B tree

(i) Balanced m way search tree

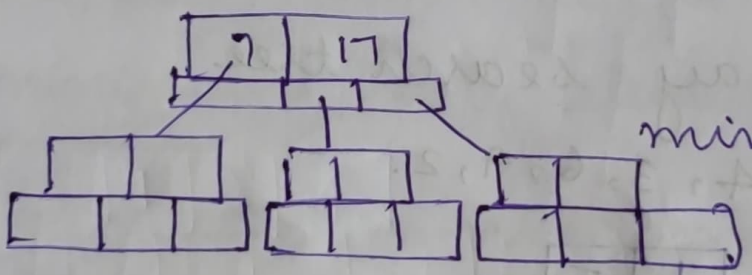
(ii) Root node should have atleast 2 children, atmost m children (non empty)

(iii) internal (except root node) node should have min of $\lceil \frac{m}{2} \rceil$ and max of m children. (non empty)

(iv) All leaf node should be at the same level.

eg: Order 3 (2-3) tree

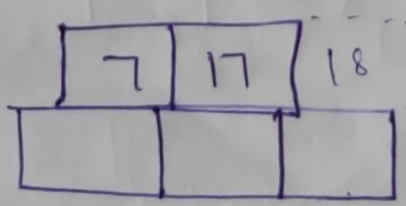
root ⇒ min = 2, max = 3.



Symmetric binary B tree (2-3) tree.

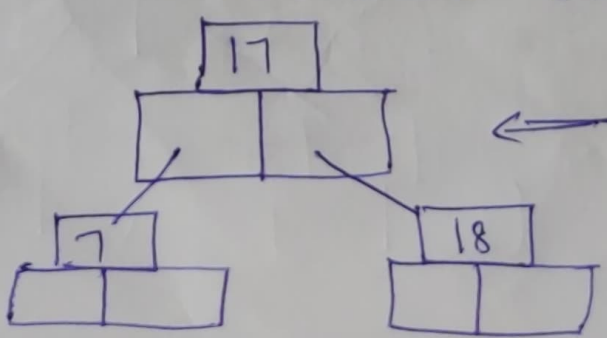
Order 4: (2-4) trees (or) (2-3-4) tree

Insertion in B tree



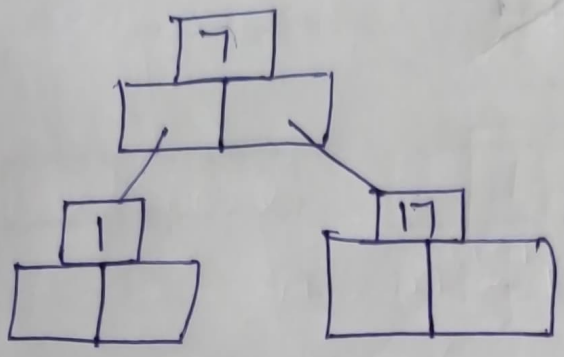
Insert 18.

(i) Median of 7, 17, 18 is 17



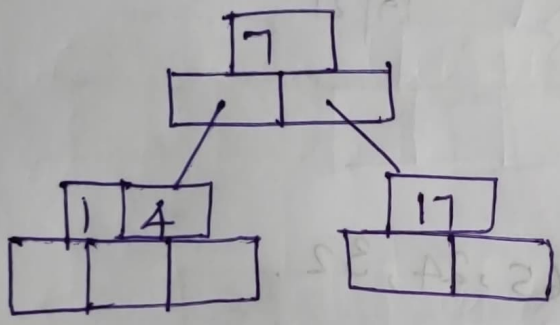
⟵ Pull 17 up.

Insert 1 in [1 | 7 | 17]

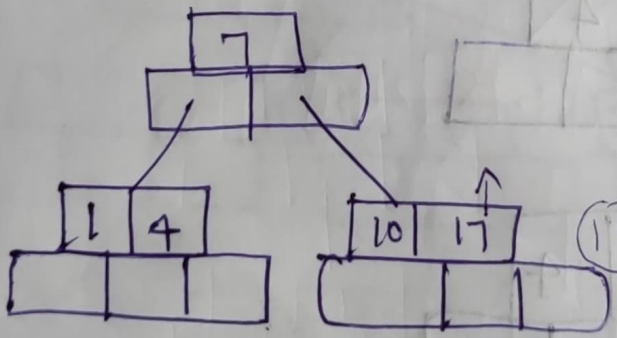


Insert 4.

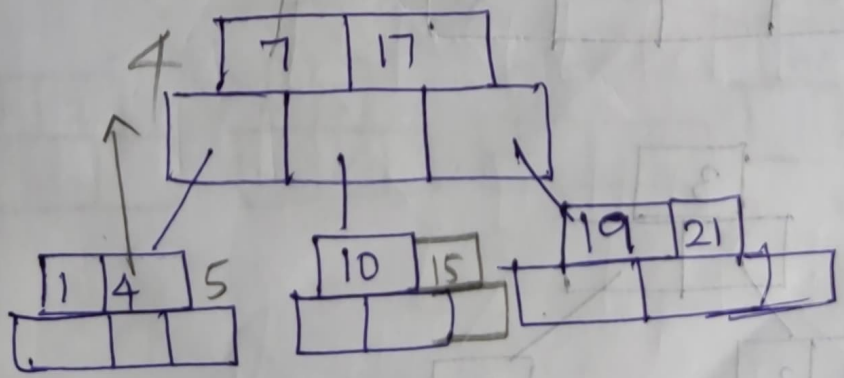
node are constructed from the bottom

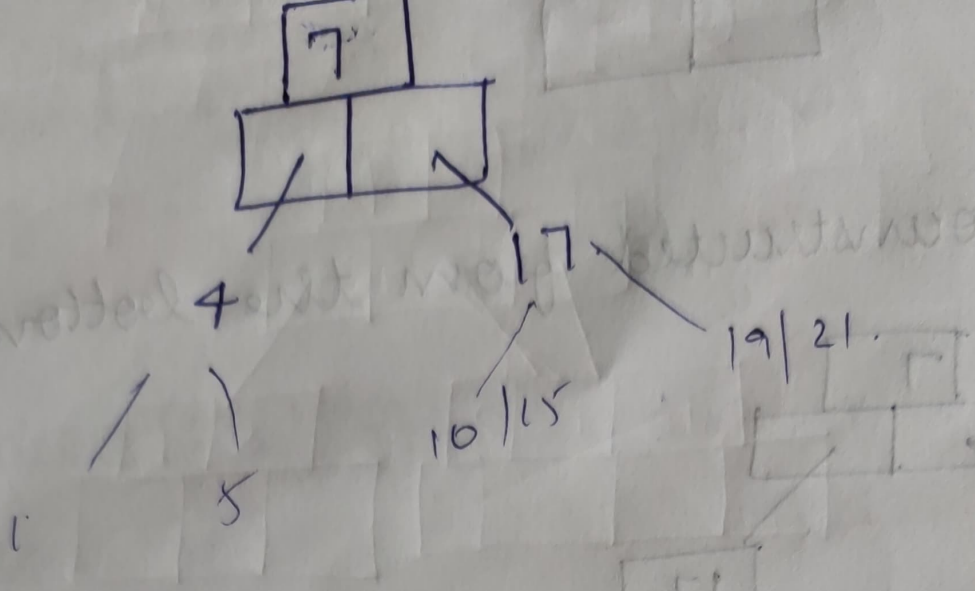


Insert 10.



Insert 19, 21, 15, 5.

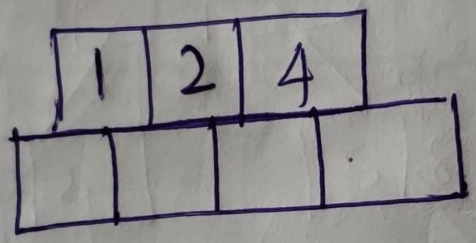




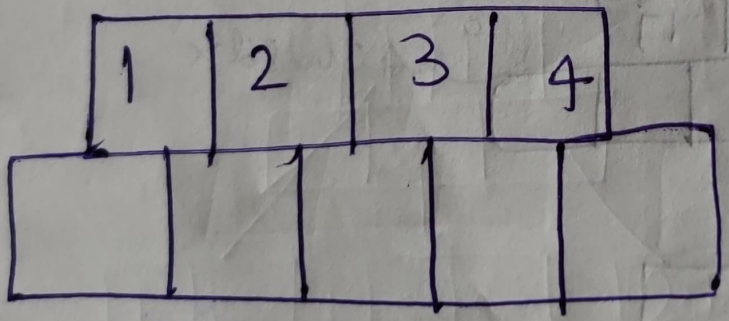
Order 4:

1, 2, 4, 3, 8, 5, 25, 65, 24, 32.

