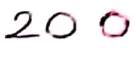
Homework 7

Q1

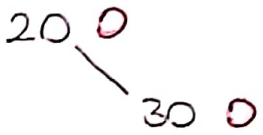
"20, 30, 8, 47, 39, 18, 40, 23"

AVL Tree:

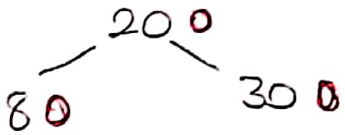
Insert 20



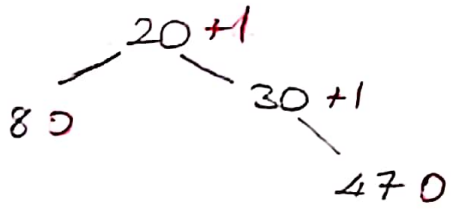
Insert 30



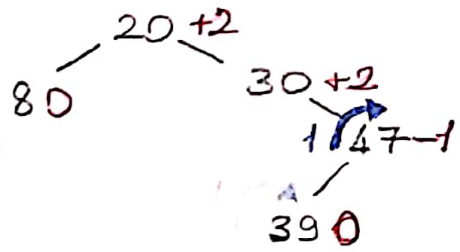
Insert 8



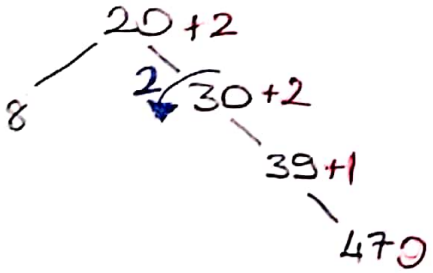
Insert 47

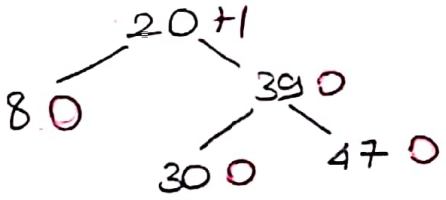


Insert 39

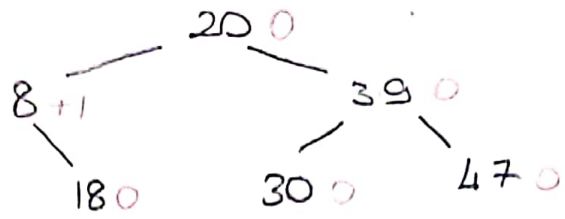


* Right Left
(parent balance +2, child balance -1)
1. Rotate right around child
2. Rotate left around parent

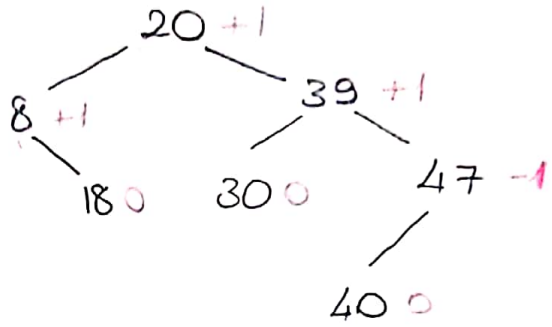




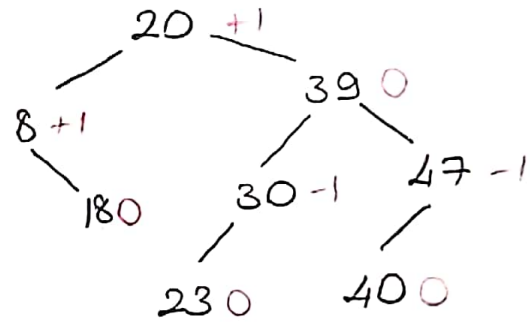
Insert 18



Insert 40

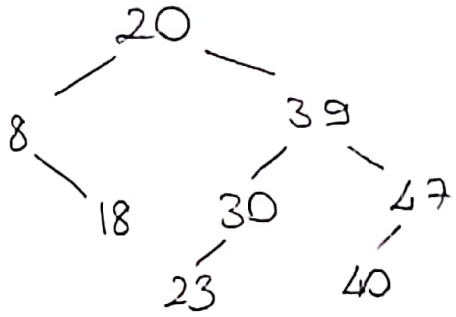


Insert 23



Remove

AVL Tree:



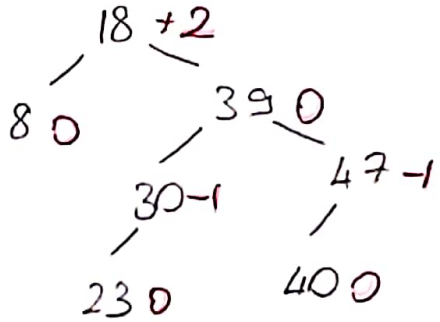
Delete 20

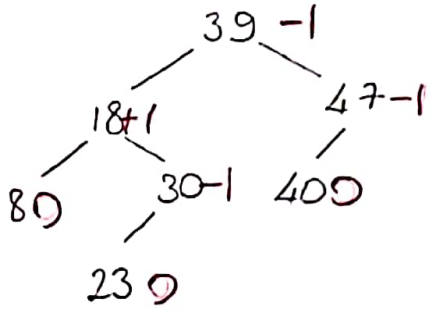
Perform standard BST delete.

Check the balance

18 is critical node.

