17/8/21

B TREES

A B tree of order m is a m-way search tree and hence may be empty. If not empty then the following properties are satisfied.

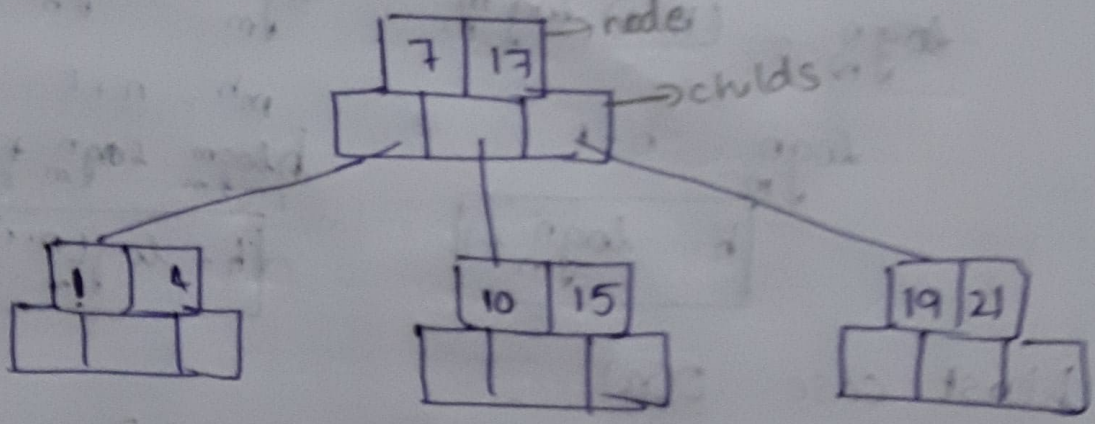
① The root node must have at least 2 children and at most m children.

② All other nodes except the root nodes must have atleast $\lceil \frac{m}{2} \rceil$ non empty child nodes and at most m - non empty children.

③ The number of keys is always 1 less than the numbers of children.

④ All external nodes are at the same level.

eg :-



Sol:

if order - 5

root node shud have nodes - 2 to 5

other nodes - $\lceil \frac{5}{2} \rceil = 3$

min - 3 max - 5

* A B-tree of order 3 is called - 2-3 tree

Also called Symmetric Binary B-tree

* A B-tree of order 4 is called 2-4 tree (or) 2-3-4 tree.

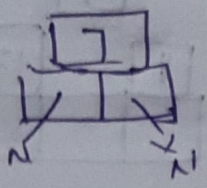
Insertion

Case-1

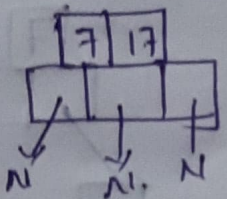
If the node can accomodate k, then insert it in the node & adjust the pointers.

eg: Insert 7, 17 Order-3.

Insert-7

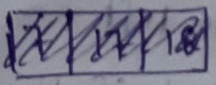


Insert-17 - 7<17



root - 2 to 3.
other nodes - 2 to 3.

Insert-18 -, 7<18, 17<18.



root can accomodate but if it is so, childs will be 4.
But order is 3.
So, u have to insert in another place.

Case-2

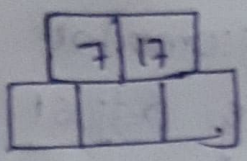
If the node can't accomodate k, then logically insert the k in the appropriate position and split the list in to two at its median. Kmedian. Kmedian is pulled up and inserted in the parent node. The keys less than Kmedian

from the left child of Kmedian and the keys greater than Kmedian from the right child of Kmedian

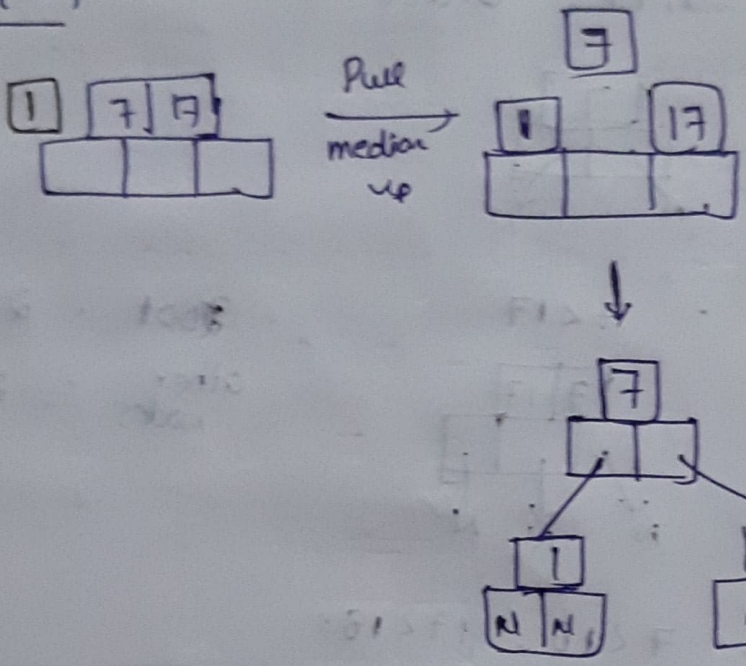
Note -

The B tree grows UPWARD.

eg:-

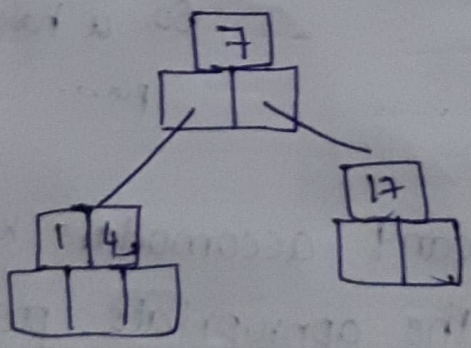


Insert -1



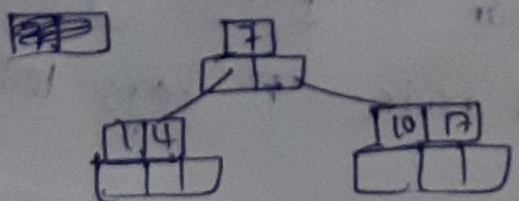
Insert -4 7 > 1 - move left.

1 < 4 - check if the node can be accomodated.



Insert -10 7 < 10 - move right.

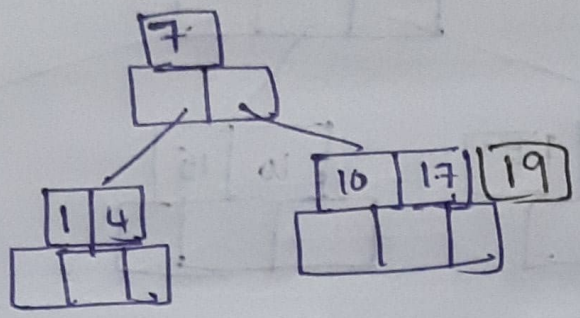
17 > 10 - Check if 10 can be accomodated



Insert -19 $7 < 19$ - move right

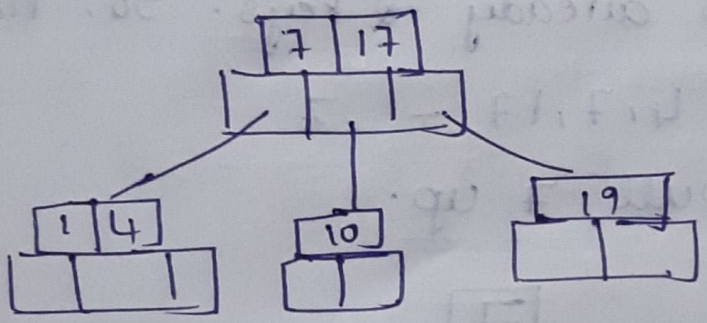
$19 > 10$ & $19 > 17$ - check if this can accomodate.

As order is 3, we can't insert 19.



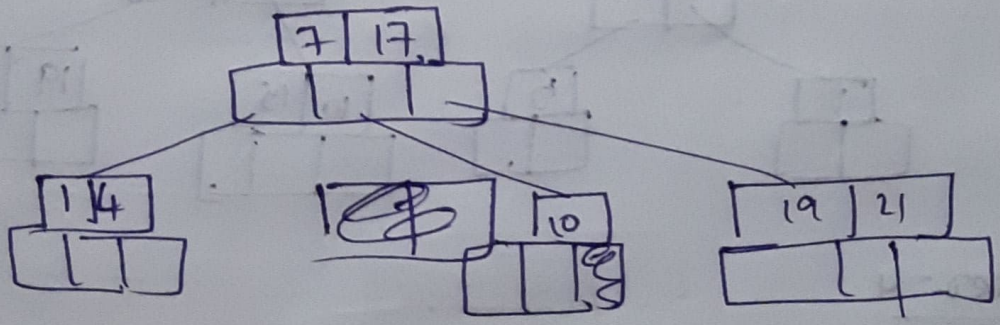
Median of 10, 17, 19 - 17.