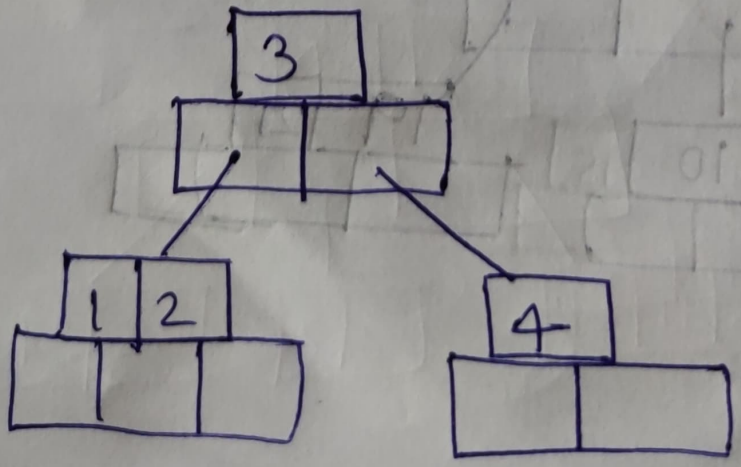
(i)

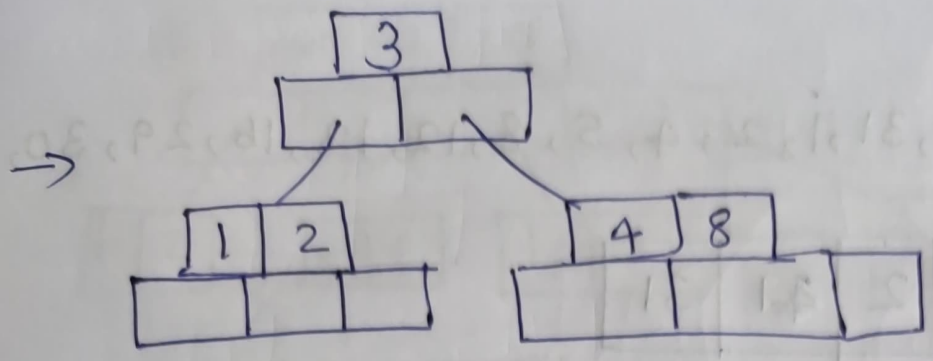


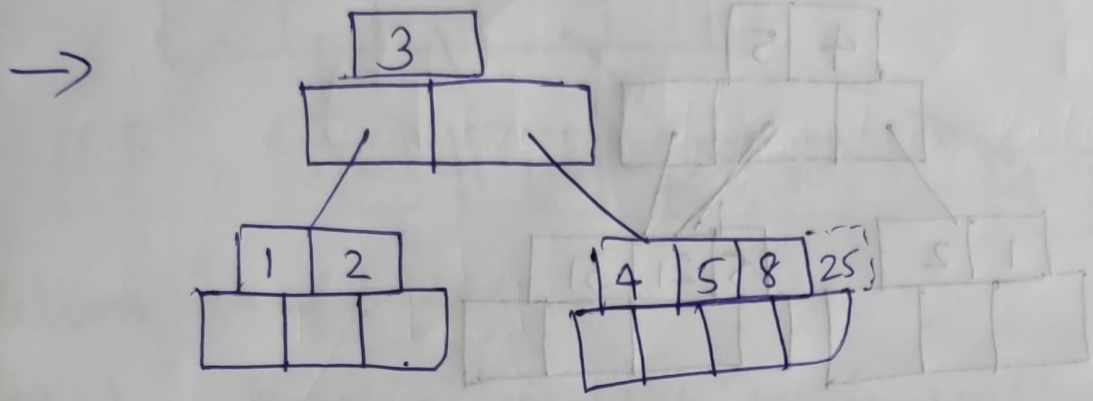
(ii)

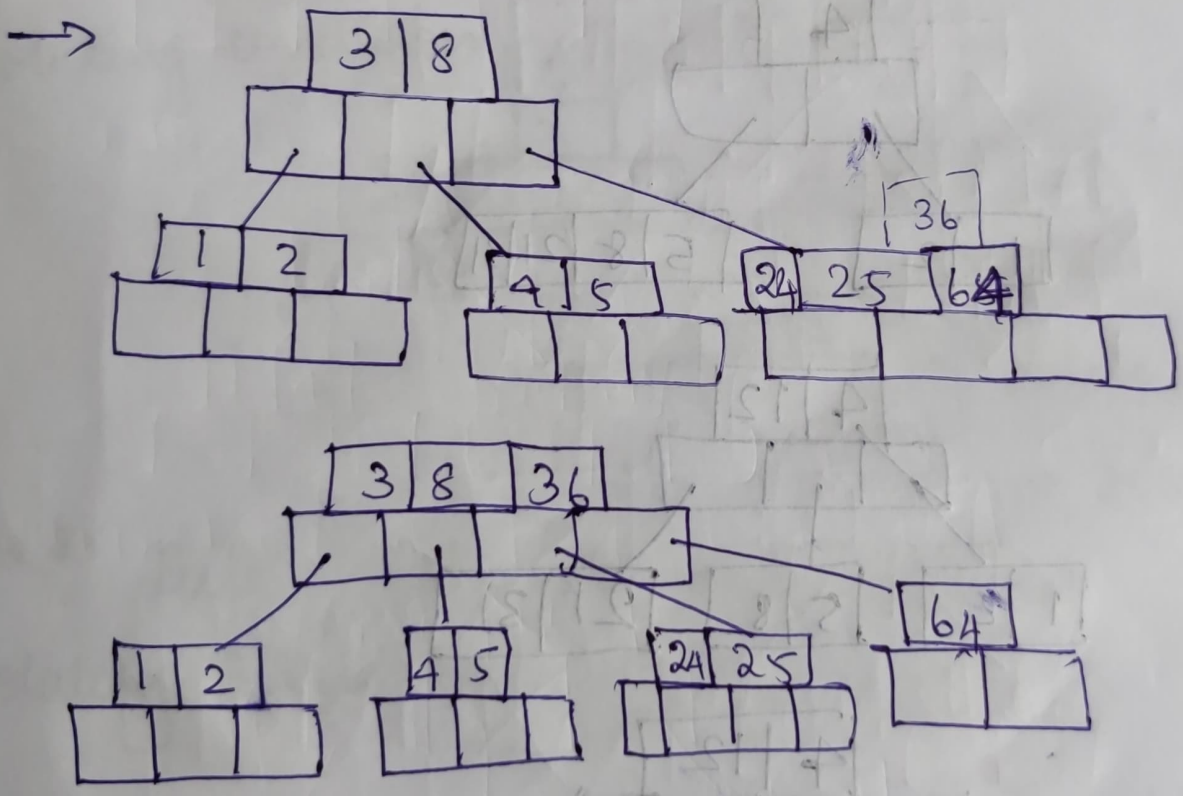


(iii)



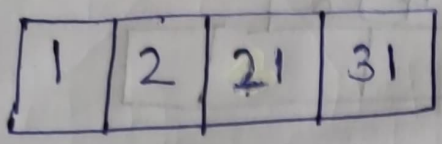

3
1
2
4
8

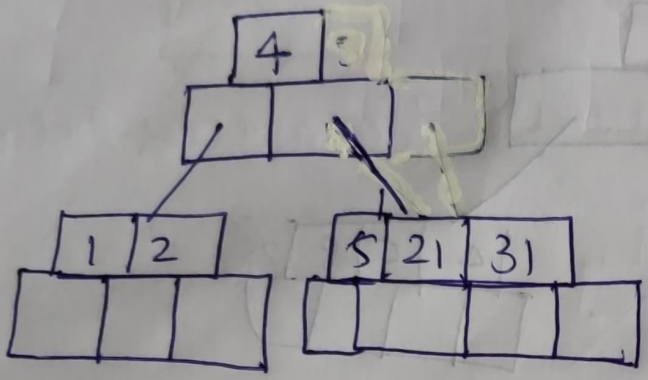

3
1
2
4
5
8
25

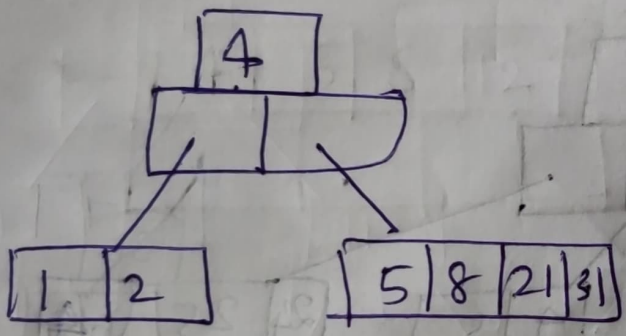

3
8
1
2
4
5
36
24
25
64
3
8
36
1
2
4
5
24
25
64

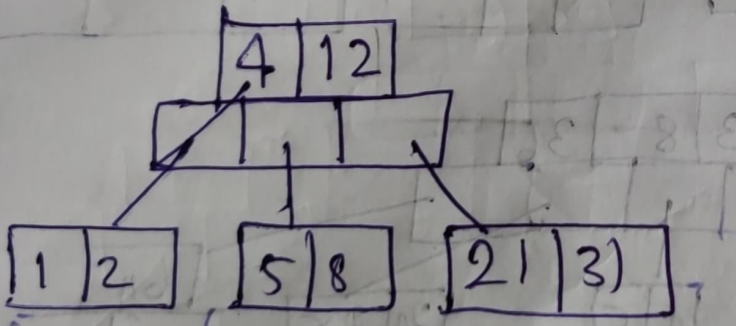
Order 5:

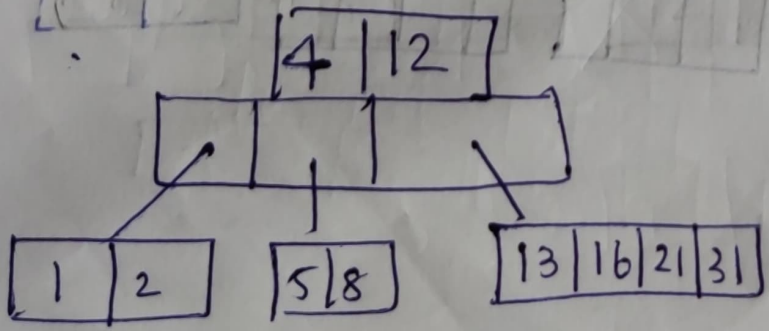
Insert: 21, 31, 1, 2, 4, 5, 8, 12, 13, 16, 29, 30, 4