Rotate left

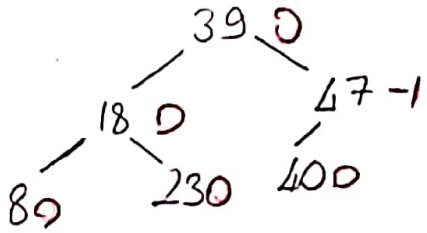




Delete 30

Perform standard BST delete

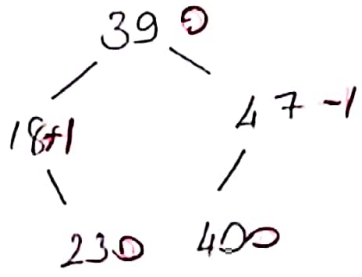
check the balance



Delete 8

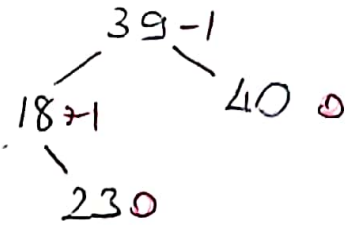
Perform standard BST delete

Check the balance



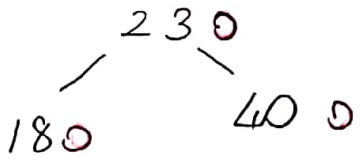
Delete 47

Perform standard
BST delete
Check the balance.



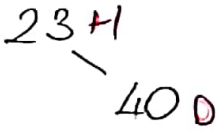
Delete 39

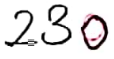
Perform standard
BST delete
Check the balance



Delete 18

Perform standard
BST delete
Check the balance.





Delete 40

Perform standard
BST delete
Check the balance

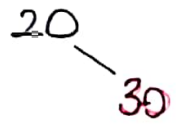
Delete 23

Perform standard
BST delete

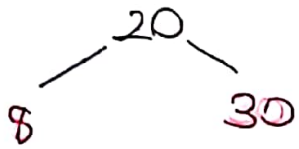
Red-Black Tree

Insert 20

Insert 30



Insert 8

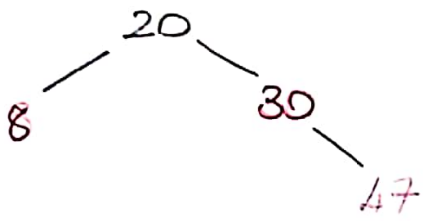


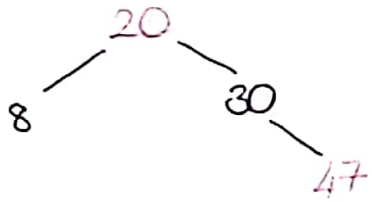
Insert 47

*case 1

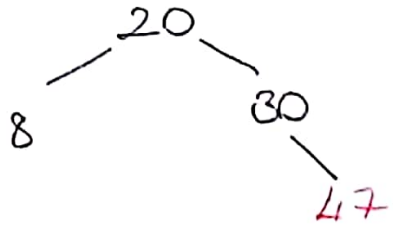
47's parent and its parent's sibling are red.

- Change colors.





- we can change root's color to black.

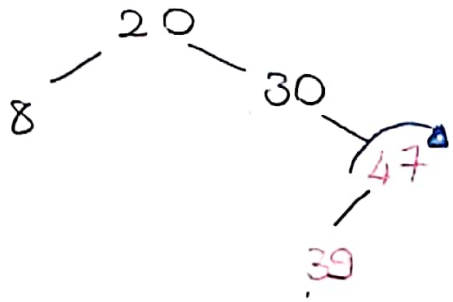


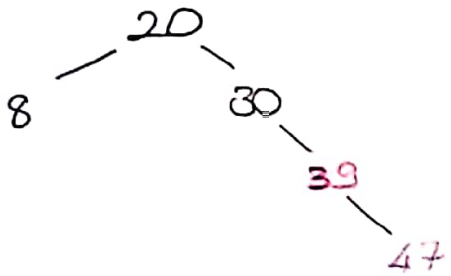
Insert 39

*case 3

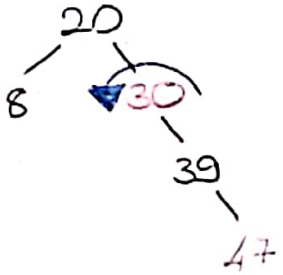
39's parent is red and its parents has not a sibling.

- Rotate so that the child is on the same side of its parent as its parent is to the grandparent

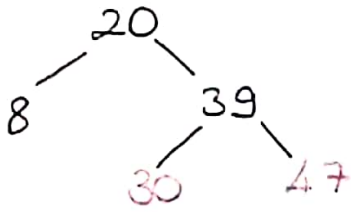




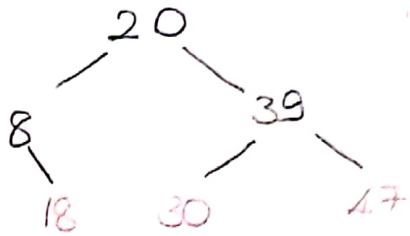
- change colors



- rotate left



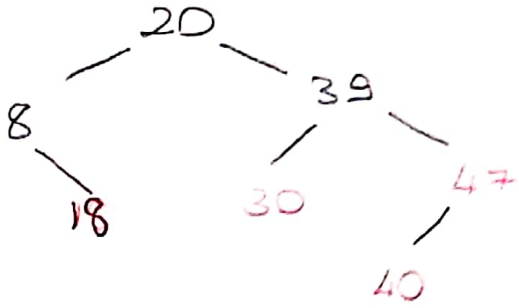
Insert 18

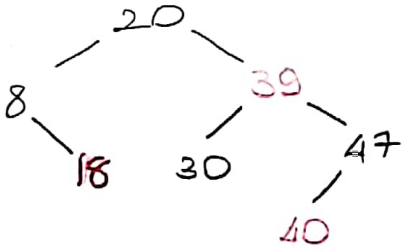


Insert 40

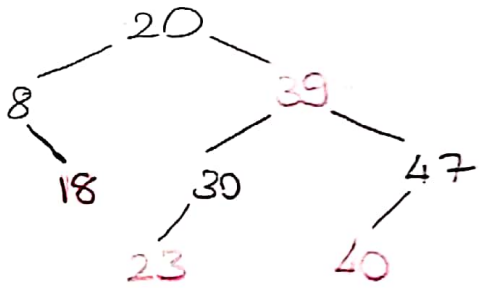
*case 1

40's parent is red, and its parent's sibling is red. Change colors.