↓ pull it up.

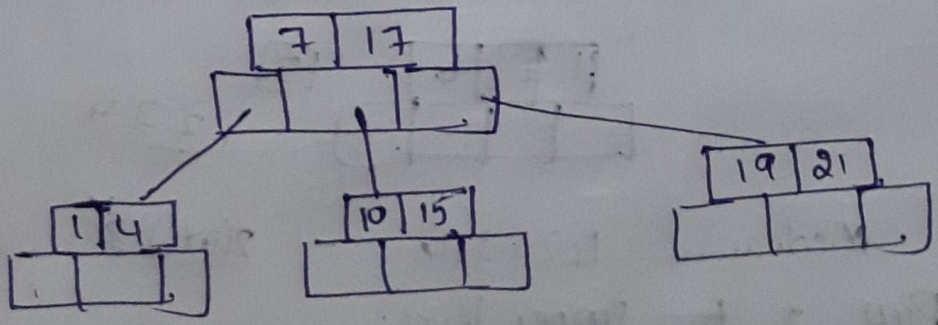


Insert -21 $7 < 21$ & $21 > 17$ - move right

$19 < 21$ - check if node can accomodate.

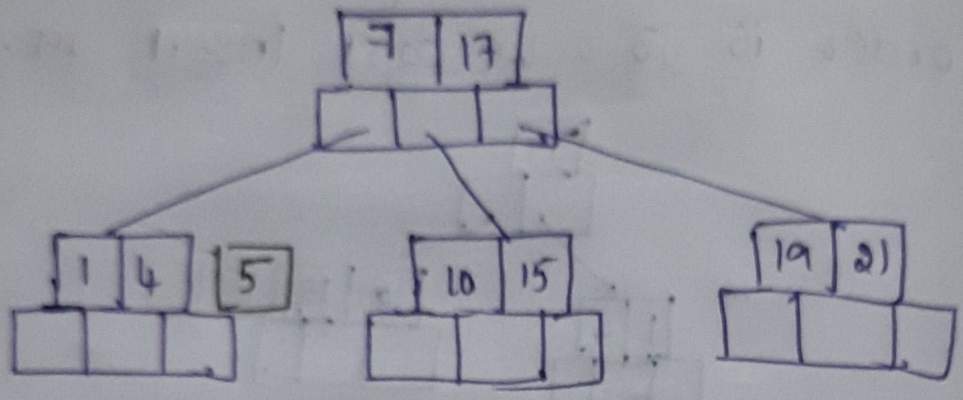


Insert -15 $7 < 15$ & $17 > 15$



Insert 5 — 5 < 7 — move left.

5 > 1 & 4 < 5 — Check if it can accomodate.

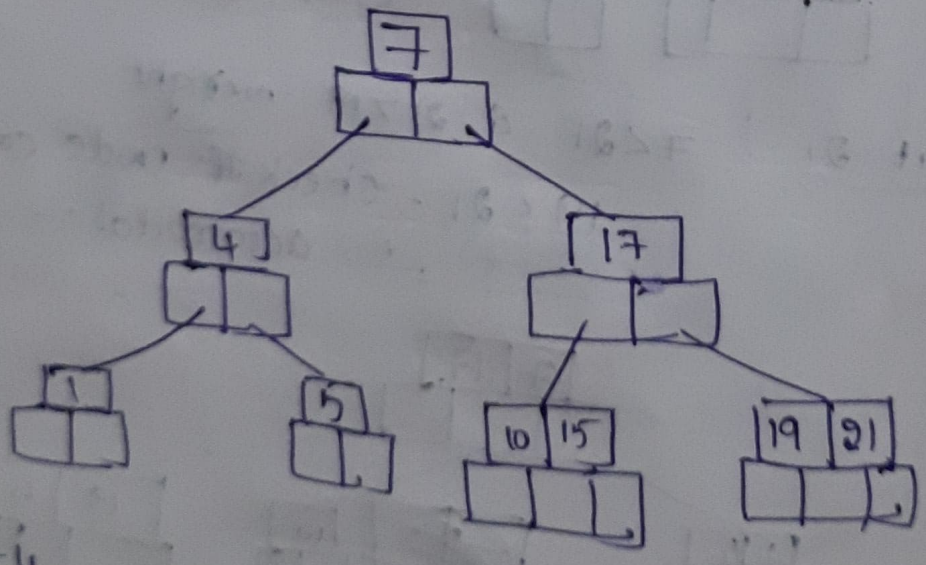


Median 1, 4, 5, 7 = 4.

Pull median '4' - to up.

Root has already 2 keys. So, find median of 4, 7, 17 — 7

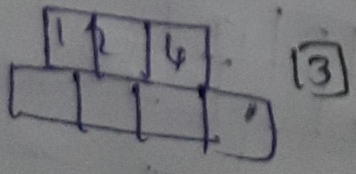
So, pull 7 up.



Q)

Order - 4

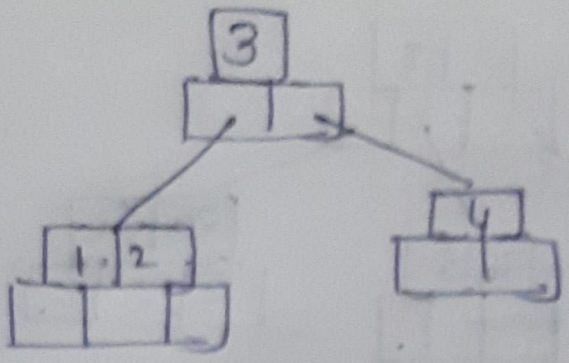
Insert - 1, 2, 4, 3, 8, 5, 95, 65, 24, 32



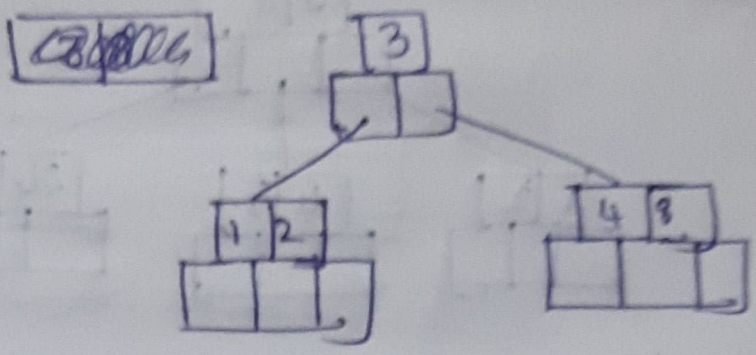
1 2 3 4

Median - 1, 2, 3, 4 — 2 (or) 3 ✓

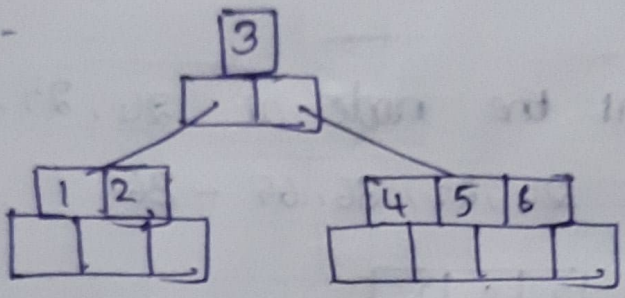
Pull 3 to upper level



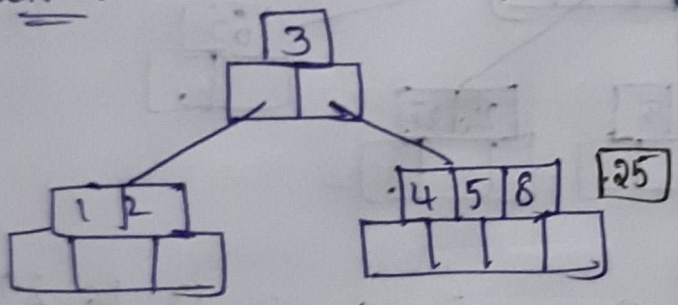
Insert - 8 - 3 < 8.