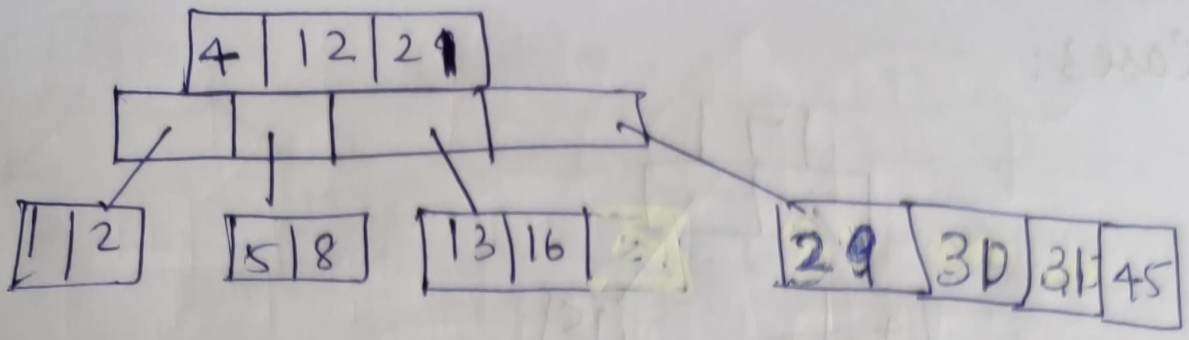


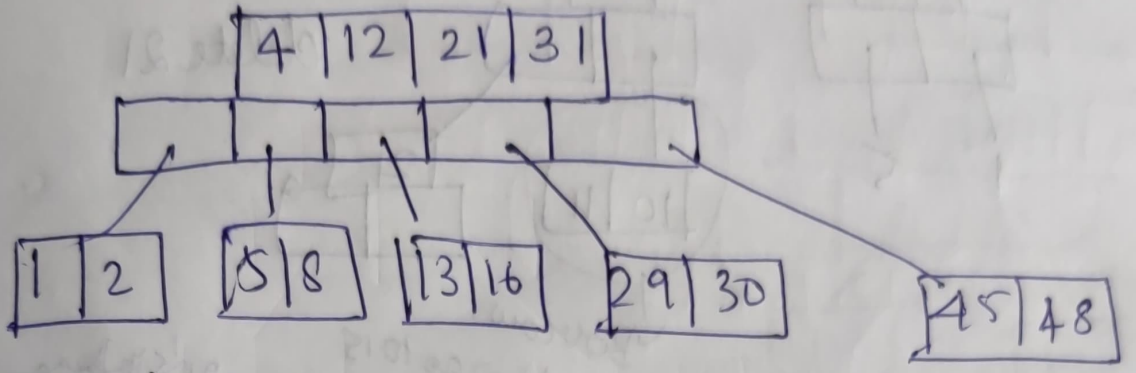






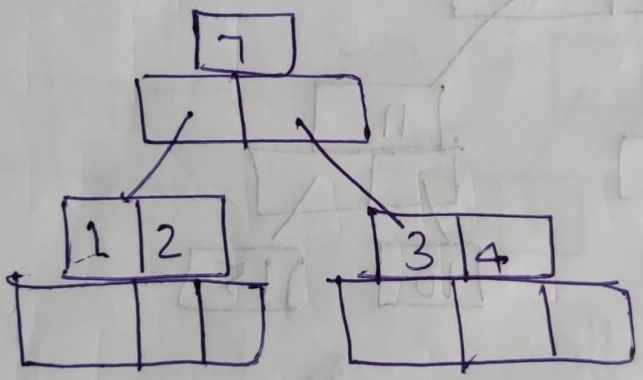






Deletion in a B tree

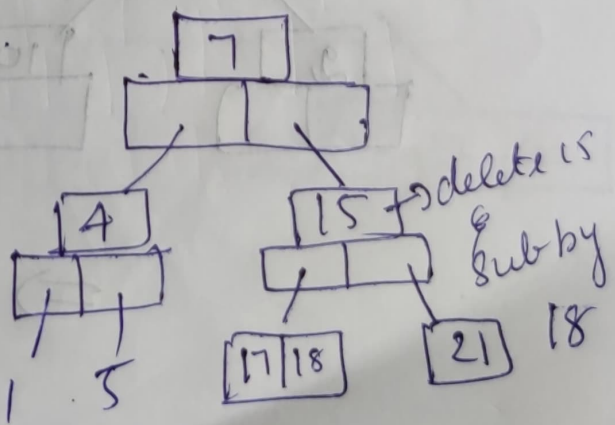
Case: 1 Deletion in leaf node without violating condition.



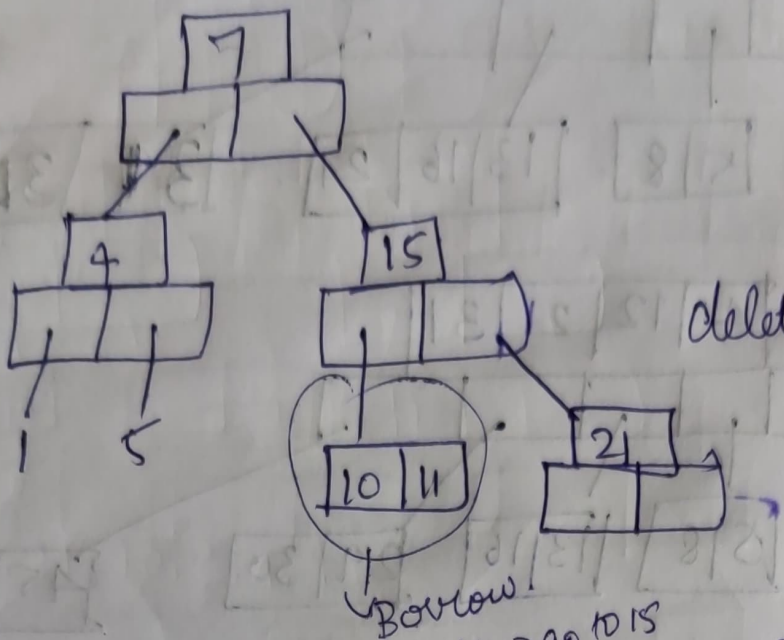
delete 3

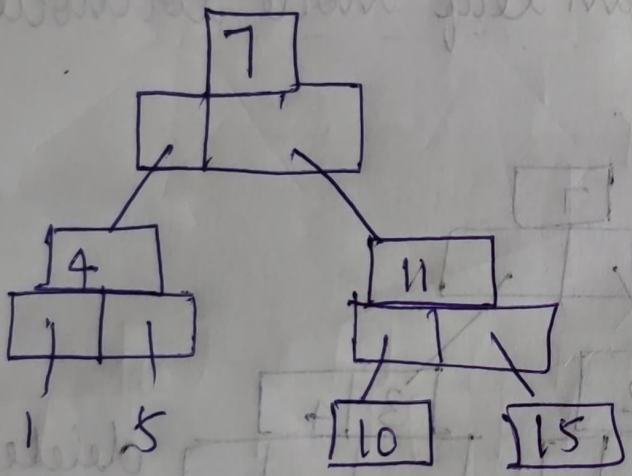
no violation.

Case 2: Delete internal node without violating B tree property.

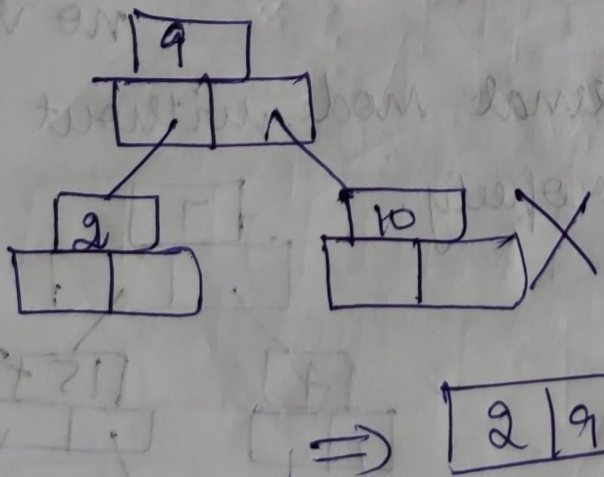


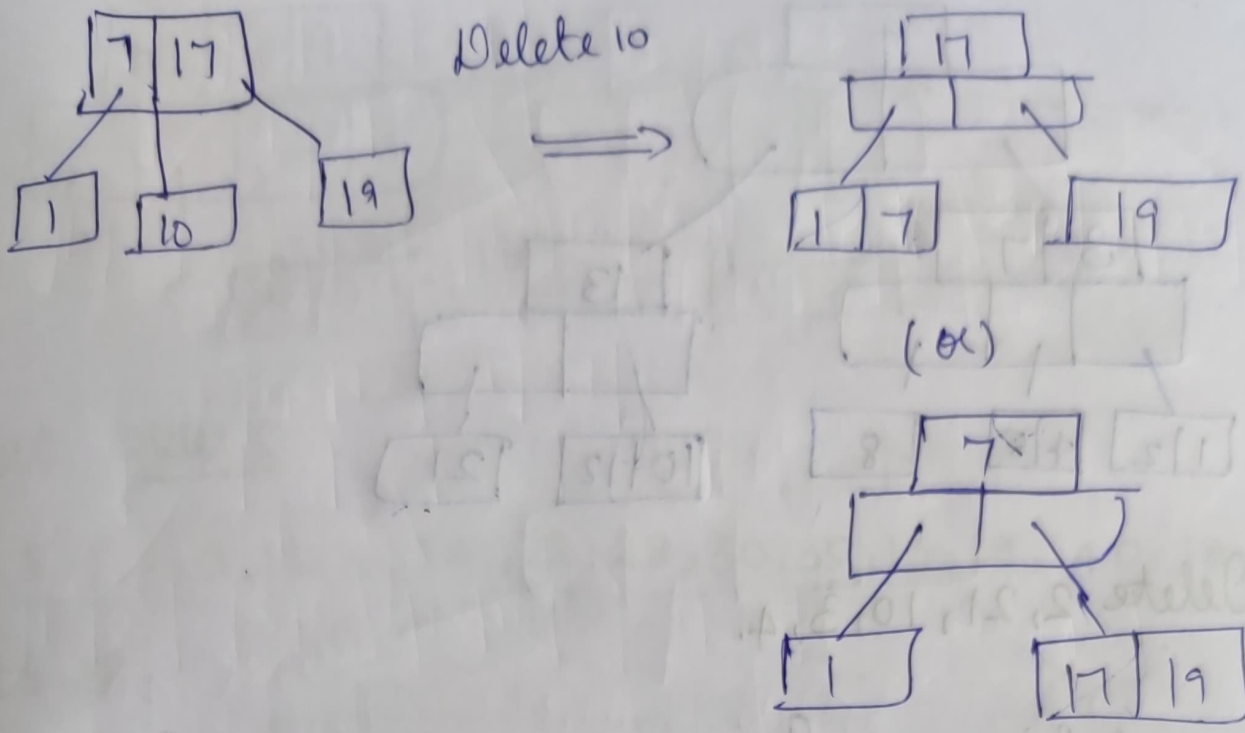
Case3:


7
4
15
1
5
10
11
21
delete 21
Borrow
11 → go to 15


7
4
11
1
5
10
15

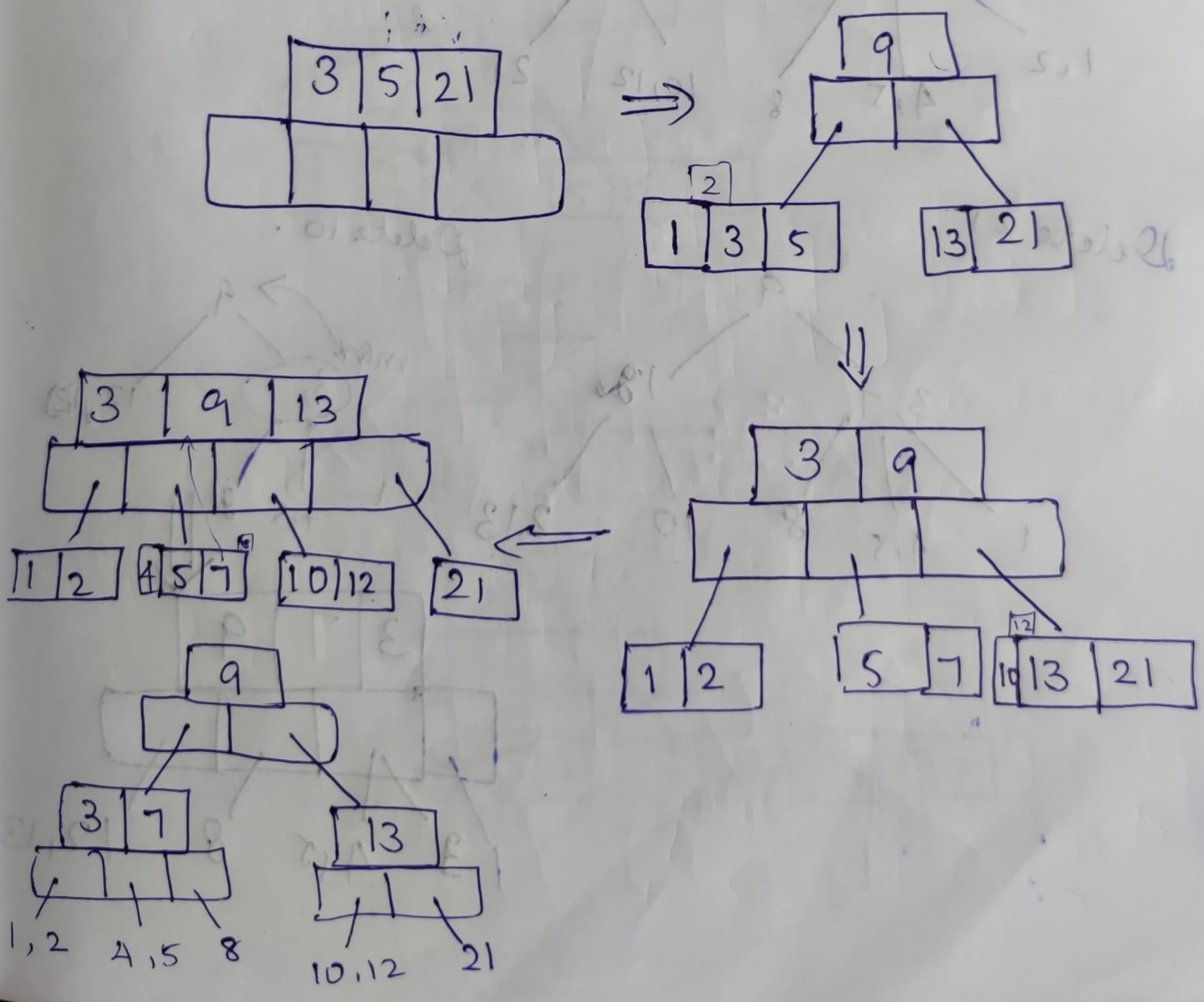
Case4:

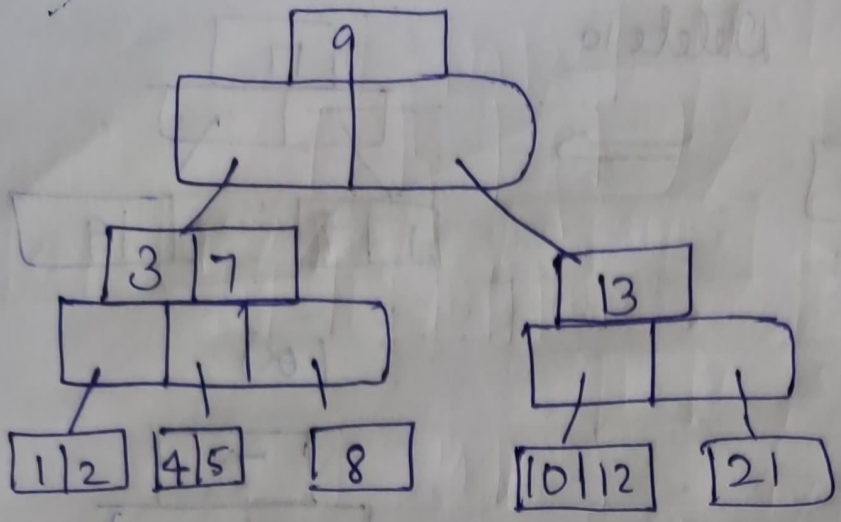

9
2
10
merge left and right with parent.
⇒
2
9



Construct B tree order 4.

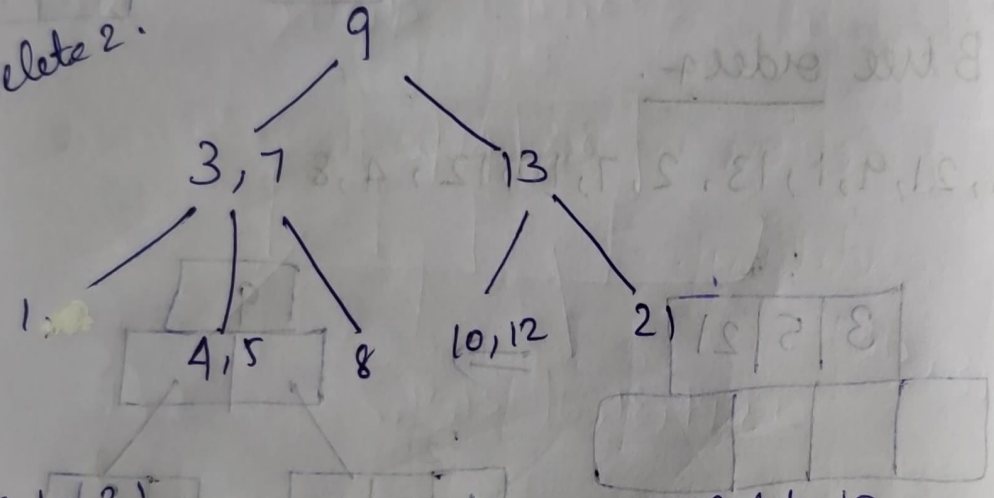
5, 3, 21, 9, 1, 13, 2, 7, 10, 12, 4, 8



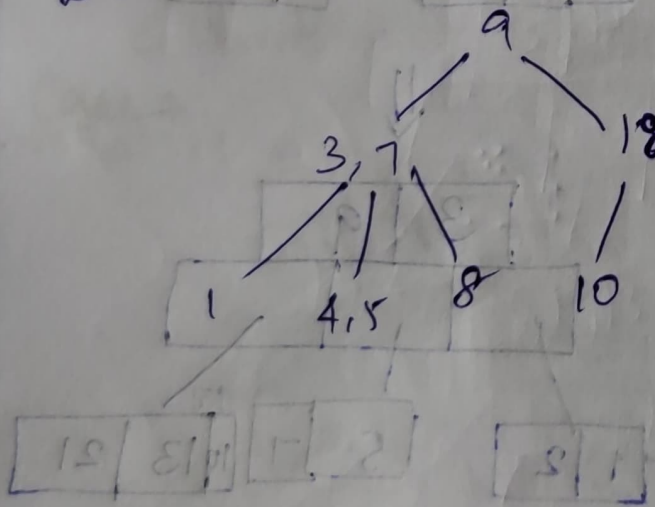


Delete 2, 21, 10, 3, 4.

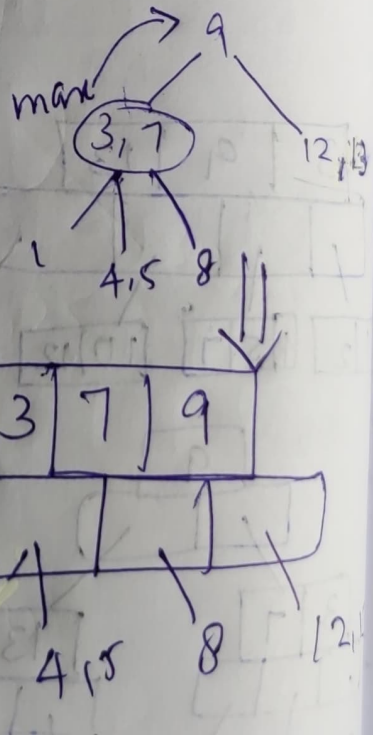
Delete 2.

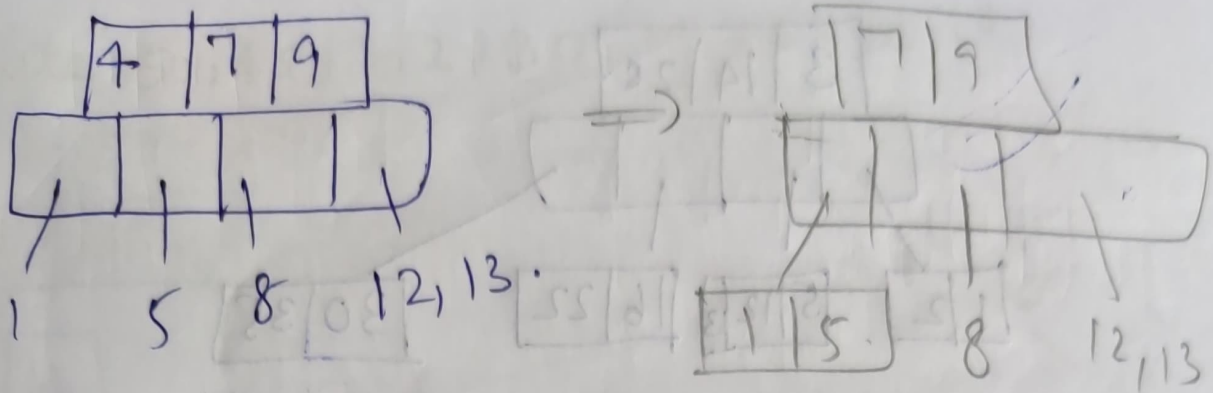


Delete 21



Delete 10.