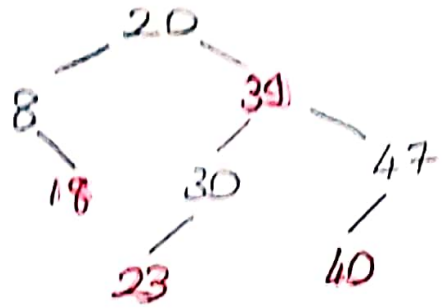




Insert 23



Red-Black Tree:

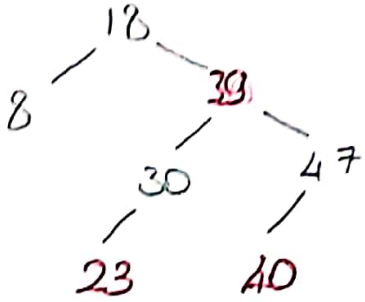


Delete 20

Perform BST deletion

We replace 20 with 18 and remove the node containing 18.

So, the deleted node is red.

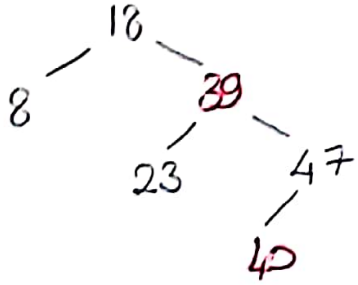


Delete 30

Perform BST deletion

We replace 30 with 23 and remove the node containing 23.

So, the deleted node is red.



Delete 8

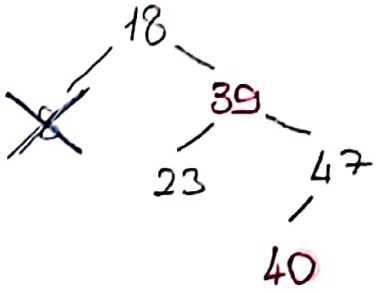
Perform BST deletion.

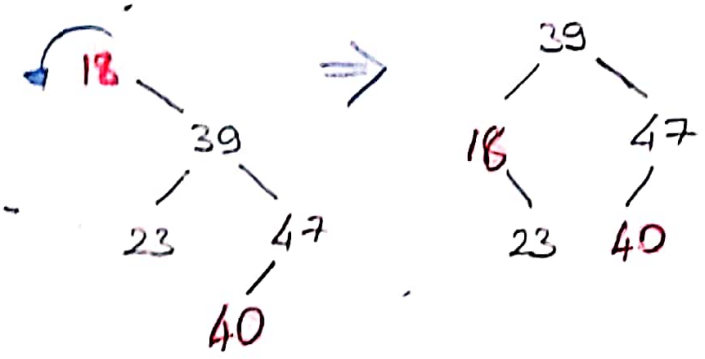
The deleted node is black.

Its sibling is red.

We swap colors of its parent and its sibling.

Rotate parent in deleted node direction.

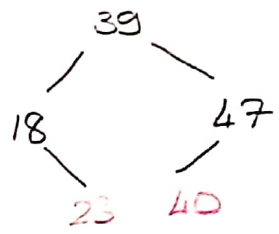




- Deleted node's sibling is black, and both its children are black. (null = black)

If parent is red, we change its color to black.

Make sibling is red.



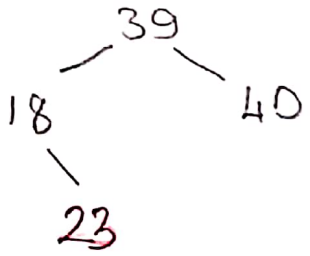
Delete 47

Perform BST deletion

We replace 47 with 40

and remove the node containing 40.

The deleted node is red.



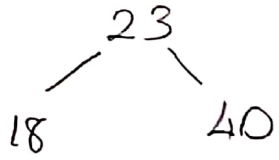
Delete 39

Perform BST deletion

We replace 39 with 23

and remove the node containing 23.

The deleted node is red.



Delete 18

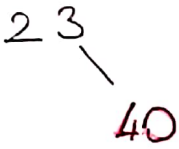
Perform BST deletion

The deleted node is black

Its sibling is black and both its children are black (null = black)

Parent is not black, there is no change.

Make sibling red.



Delete 40

Perform BST deletion

The deleted node is red,

we simply delete it.

Delete 23

2-3 Tree :

Insert 20

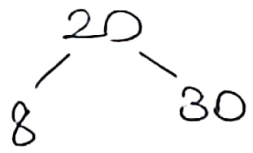
20

Insert 30

20, 30

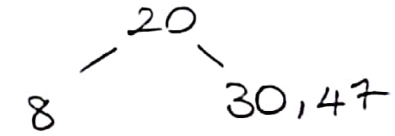
20 ← this node is a 2-node.
We insert directly into the node creating a 3-node.

