are optimised for disk access reducing height and Input/output operations · unlike BSTs · which can · become unbalanced · and · deep ·