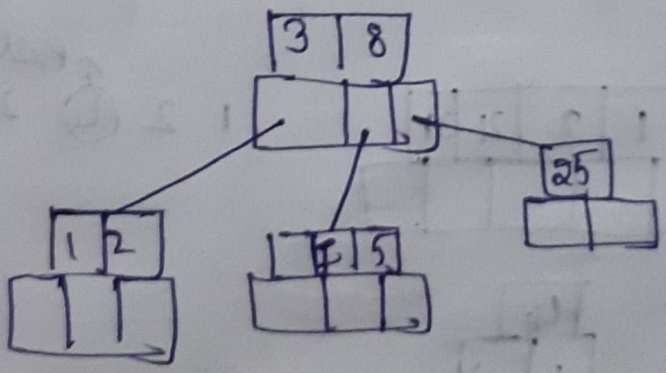
Insert - 5 -



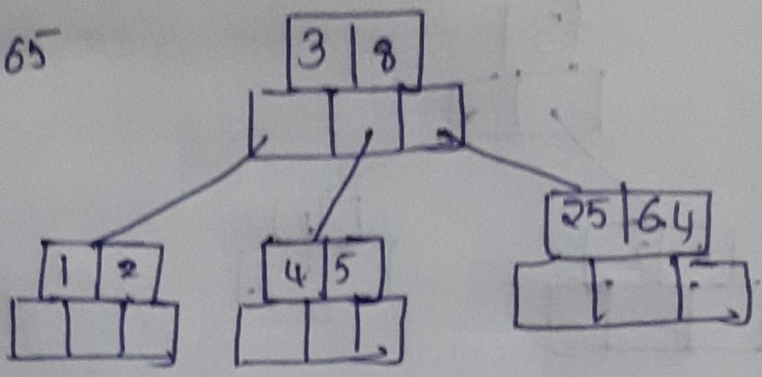
Insert - 25



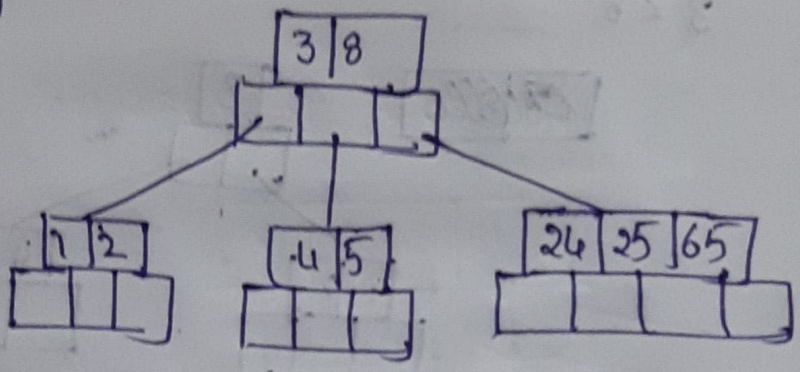
Median of 4, 5, 6, 25 - 8.



Insert - 65



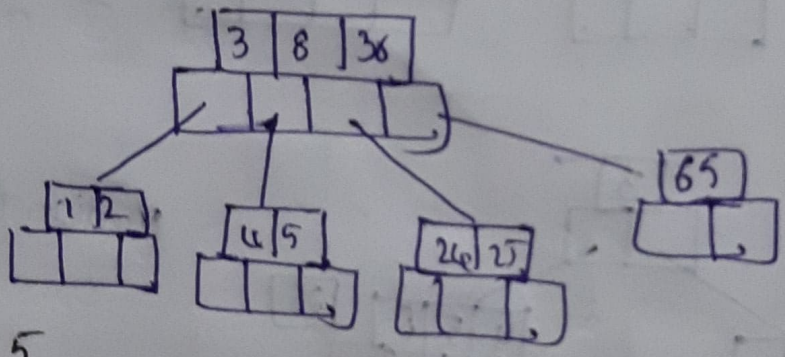
Insert - 24



Insert - 36

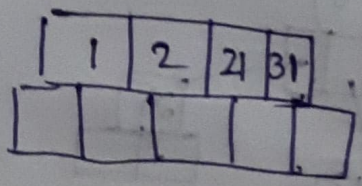
Logically insert the node at 24, 25, 36, 65

Median of 24, 25, 36, 65 - 36.



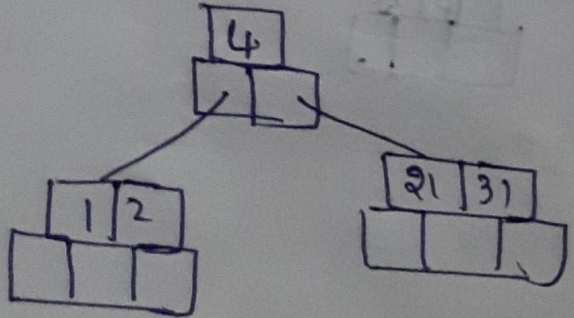
Q:- Order - 5

Insert 21, 31, 1, 2, 4, 5, 8, 12, 13, 16, 29, 30, 45, 48.

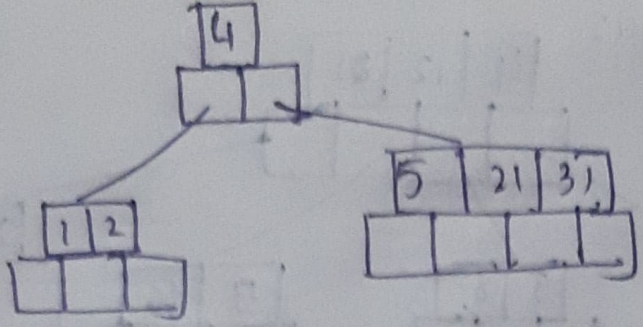


1 2 (4) 21 31 ← Pull

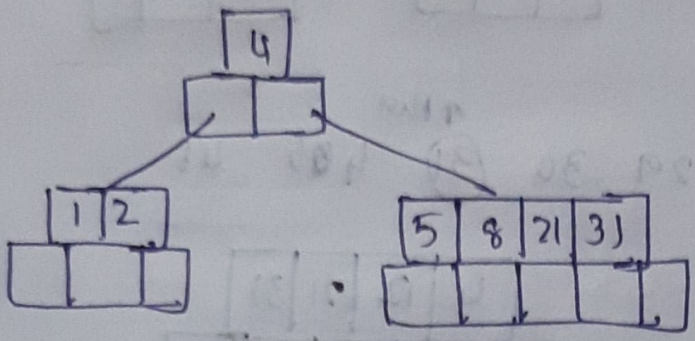
Insert - 4



Insert - 5



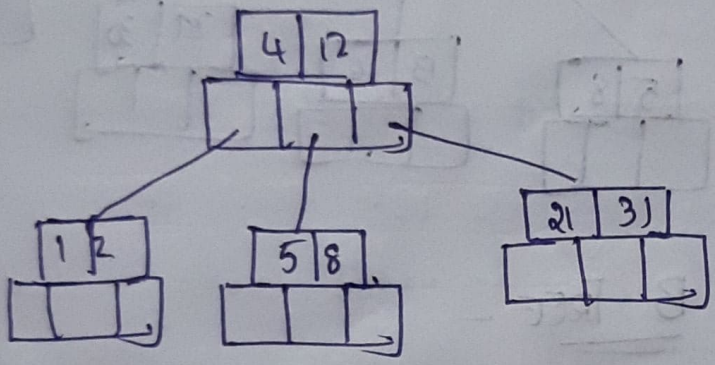
Insert - 8



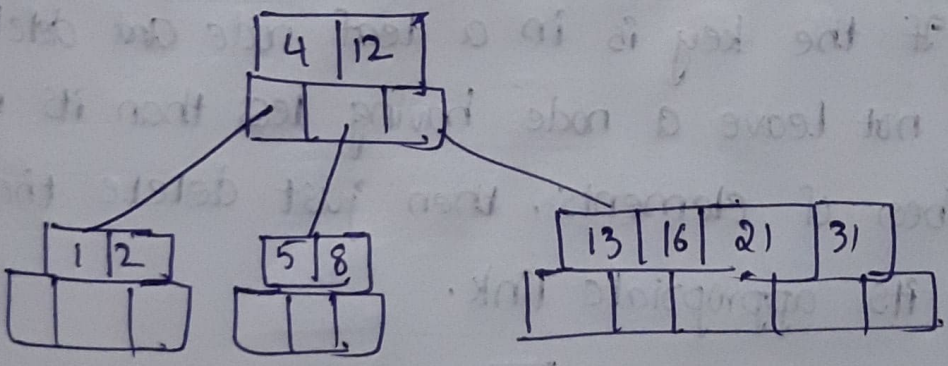
Insert - 12

Pull up.