Insert 8



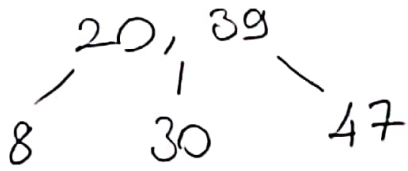
8, 20, 3 ← a node cannot store three values.
The middle value creates a 2-node parent and this node splits into two new 2-nodes

Insert 47



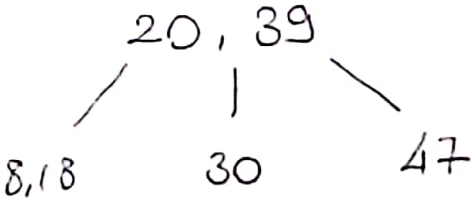
30 ← this node is a 2-node, so we insert directly into the node

Insert 39



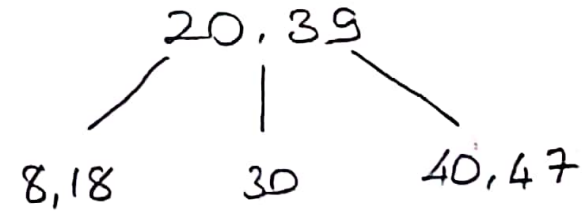
30, 39, 47 ← Because a node cannot store three values, the middle value propagates up to the 2-node parent and this leaf node splits into two new 2-nodes.

Insert 18



8 ← this node is a 2-node, so we insert directly

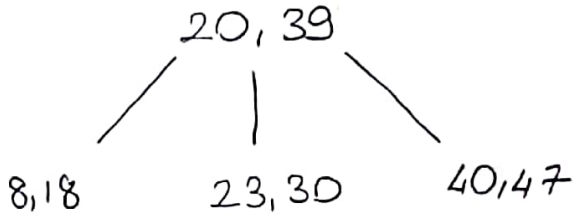
Insert 40



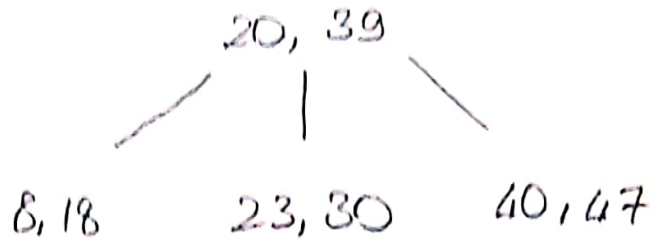
47 ← this node is 2-node, so we can insert directly

Insert 23

30 ← this node is 2-node.
we insert directly.

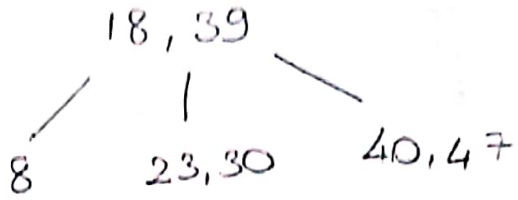


2-3 Tree :



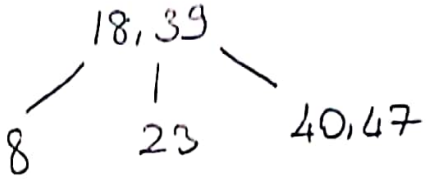
Delete 20

Replace it with its leaf predecessor (18), because 20 is not in a leaf.



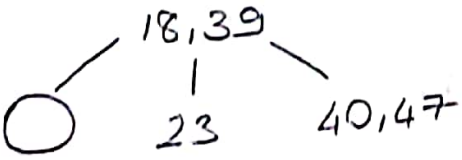
Delete 30

30 is in a leaf, we simply delete it.

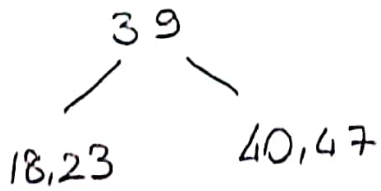


Delete 8

8 is in a leaf, we delete it.

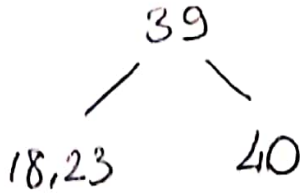


The left leaf is now empty. Merge the parent (18) into its right child (23).



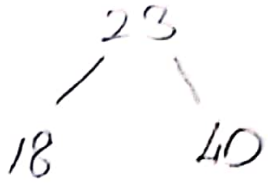
Delete 47

47 is in a leaf, we simply delete it



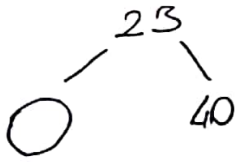
Delete 39

39 is not in a leaf node.
Swap it with its inorder predecessor in a leaf node and delete it from the leaf node



Delete 18

18 is in a leaf node, we delete it.
The leaf node is now empty.



Merge the parent (23) into its right child (40).

23, 40

Delete 40

40 is in a leaf, we delete it.

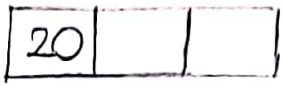
23

Delete 23

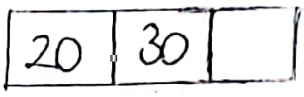
23 is in a leaf, we delete it.

