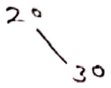
AVL Tree : inserting {20, 30, 8, 47, 39, 18, 26, 79}

insert 20; The tree is empty

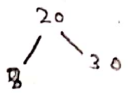
Tree is balanced

insert 30 : insert as right child



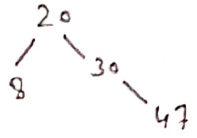
Tree is balanced

insert 8 : insert as left child



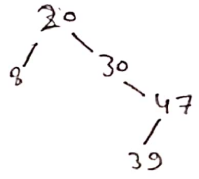
Tree is balanced

insert 47 : insert as right child of 30



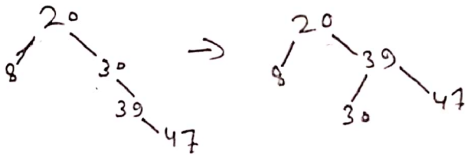
Tree is balanced

insert 39 : insert as left child of 47



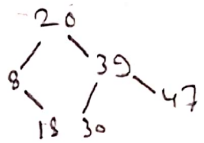
Tree is unbalanced (The imbalance is in the node of 30)

Balance the tree by performing a right rotate on 47 then a left rotate on 30



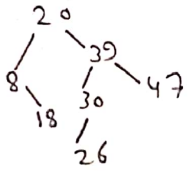
Tree is balanced

insert 18 as a right child of 8



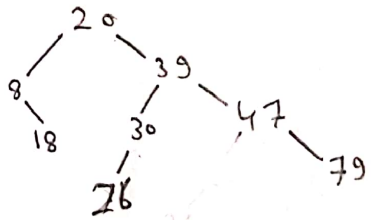
Tree is balanced

insert 26 as left child of 30



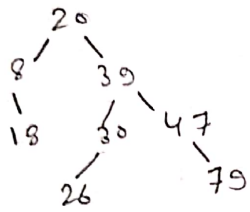
Tree is balanced

insert 79 as right child of 47

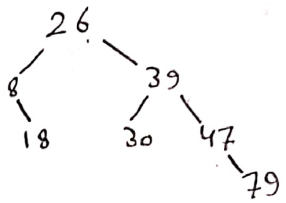


Tree is balanced

AVL Tree : removing element in FIFO format

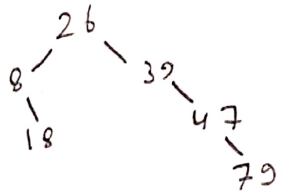


remove 20 →

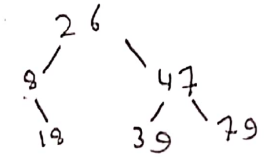


Tree is balanced

remove 30 →

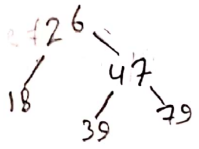


The tree is imbalanced at 39 (right right case) → perform a left rotate on 39



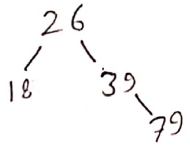
Tree is balanced

remove 8 →



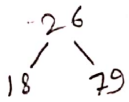
Tree is balanced

remove 47 →



Tree is balanced

remove 39 →



Tree is balanced

remove 18 →

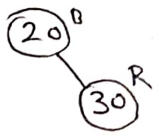
Tree is balanced

remove 26 → 79

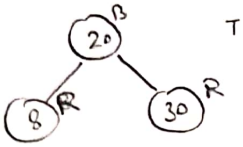
remove 79 → empty

Red-Black Tree: inserting: {20, 30, 8, 47, 39, 18, 26, 79}

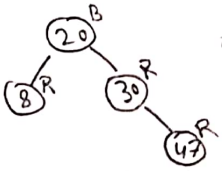
Tree is empty → inserting 20 as Black → 

→ inserting 30 as red → 

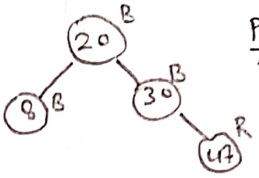
Tree is Balanced (no adjacent red nodes and no. of black nodes in each Path is the same)