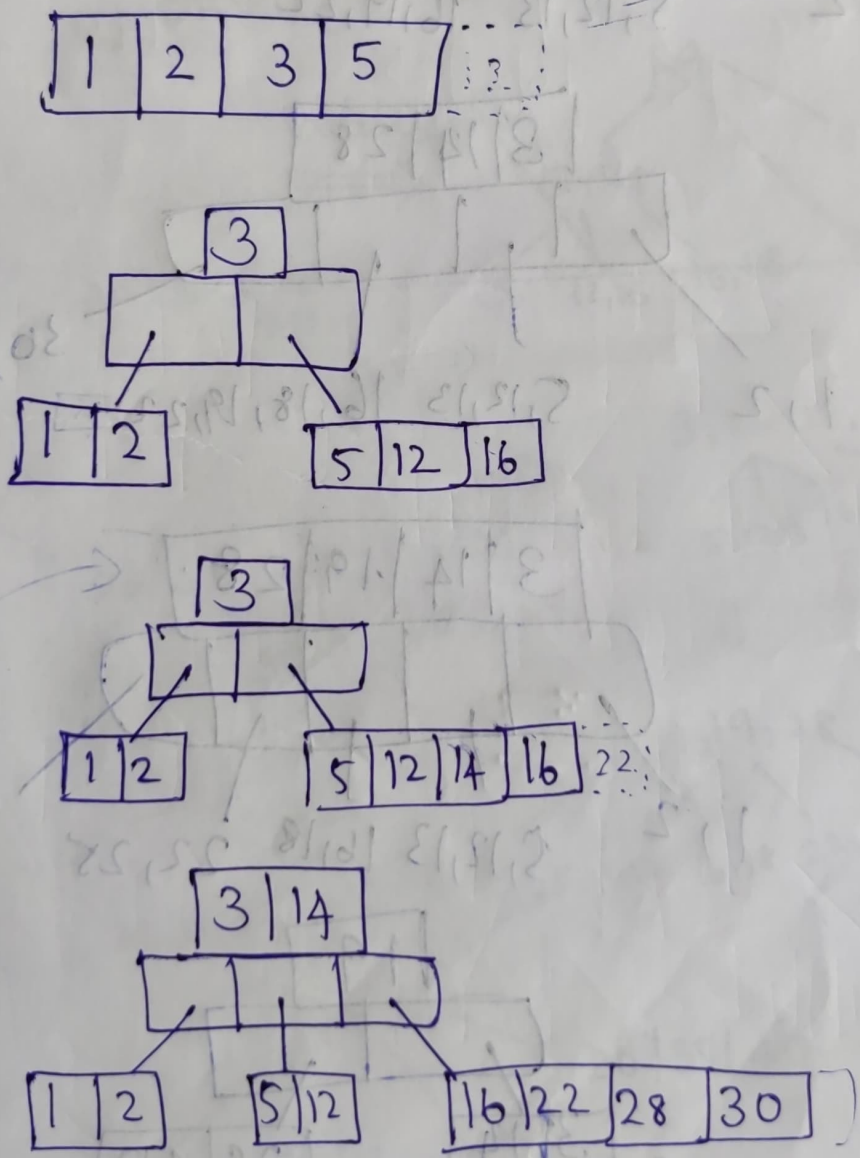


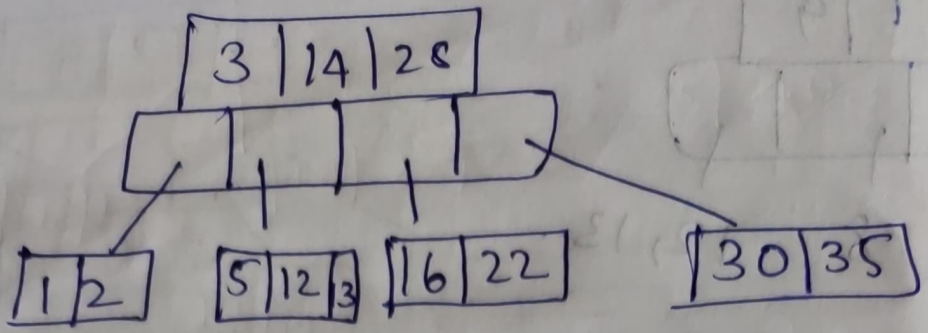


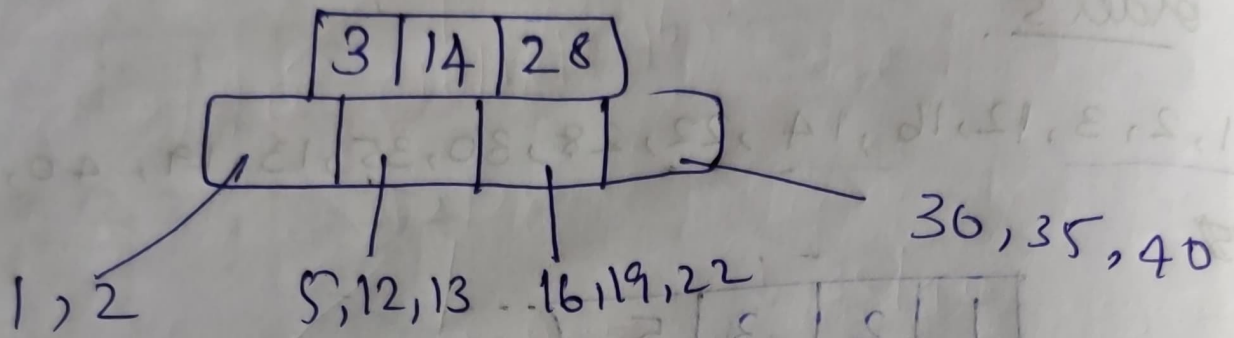
Insert orders.

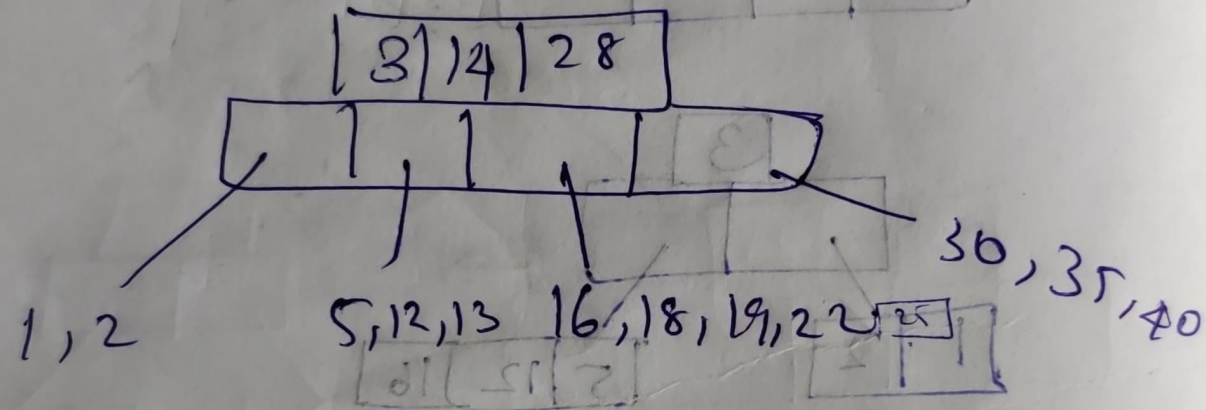
5, 1, 2, 3, 12, 16, 14, 22, 28, 30, 35, 13, 19, 40, 18