B-Tree With Order 4

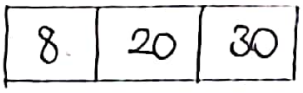
Insert 20



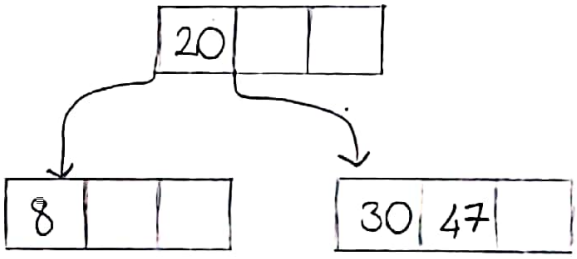
Insert 30
The node is not full



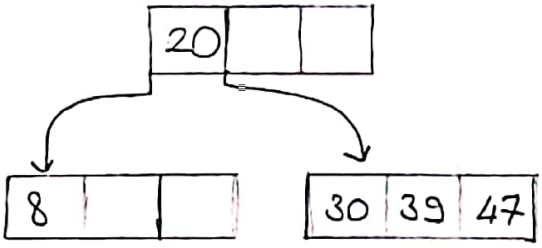
Insert 8
This node is not full



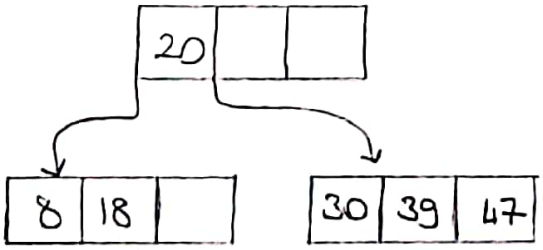
Insert 47
This node is full,
it is split into two nodes,
each containing approximately
half the items, and the
middle item is passed up.



Insert 39



Insert 18

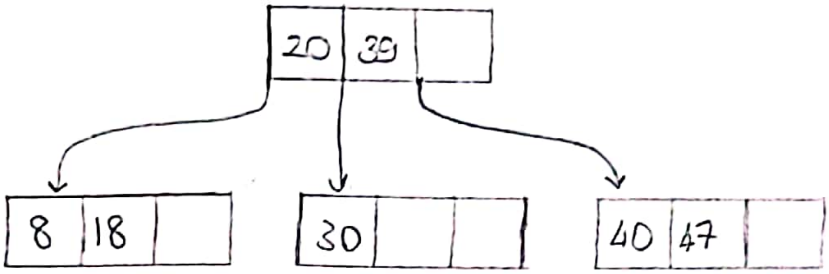


Insert 40

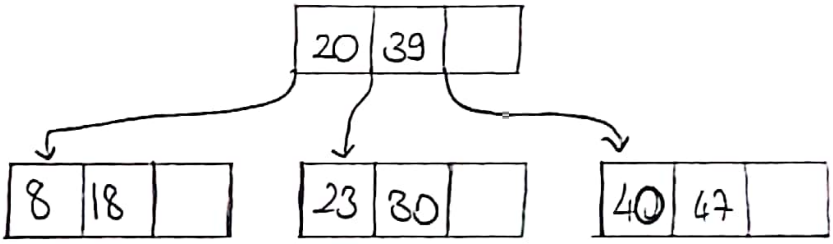
This node is full.

30	39	40	47

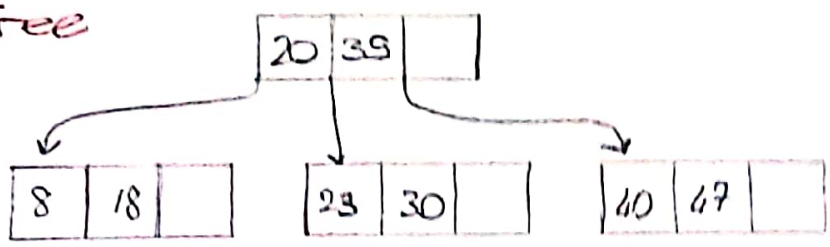
This leaf node is split into two nodes, and its middle item is passed up to its parent



Insert 23

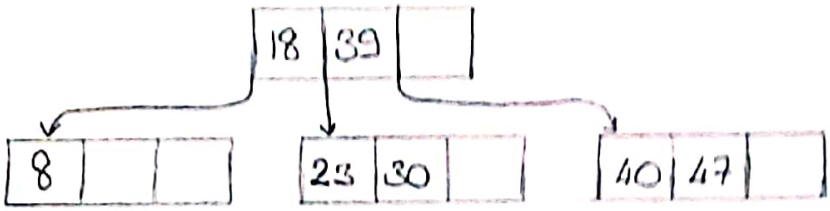


B-Tree



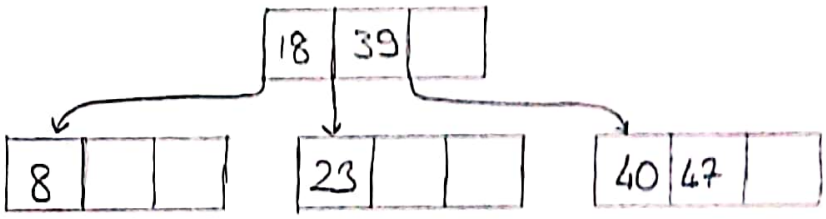
Delete 20

20 is not in a leaf.
It is replaced by its inorder predecessor in a leaf.



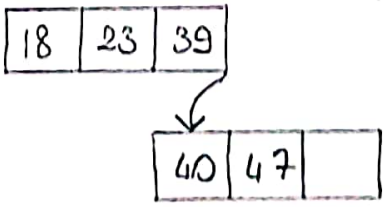
Delete 30

30 is in a leaf, we simply delete it.