Median of 5, 8, 12, 21, 31 - 12



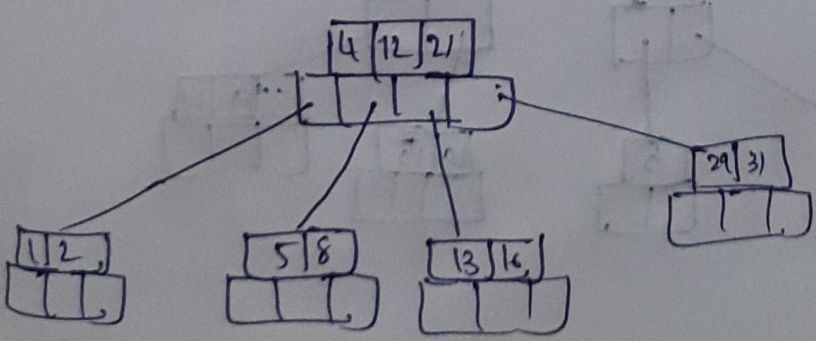
Insert - 13 & 16



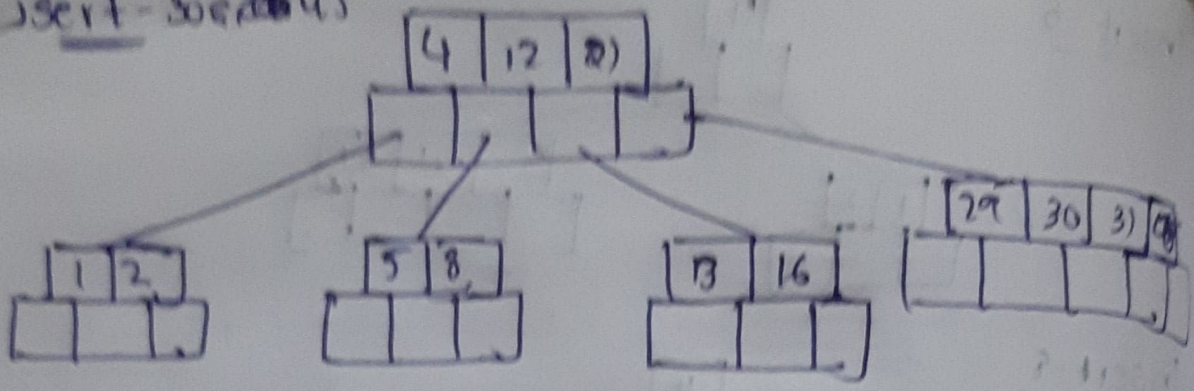
Insert - 29

Pull up.

13 16 (21) 29 31

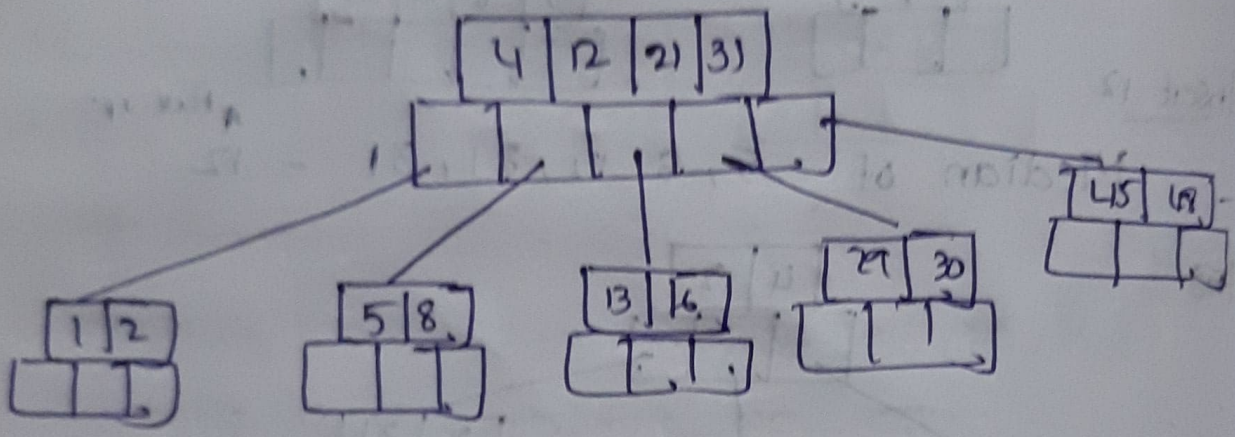


Insert - 30 & 45



Insert - 48

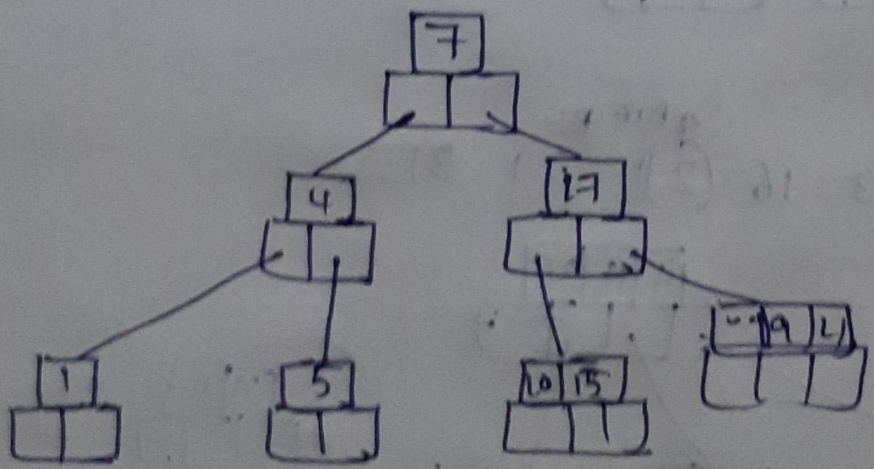
29 30 (31) ↑Push 45 48



Deletion in B TREE -

Case-1

If the key is in a leaf node and deletion does not leave a node having less than its min. number of elements, then just delete the key with its appropriate link.



Case-2

If the key to be deleted is in a non-leaf node and if deletion of key satisfies the minimal rule, then replace the deleted key with the largest key in the left subtree or the smallest key in right subtree.

Case-3

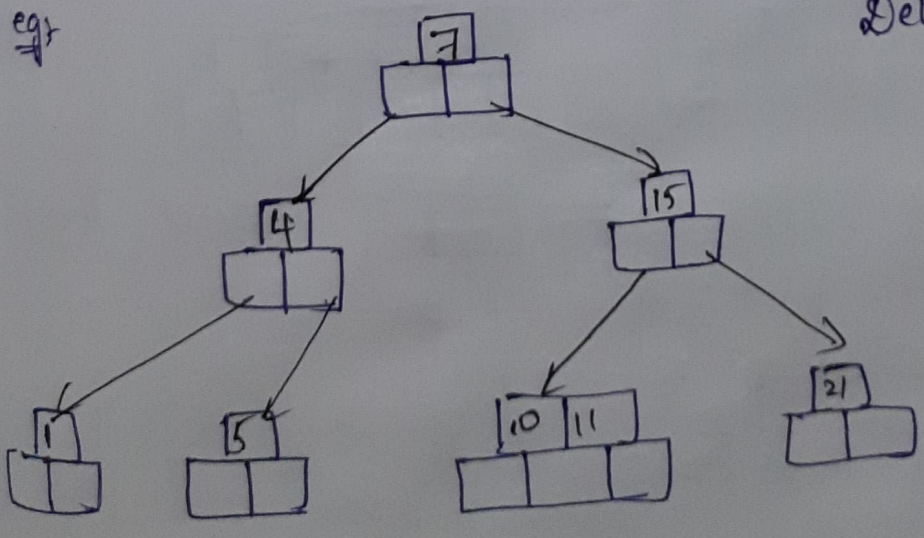
If deletion of a key (K) from a node leaves it with less than its minimum number of elements, elements are borrowed from one of its left or right sibling nodes.

Thus, if the left sibling has keys to spare, move the largest key from the left subtree to the parent node. The intervening element in the parent node is moved down to set the vacancy created by K.

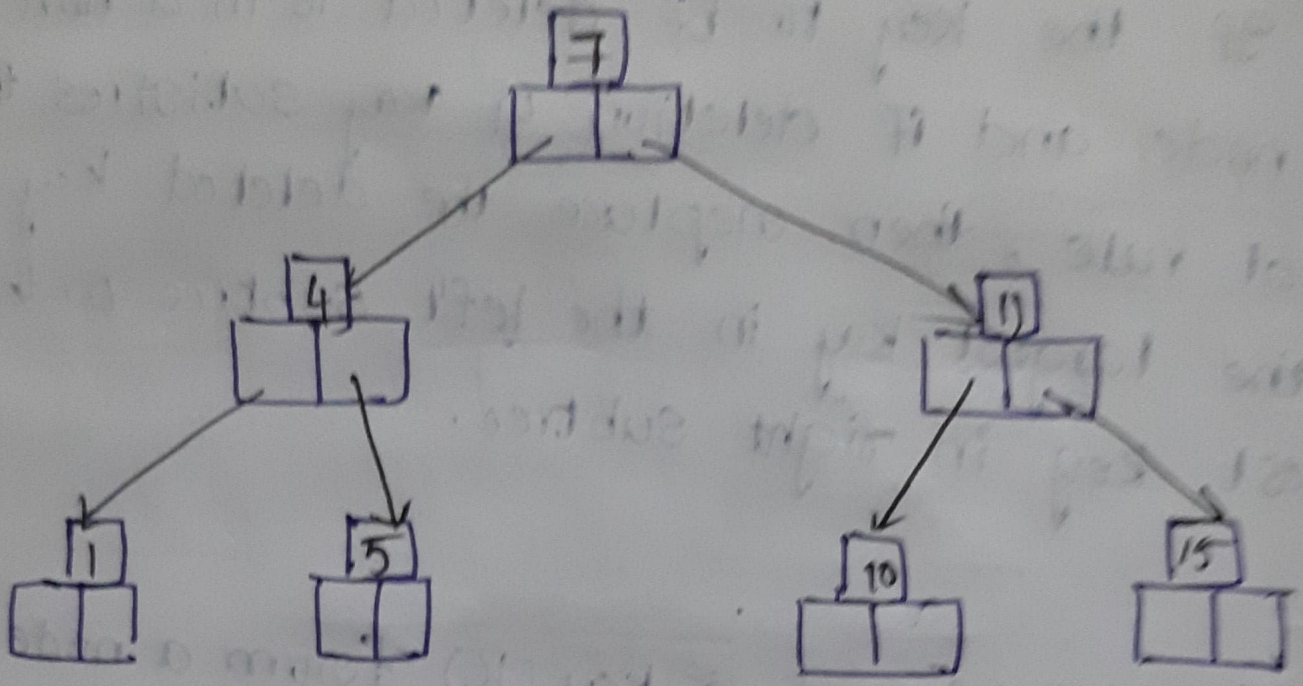
23/8/21

eg:

Delete 21



21 has left sibling.