Delete 18

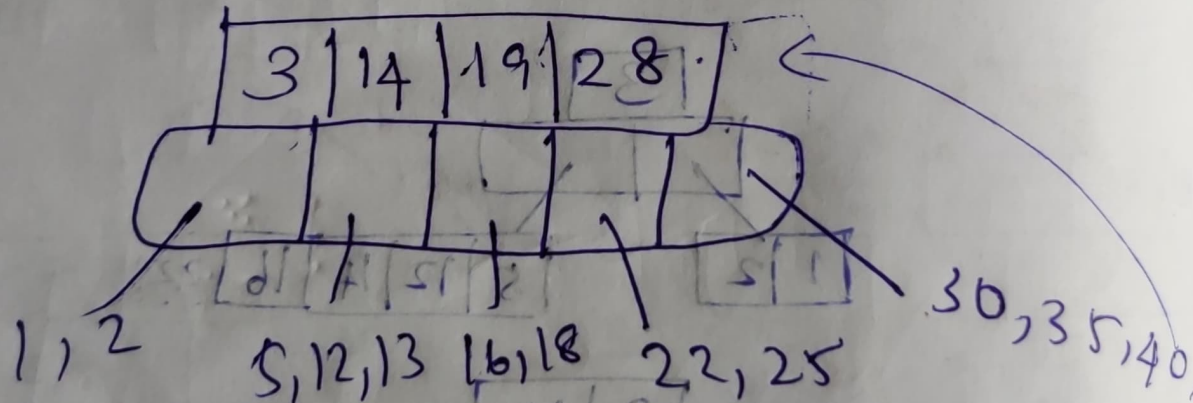


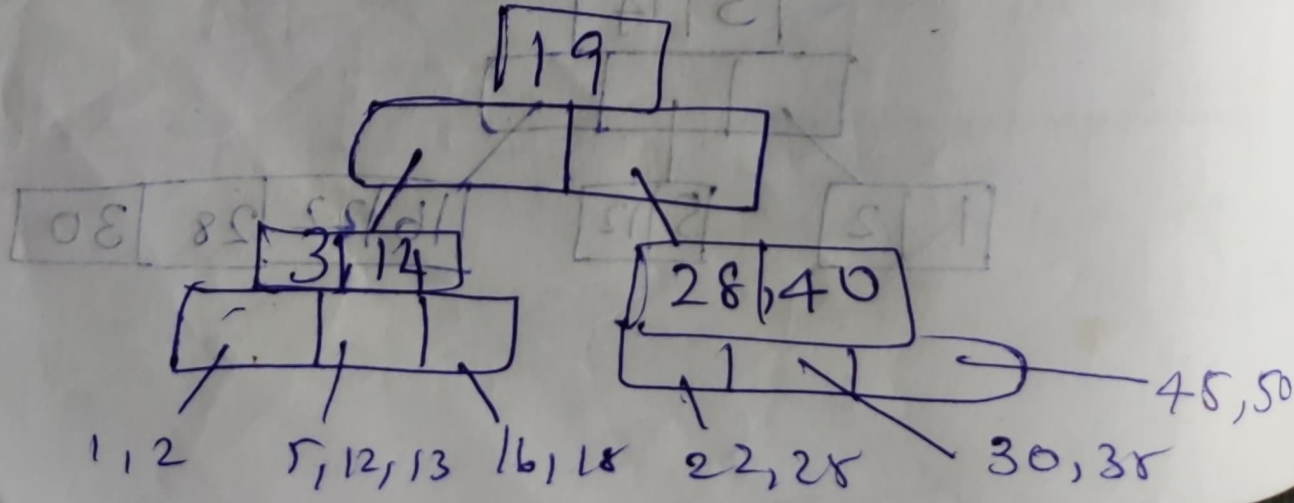

3 14 28
1 2
5 12 13
16 22
30 35


3 14 28
1, 2
5, 12, 13
16, 19, 22
30, 35, 40

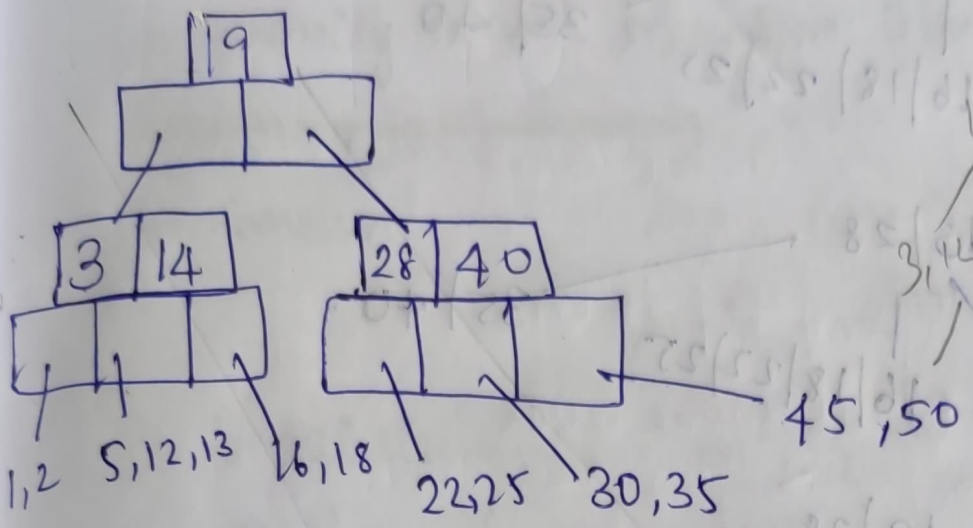

3 14 28
1, 2
5, 12, 13
16, 18, 19, 22
30, 35, 40

45, 50

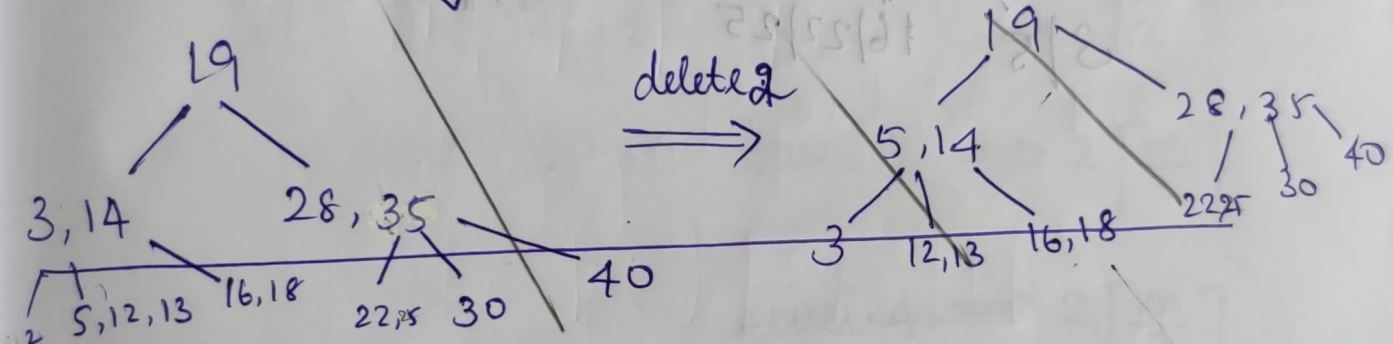

3 14 19 28
1, 2
5, 12, 13
16, 18
22, 25
30, 35, 40

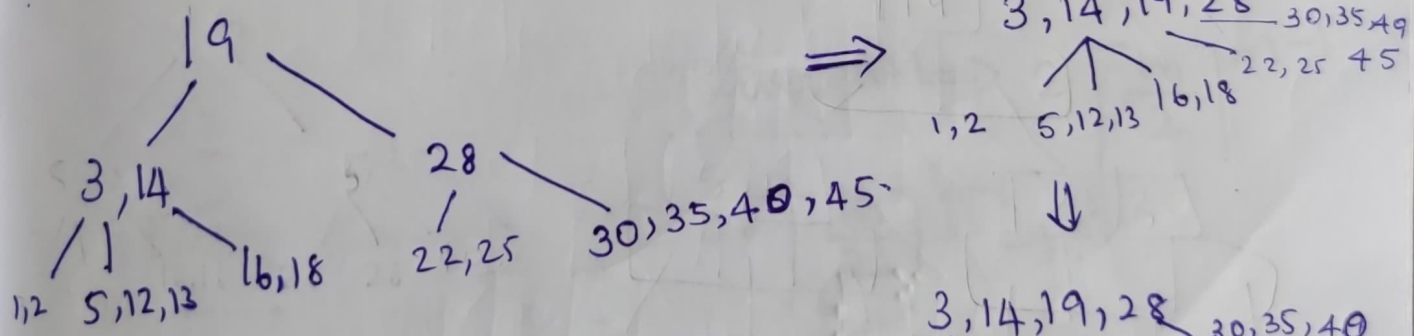

19
3 14
28 40
1, 2
5, 12, 13
16, 18
22, 25
30, 35
45, 50

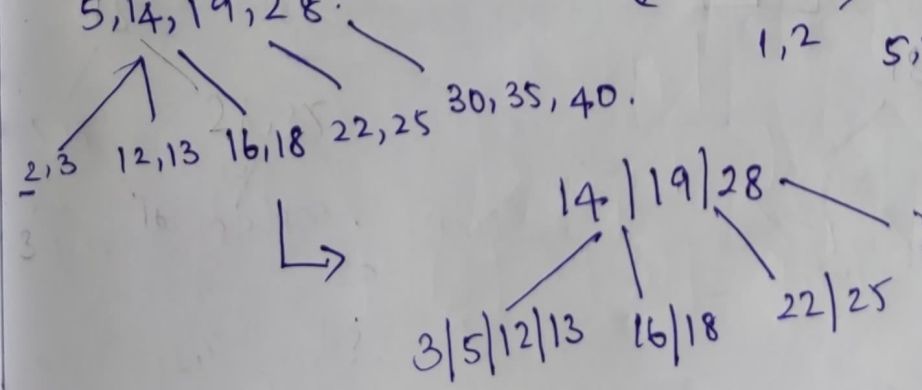
Delete 50, 45, 1, 2, 30, 19, 14, 13, 18

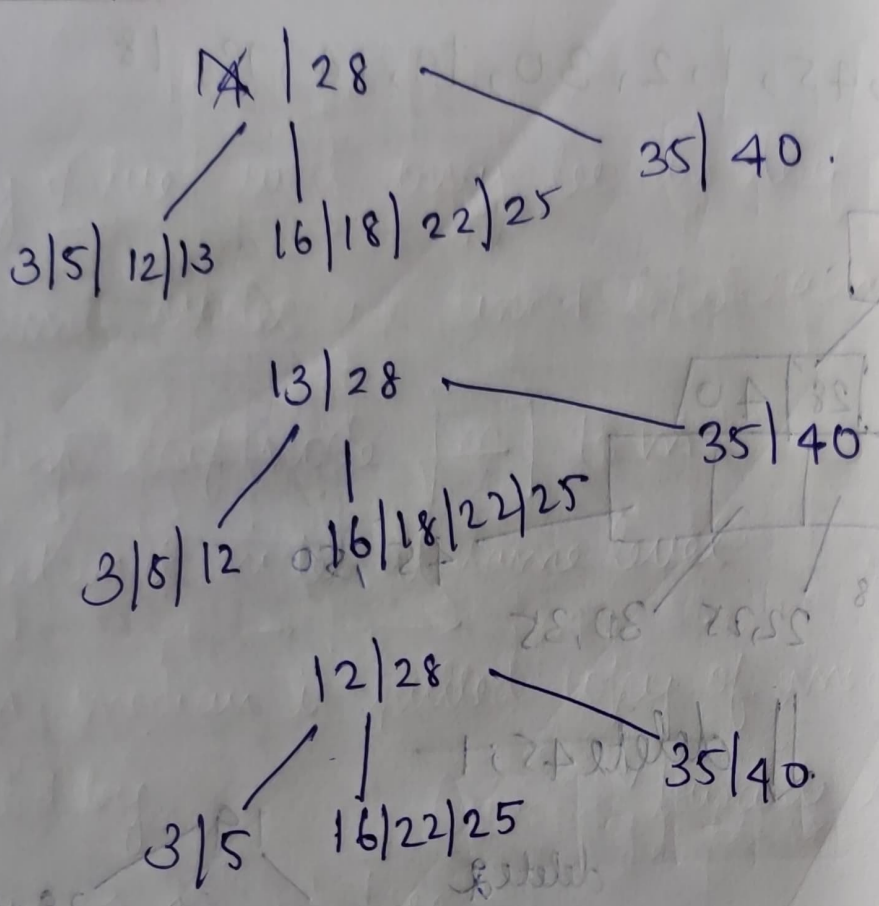


⇓ ~~delete 45, 1~~








28
35|40.
3|5|12|13
16|18|22|25
13|28
35|40.
3|5|12
16|18|22|25
12|28
35|40.
3|5
16|22|25

Analysis of B tree:

Complexity of any operation $O(h)$.