inserting 8 as red → 

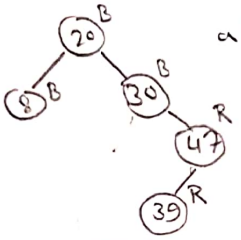
Tree is balanced

inserting 47 as red → 

adjacent red nodes property violated

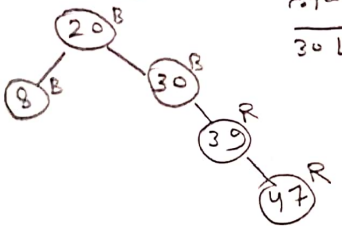
Parent of 47 is red → Parent's sibling (8) is red → recolor 30 and 8 → 

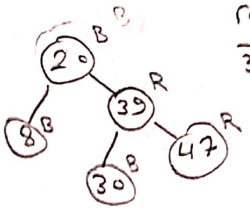
Parent's Parent is root → balancing done

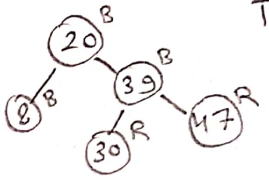
insert 39 as red → 

adjacent red nodes 47 and 39

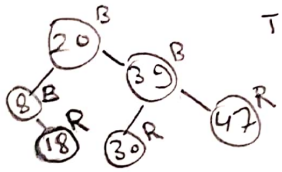
Paren sibling is NIL → grad parent is not root → rotate in Right Left case

rotate 47 to the right → 

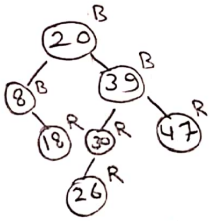
rotate 30 Left → 

recolor 30 and 39 → 

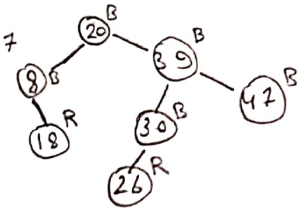
Tree is balanced

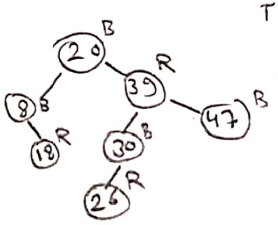
insert 18 as red → 

Tree is Balanced

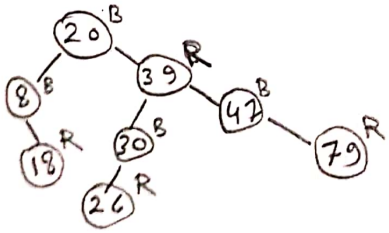
insert 26 as red → 

adjacent red nodes 30 and 26

Parent's sibling is Red → recolor 30 and 47 → 

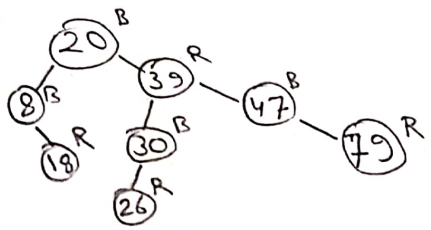
Parent's Parent (39) is not root → recolor 39 → 

Tree is balanced

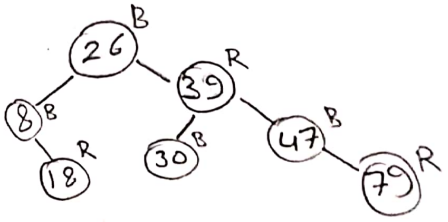
insert 79 as red → 

Done !!