Case 4

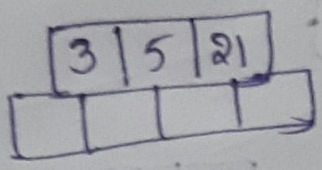
If deletion of a key 'K' leaves the node with less than its minimum number of elements & if both siblings do not have elements to spare, the ~~merged~~ node is merged with one of the sibling nodes along with the intervening element in parent node.

Q: Order - 4

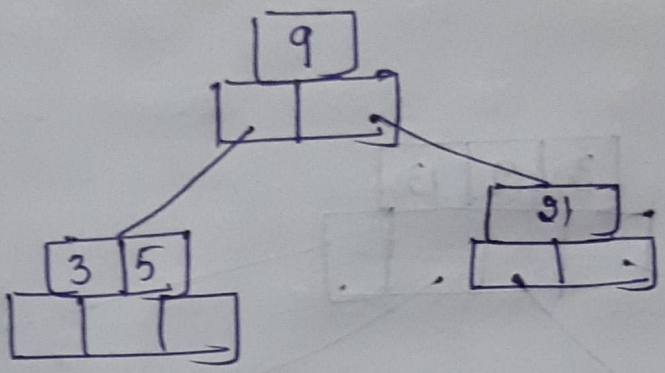
Insert 5, 3, 21, 9, 1, 13, 2, 7, 10, 12, 4, 8.

root: 2 to 4
else: 2 to 4

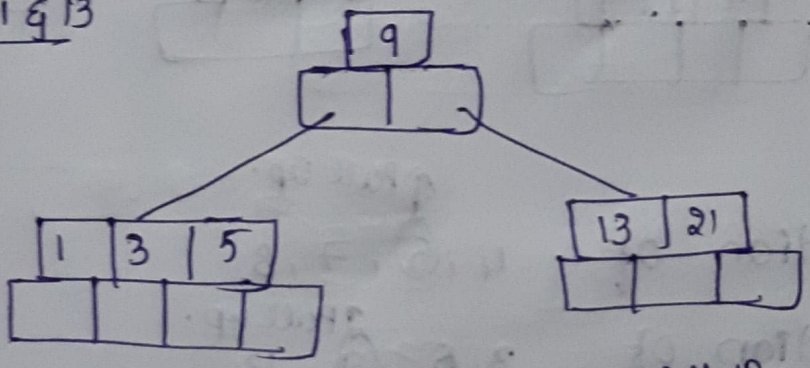
Insert - 5, 3, 21



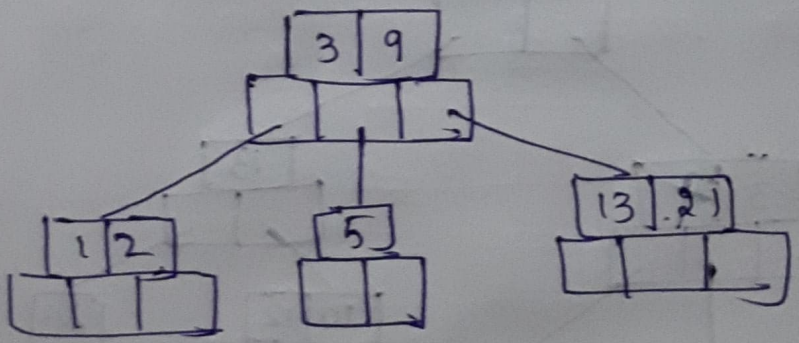
Insert 9 - Median of 3, 5, (9), 21. Pull up.



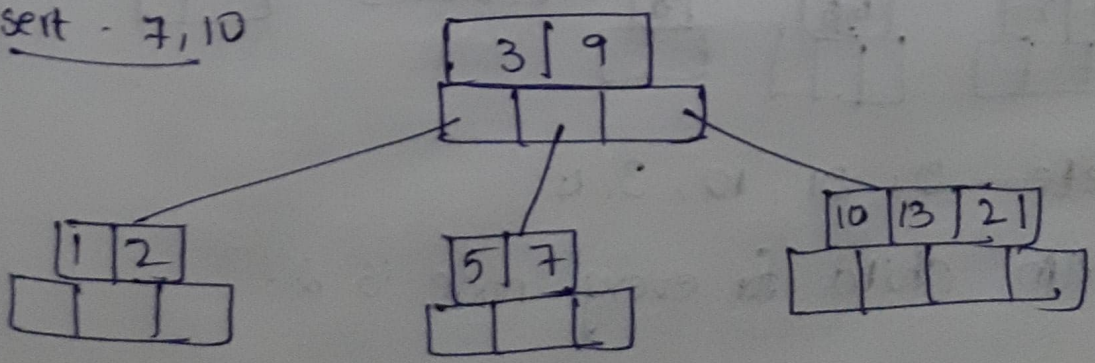
Insert 1 & 13



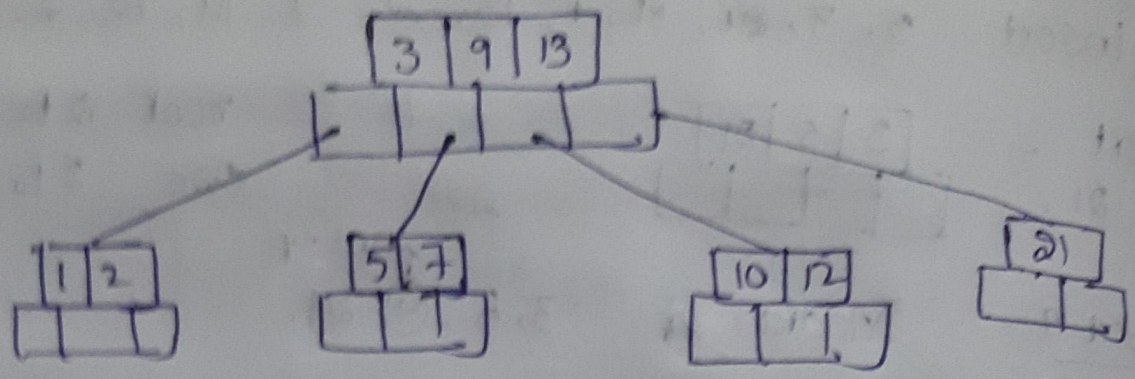
Insert - 2 Median of 1, 2, (3), 5 - Pull up



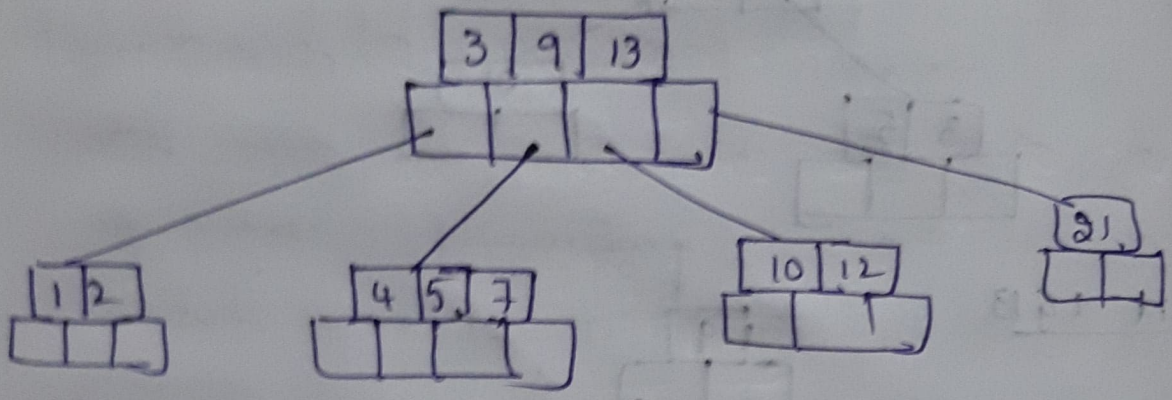
Insert - 7, 10



Insert - 12 - Median of 10, 12, 13, 21 — Pull up.