~~Minimum no of elements~~ $= h$.

Maximum " $h = 1$ $(m-1)$ keys.

$h = h \cdot m^{h-1} (m-1)$ keys

Total elements $= m^h - 1$

Minimum no of node. at level 1 $\Rightarrow$ 1

level 2 $\Rightarrow$ 2

level 3 $\Rightarrow 2\left\lceil \frac{m}{2} \right\rceil$

level 4 $\Rightarrow 2\left\lceil \frac{m}{2} \right\rceil^2$

level $h+1 \Rightarrow 2\left\lceil \frac{m}{2} \right\rceil^{h-1}$

No. of elements $n \geq 2 \cdot \left\lceil \frac{m}{2} \right\rceil^{h-1} - 1$.

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

B^+ trees

* key value will only be in leaf node.
* External nodes are linked b/w each other

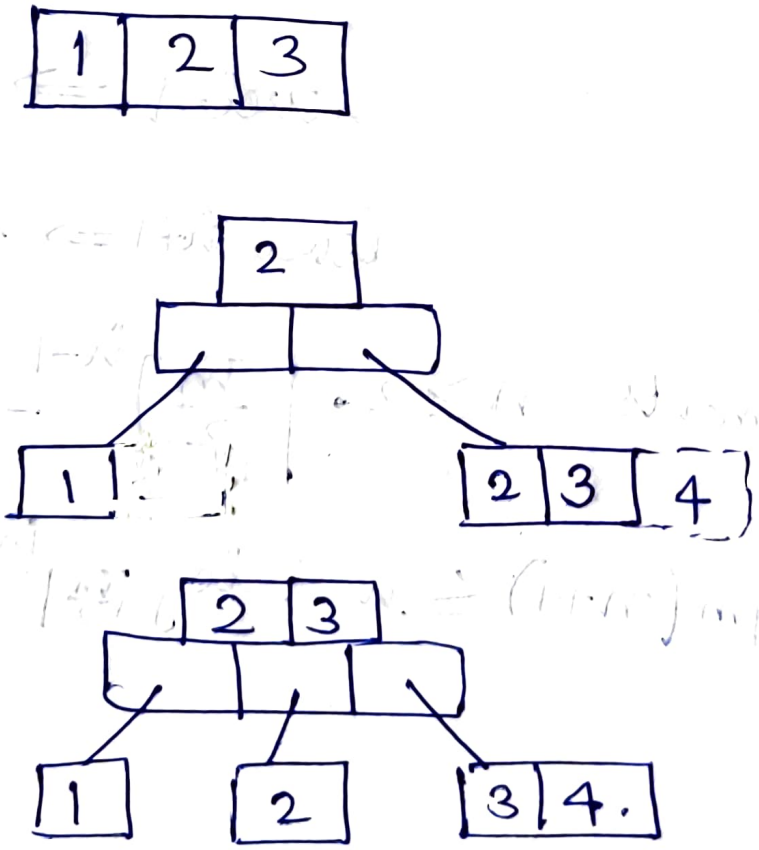
Non leaf node $\lceil \frac{m}{2} \rceil$ to m.

External nodes at same level

When median is pulled copy of median is pulled by. ↓ towards right.

1, 2, 3, 4 B^+ tree.

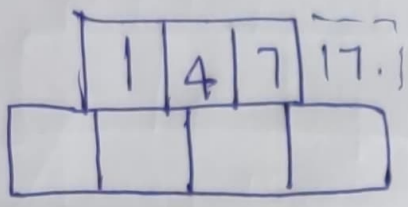
Order = 3.



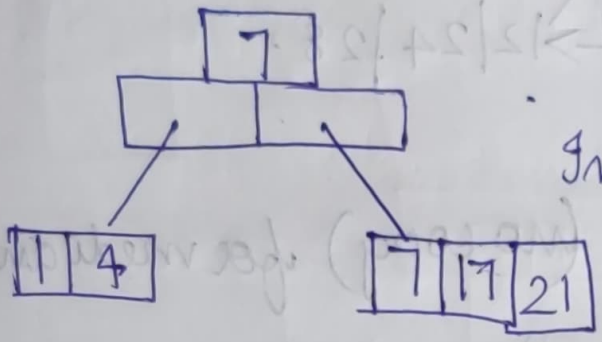
B+ tree

Order 4:

Insert 1, 4, 7

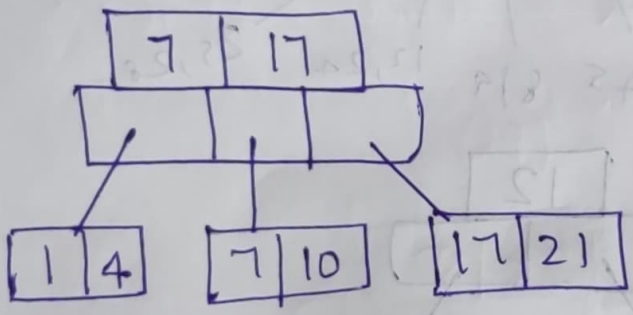


Insert 17



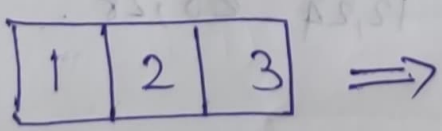
Insert 21

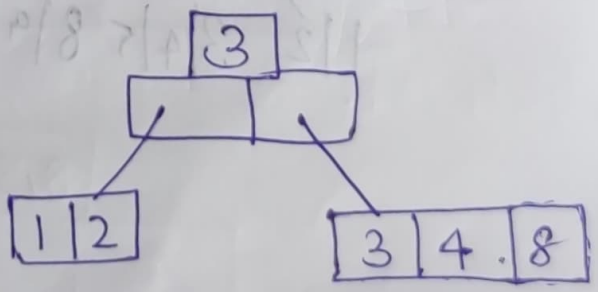
Insert 10.

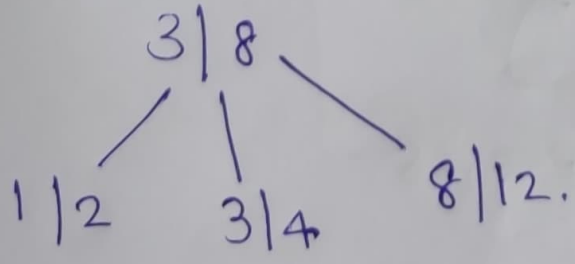


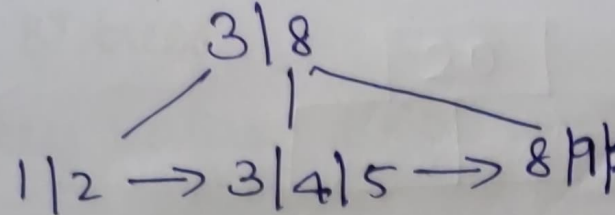
Order 4

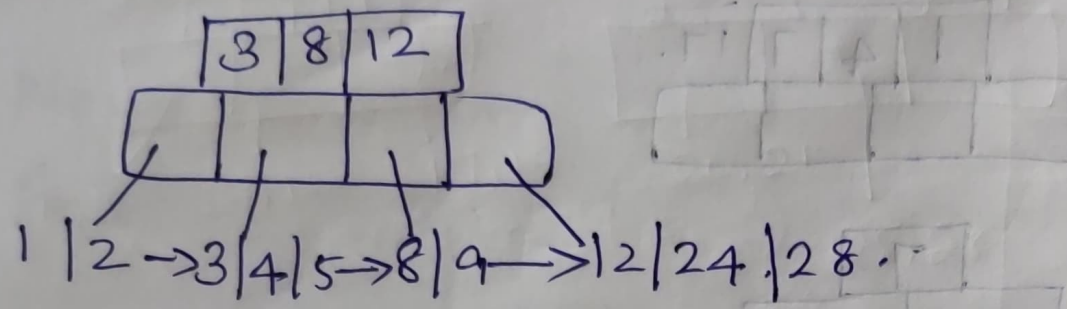
1, 2, 3, 4, 8, 12, 5, 9, 24, 28, 25.