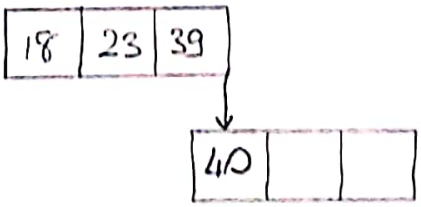
Delete 8

8 is in a leaf. we delete it
The leaf has now zero item, we merge it with its parent and its sibling.



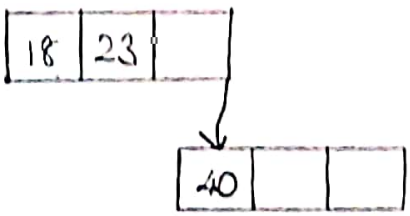
Delete 47

47 is in a leaf, we delete it.



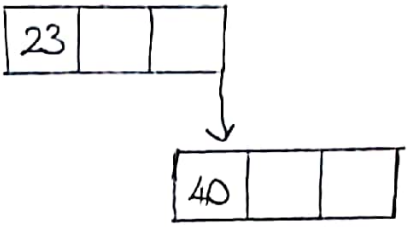
Delete 39

39 is not in a leaf.
It is replaced by its Inorder predecessor in a leaf.
There is no inorder predecessor in a leaf



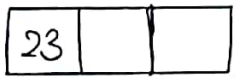
Delete 18

18 is not in a leaf.
It is replaced by
its inorder predecessor
in a leaf.
There is no inorder
predecessor in a leaf.



Delete 40

40 is in a leaf node, we
simply delete it.
The leaf node has a
zero item now. We
merge it with its parent
and its sibling.

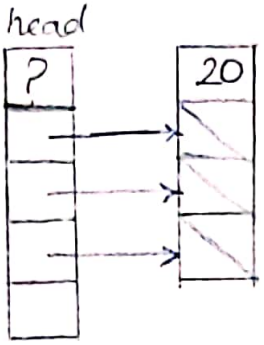


Delete 23

23 is in a leaf node, we simply delete it.

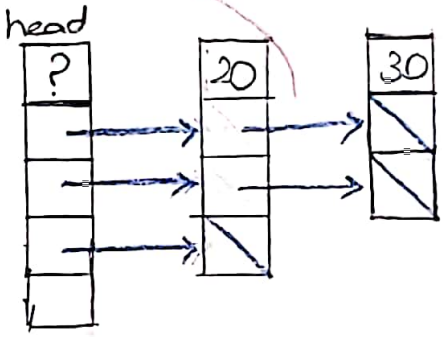
Skip-List :

Insert 20



The random number generetor return 3. Making the node a level-3 node.

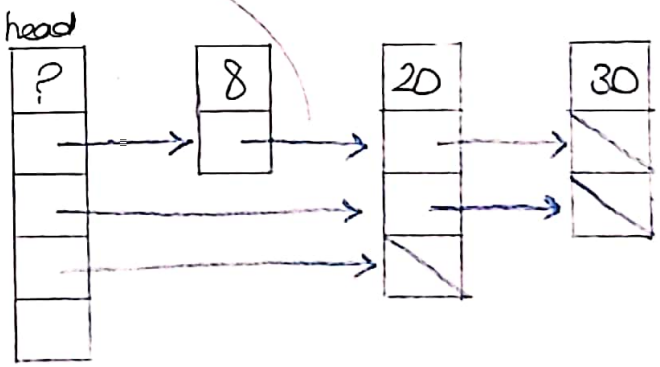
Insert 30



30 greater than 20. So the insertion point is after predecessor(20).

The random number generator return 2. Making the node a level-2 node.

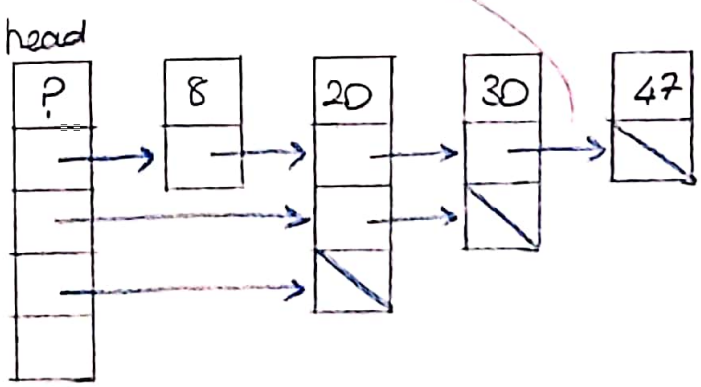
Insert 8



The next node's value is 20, which is greater than 8, so the insertion point is after head.

The generator return 1. Making the node a level-1 node.

Insert 47

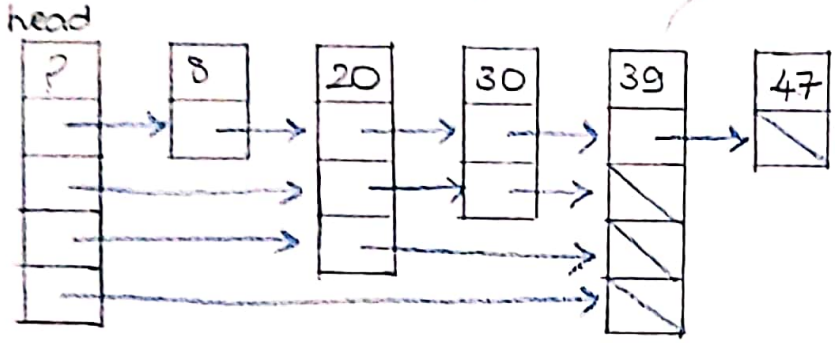


47 greater than 30, So the insertion point is after predecessor (30).