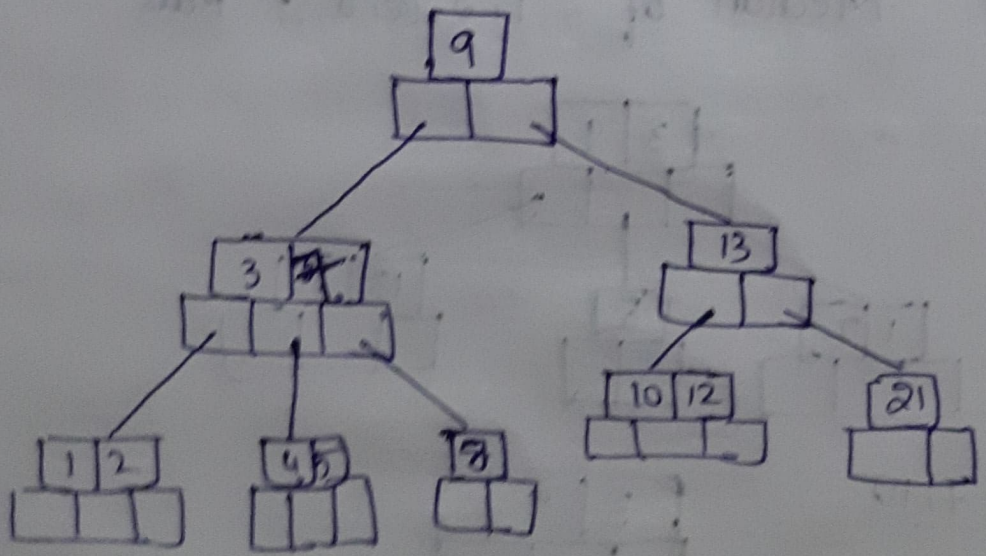
Insert - 4



Insert - 8

Median of 4, 5, 7, 8 — Pull up. Pull up

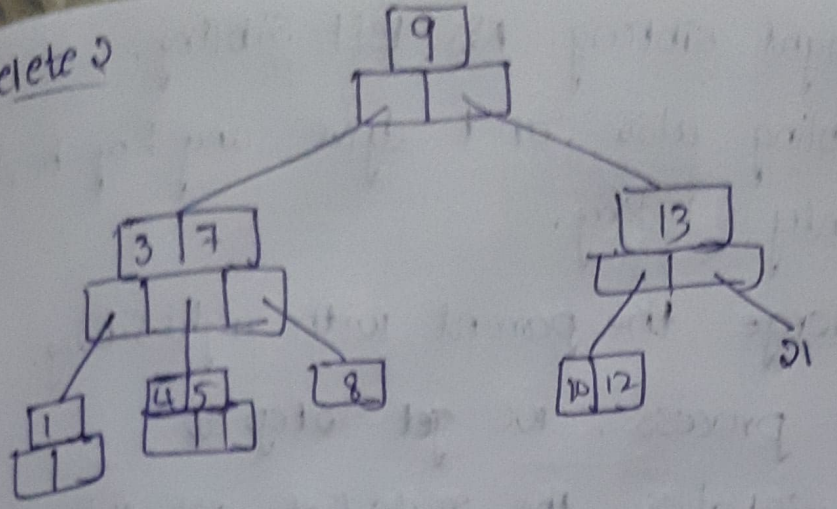
Median of 3, 7, 9, 13 — Pull up.



Delete 2, 21, 10, 3, 4

Min child for every node is 2.

Delete 9



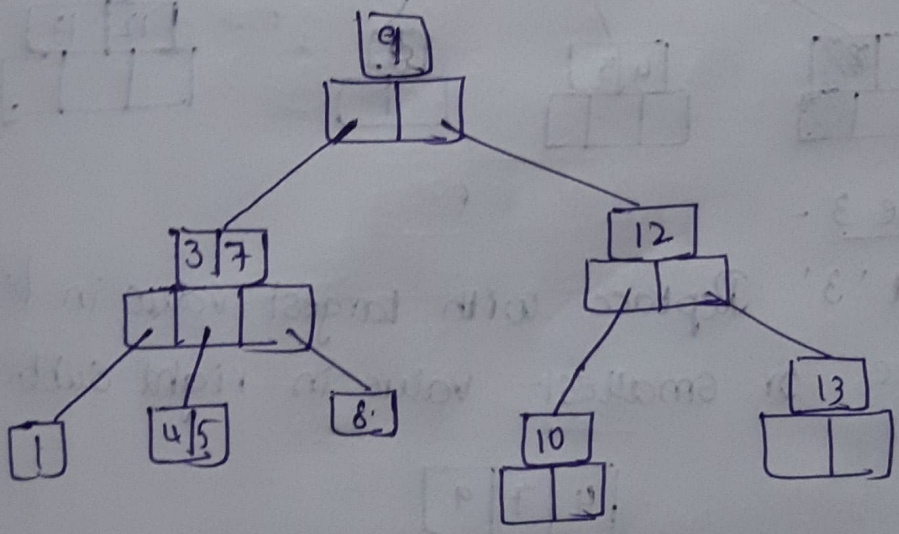
Delete - 21

Deleting 21 makes a null node which violates min num of nodes for a child.

21 has left sibling.

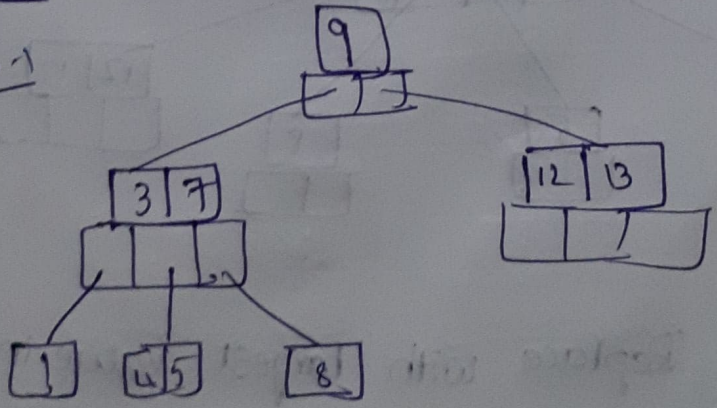
12 is greatest in left sibling.

12 to parent node ; 13 to 21's place.



Delete - 10

Step-1



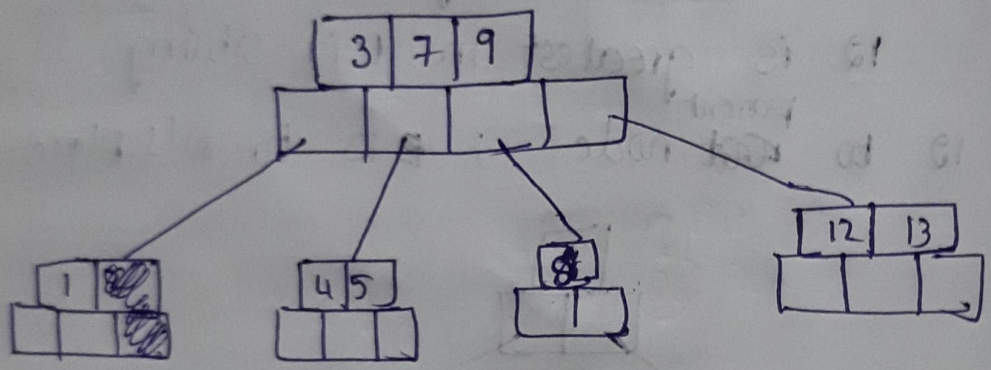
10 has right sibling. No left sibling.
Right sibling also can't give any key bcz it has only 1 key.

So, Merge the parent with right sibling.

After this process, we get step ①

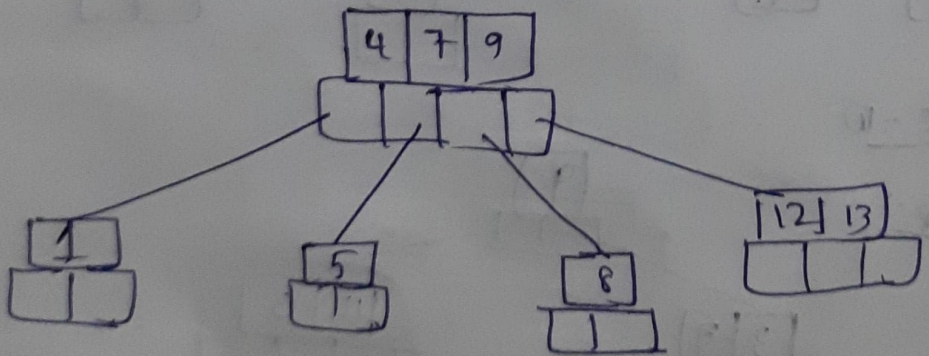
But it violates the rule that all external nodes shud be at same level. So, we have to reconstruct the Btree.