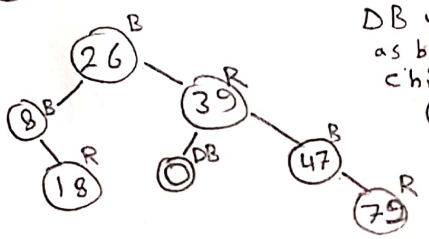
Red-Black Tree : Removing elements in FIFO format



Remove 20 → Replace 20 w 26

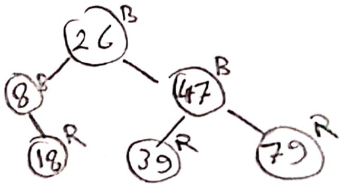


remove 30 →

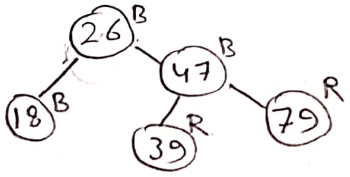


DB with sibling as black and its child as red (RR case)

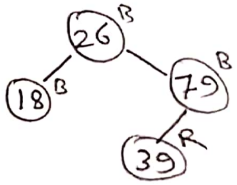
rotate 8 and parent to the left →



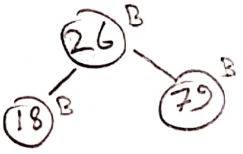
remove 8 → Replace 8 and 18 and remove leaf node



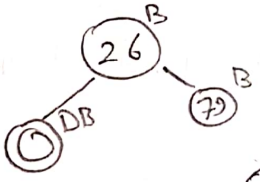
remove 47 → replace 47 and 79



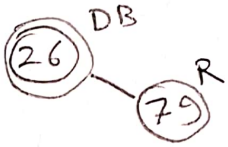
remove 39 → 39 is red and leaf remove directly



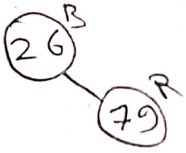
remove 18 → When removing 18 we get a DB



→ Sibling of DB is black so we make the parent a DB and change the color of sibling to red



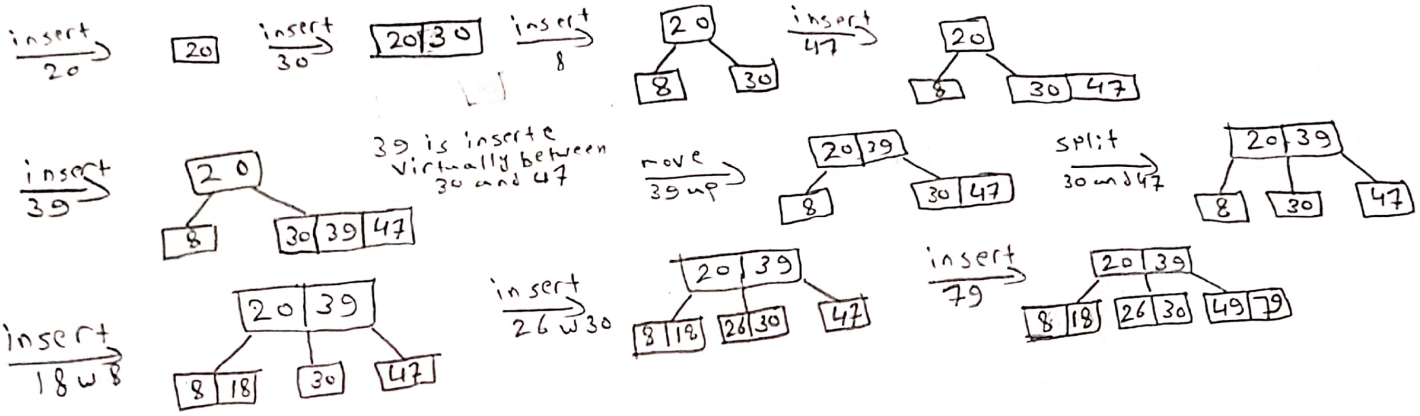
→ DB is root ⇒ remove DB and make it Black



remove 26 → replace 26 w 79 and remove the leaf node

remove 79 → empty Tree

2-3 Tree : insert elements: {20, 30, 8, 47, 39, 18, 26, 79}



2-3 Tree : remove elements in FIFO format