The random number generator return 1 Making the node a level-1 node.

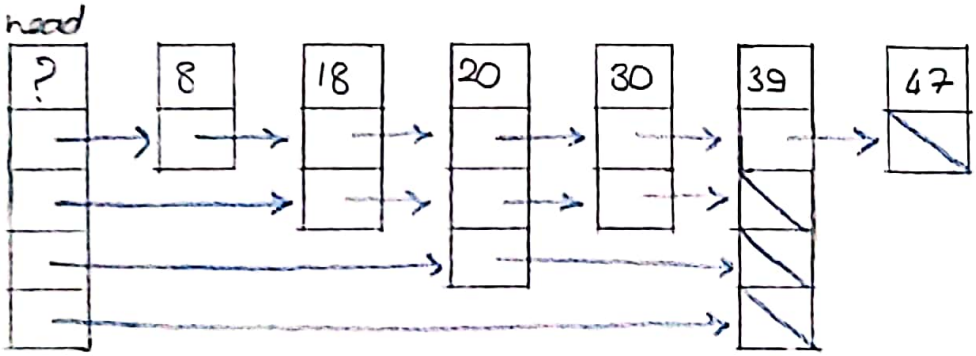
Insert 39

(search)
The next value is 47, which is greater than 39, so the insertion point is after predecessor (30)
The random number generator returns 4.
Making the node a level-4 node.



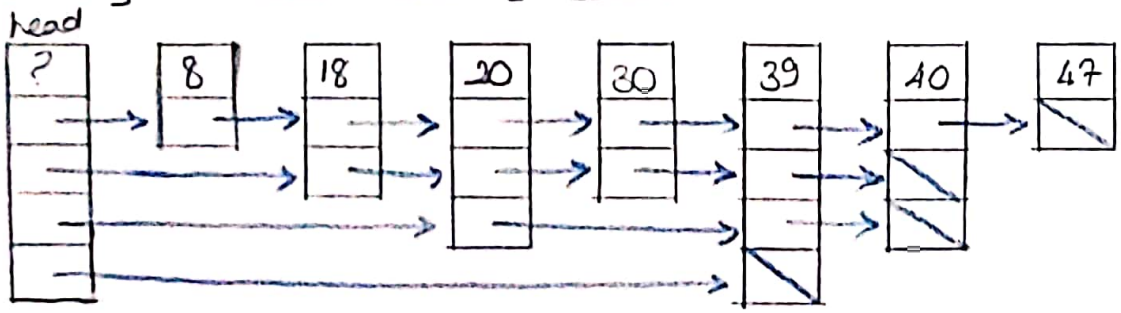
Insert 18

(search)
The next value is 20, which is greater than 18, so the insertion point is after predecessor (8)
The random number generator returns 2.
Making the node a level-2 node.



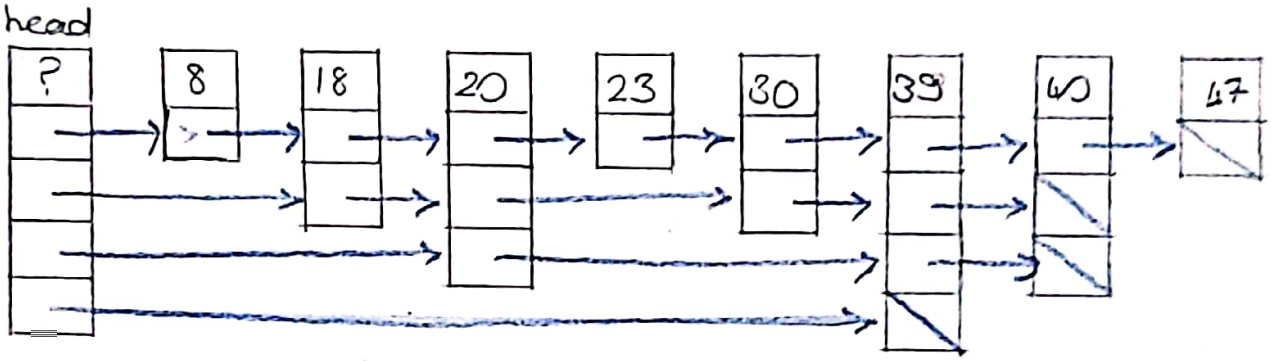
Insert 40

The next value is 47, which is greater than 40, so the insertion point is after predecessor (39)
The random number generator returns 3.
Making the node a level-3 node.



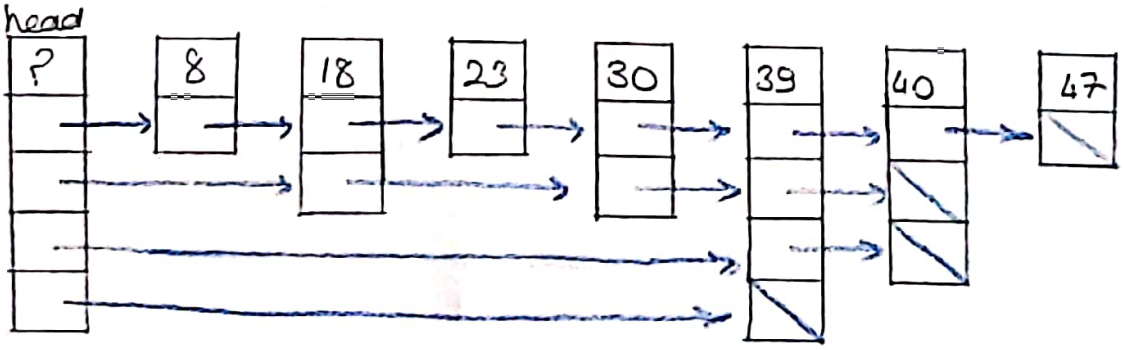
Insert 23

The next value is 30, which is greater than 23, so the insertion point is after predecessor (20).
The random number generator returns 1.
Making the node a level-1 node.