So, pull the left nodes of 9 to above.



Delete 3 -

Del '3' Replace with largest value in left subtree or smallest value in right subtree.

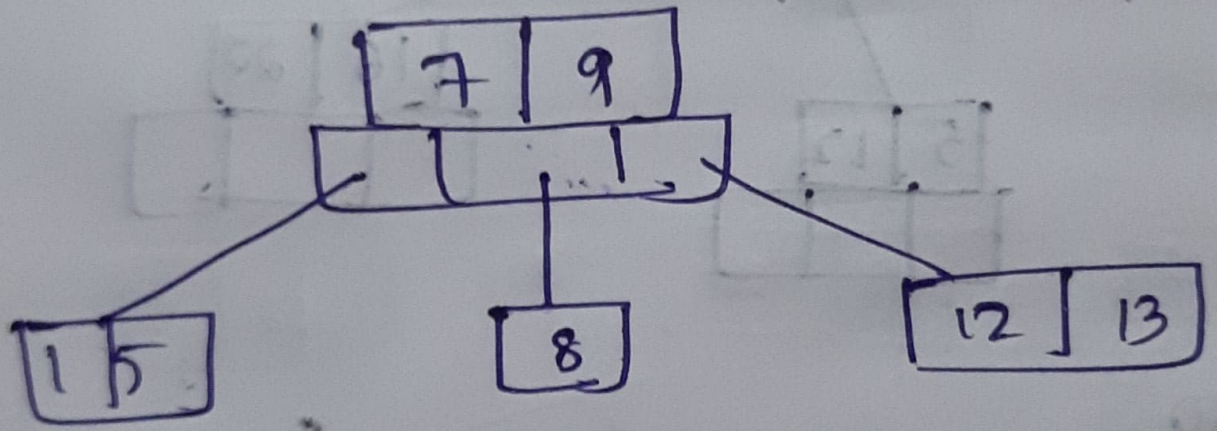


Delete-4 -

Del '4' Replace with largest value in left tree or Smallest value in right subtree.

But deleting '4' we can't replace with

'1' or '5' bcz it violates minimum num of child for a node. So, merge 1 & 5 by deleting 4. (delete the node & merge the children of the node) when u don't have any other key to replace it with



Q: Order - 5

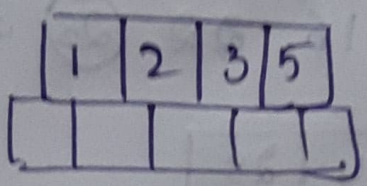
Insert 5, 1, 0, 3, 12, 16, 14, 22, 28, 30, 35

13, 19, 40, 18, 25, 45, 50

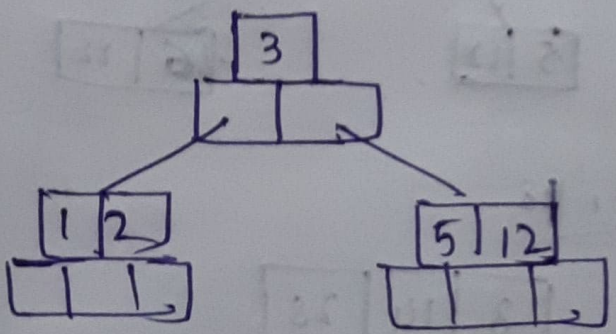
Delete 50, 45, 1, 2, 30, 19, 14, 13, 18

Insert -

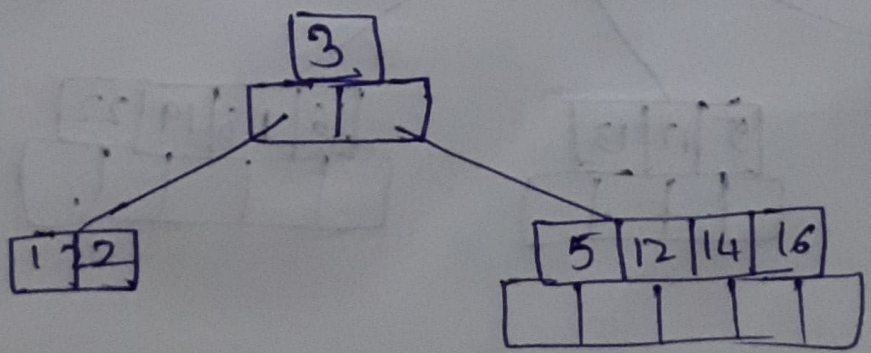
1, 2, 3, 5



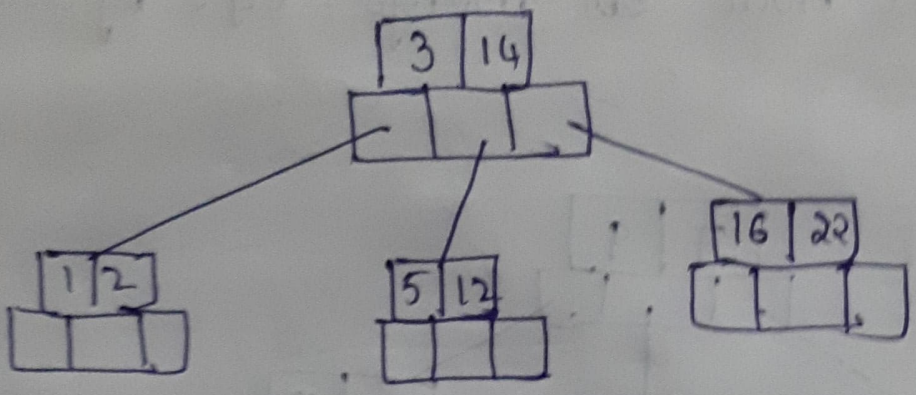
Insert 12 - Median of 1, 2, (3), 5, 12 ↑ Pull up.



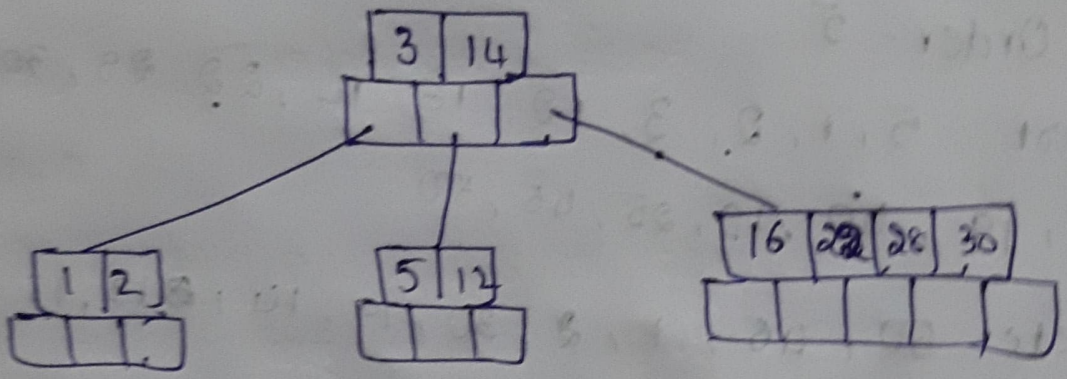
Insert 16 & 14 -



Insert 22 - Median of 5, 12, (14), 16, 22 ← Pull up

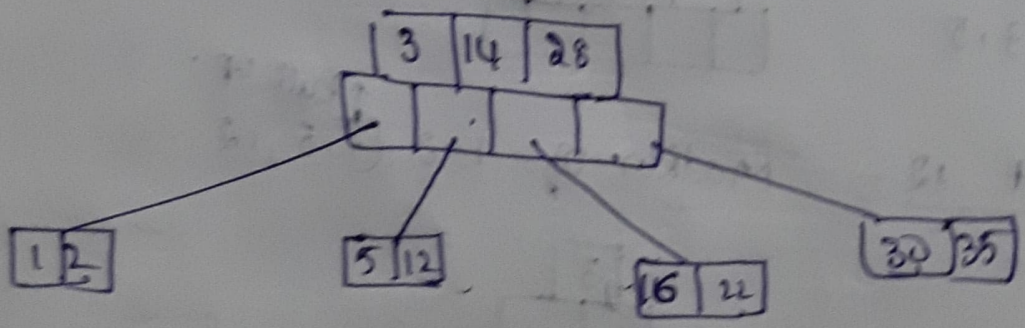


Insert 28 & 30 -

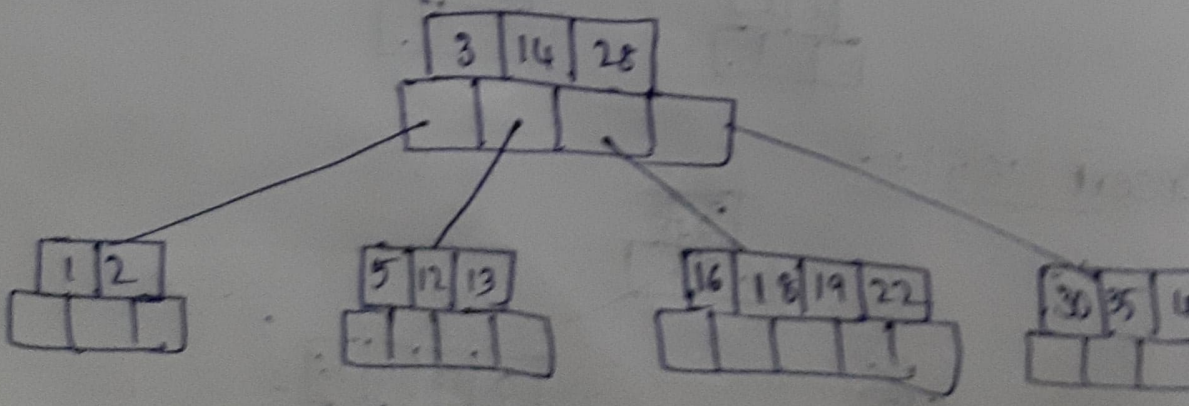


Insert 35 -

Median of 16 22 (28) 30 35 ← Pull up.