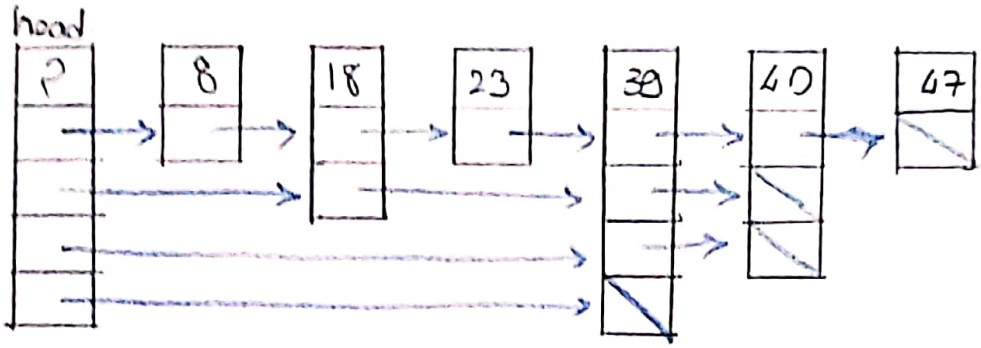
Delete 20

Search 20:
Start with the highest list, in this case, level 4.
Since 20 < 39, we move back to head and search the level 3.
20 is found and is removed

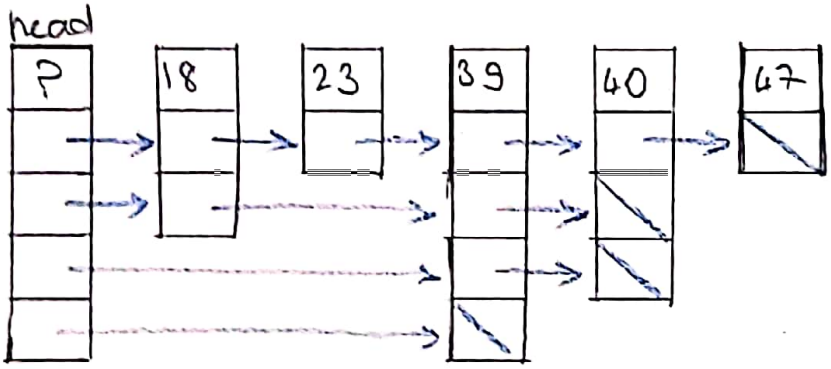


Delete 30

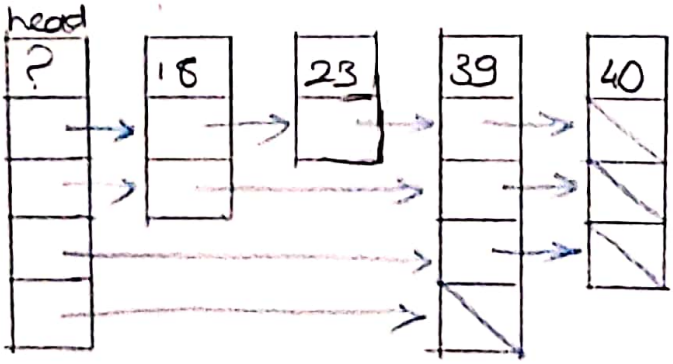
Search level 4
30<39, move back to head and search level 3.
30<39, move back to head and search level 2.
30>18, move to the next node
30 is found and deleted.



Delete 8
Search level 4
8 < 39, move back, search level 3
8 < 39, move back, search level 2
8 < 18, move back, search level 1.
8 is found and deleted



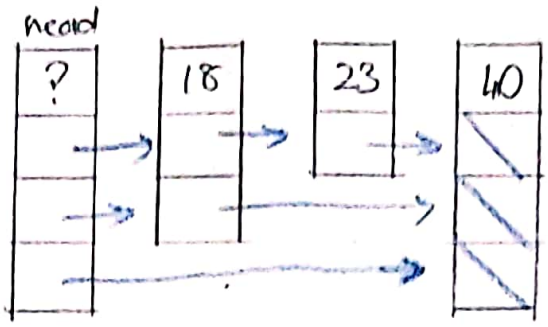
Delete 47
search level 4
47 > 39, search level 3 because its end of the list.
end of the list ⇒ search level 2.
move to the next node.
47 > 40, end of the list, search level 1.
move to the next node.
47 is found.



Delete 39

Search level 4

39 is found



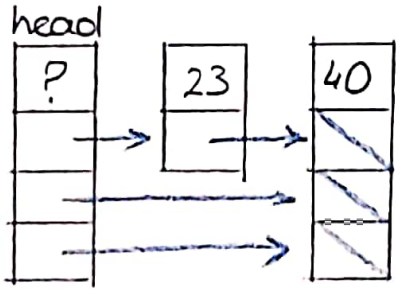
At level 4, there is no element. So we will decrement level of skip list by 1.

Delete 18

Search level 3

18 < 40, move back, search level 2

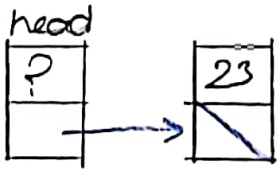
18 is found



Delete 40

Search level 3

40 is found.



At level 3 and 2, there is no element.

So we will decrement level by 2.

Delete 23

search level 1

23 is found and deleted