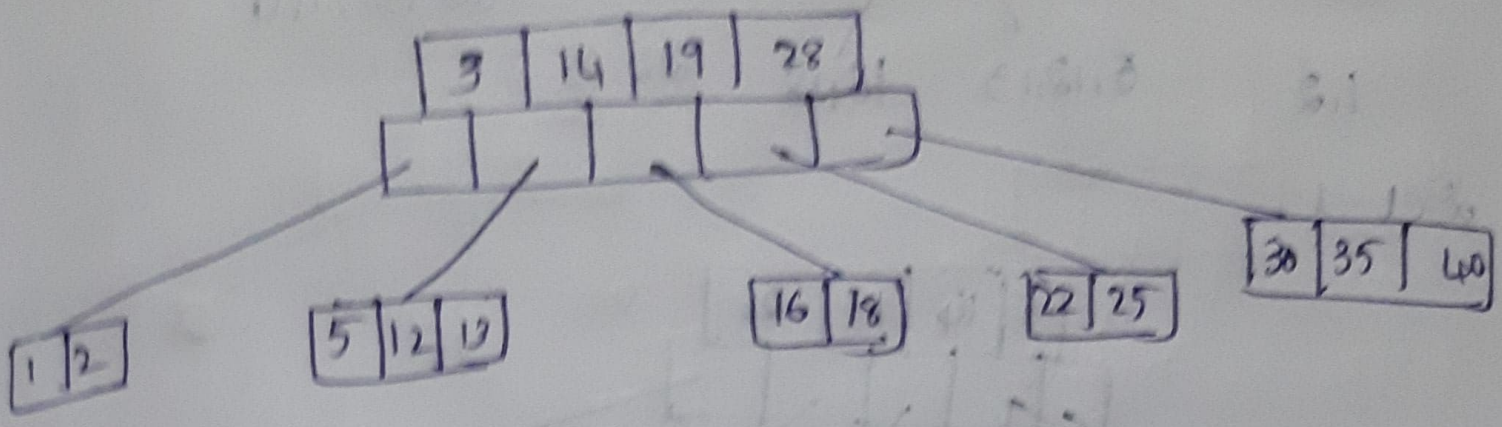
Insert 13, 19, 40, 18



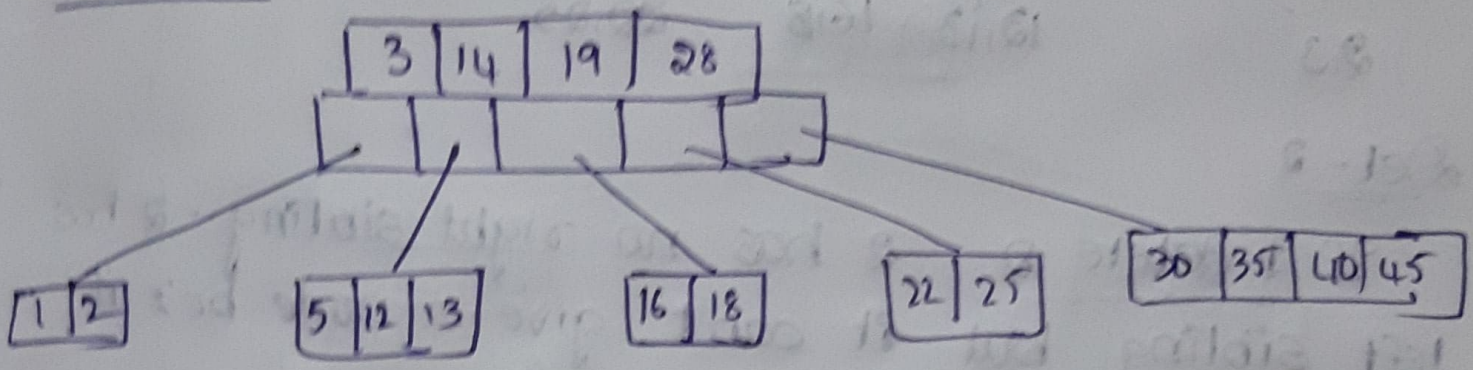
Insert 25

Pull up.

Median of 16 18 (19) 22 25



Insert 45 -



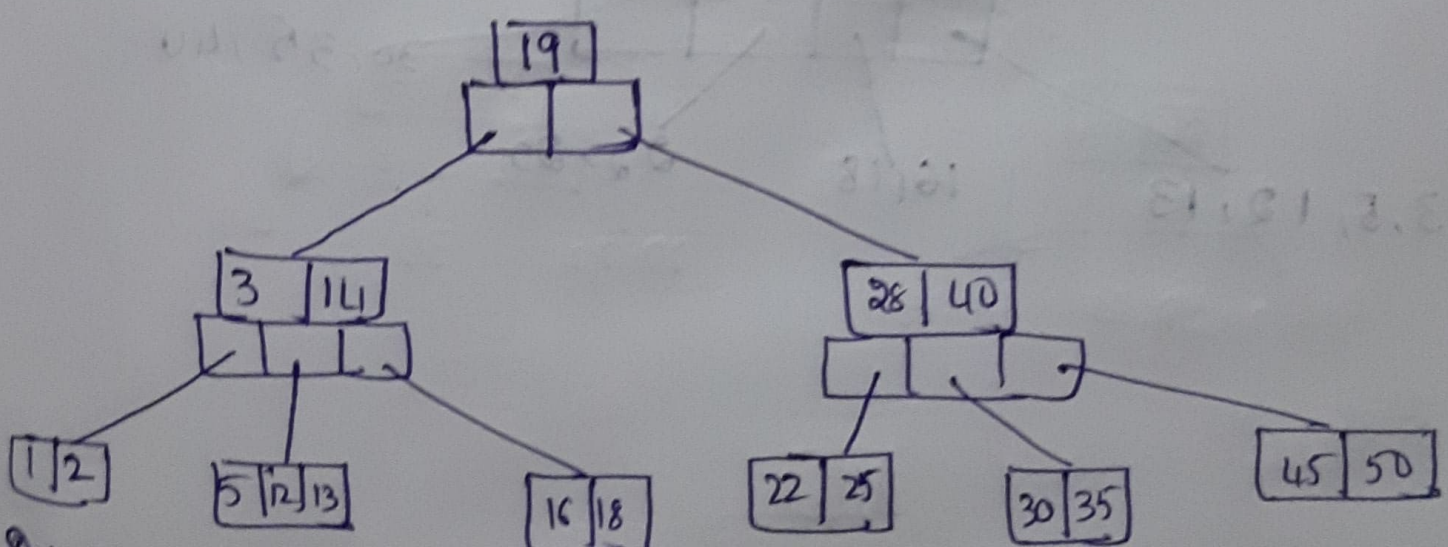
Insert 50 -

Pull up.

Median of 30 35 (40) 45 50

Pull up.

Median of 3 14 (19) 28 40



ANALYSIS OF B TREE

B tree is a m-way search tree Complexity is O(h).

UPPER BOUND -

A B tree of order m and height h with n elements satisfies $n \leq m^h - 1$.

LOWER BOUND -

Min. num of nodes at level 1 - 1

Min. num of nodes at level 2 - 2

Min. num of nodes at level 3 - $2 \cdot \lceil \frac{m}{2} \rceil$

Min. num of nodes at level 4 - $2\lceil \frac{m}{2} \rceil^2$

Min. num of nodes at level h+1 - $2\lceil \frac{m}{2} \rceil^{h-1}$

Number of elements $n \geq 2\lceil \frac{m}{2} \rceil^{h-1} - 1$

$$\log_m (n+1) \leq h \leq \log_{\lceil \frac{m}{2} \rceil} (n+1/2) + 1$$