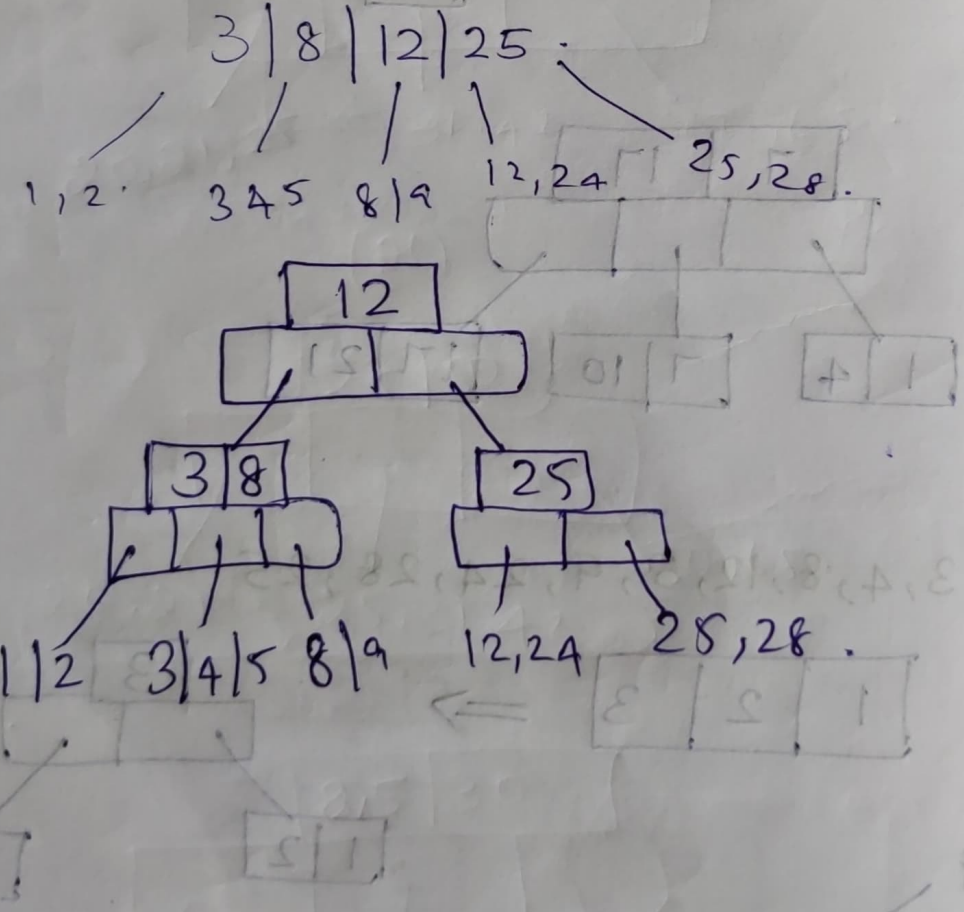




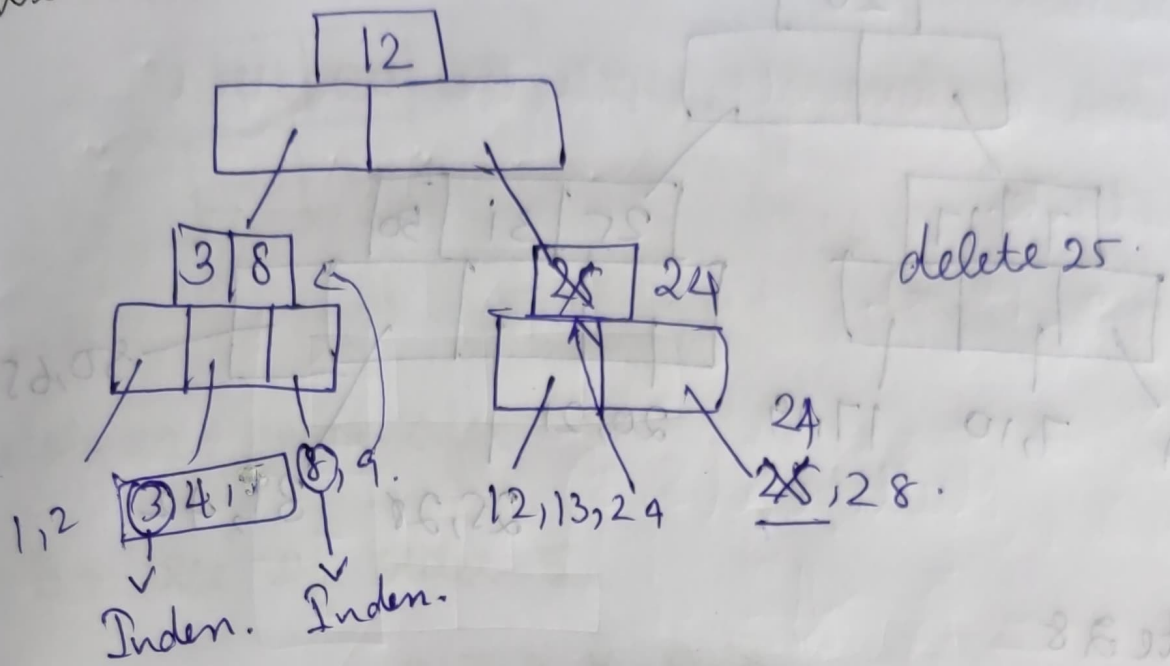


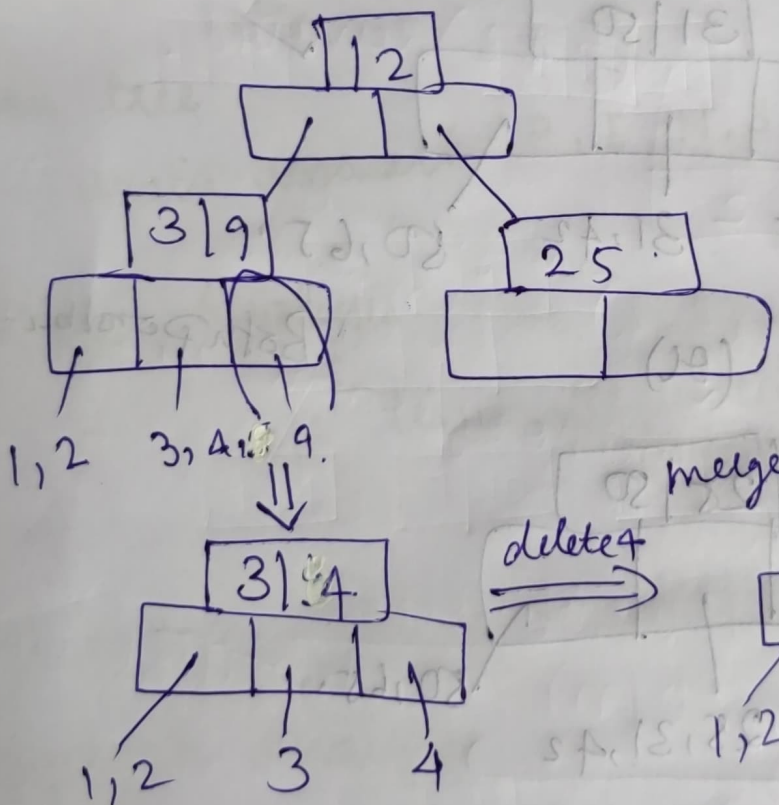
Insert 25.

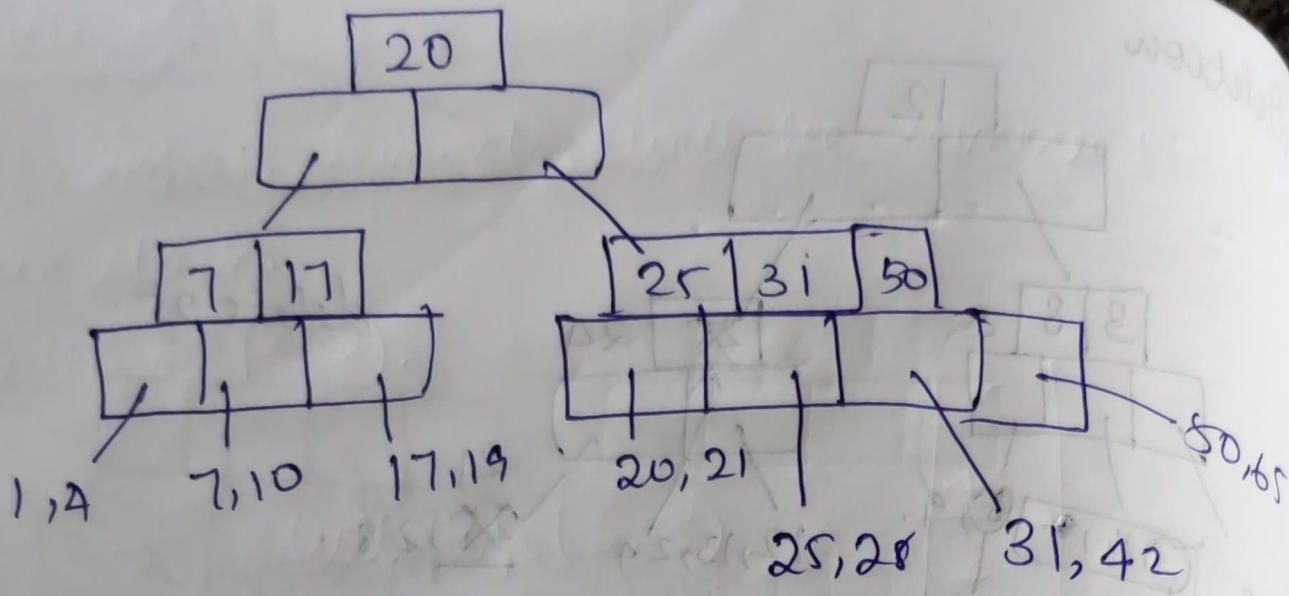
Internal node (no copy) for median



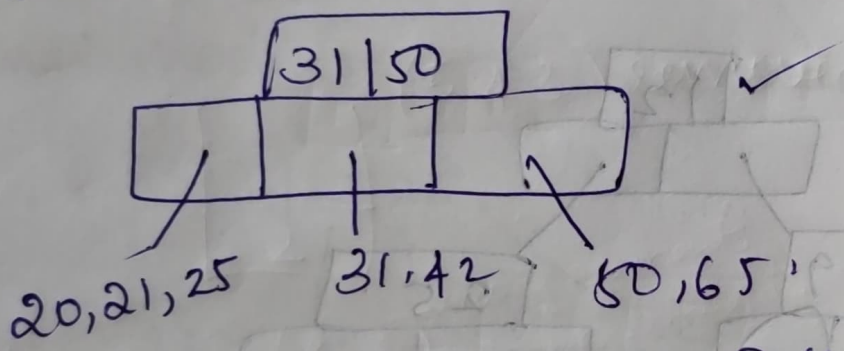
Deletion


12
3 8
25 24
delete 25.
1,2
3 4
8 9.
12,13,24
24
25,28.
Inden.
Inden.


12
3 9
25
1,2
3,4
9.
merge.
3 4
delete 4
3
1,2
3
4
1,2
3

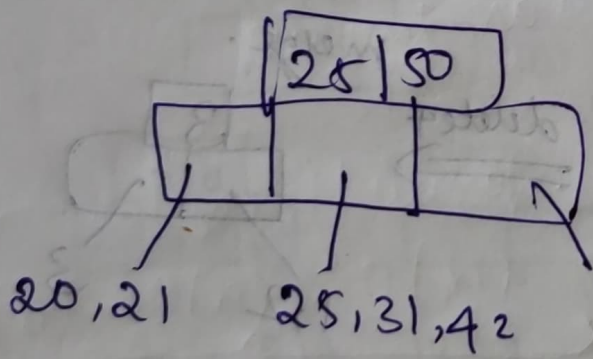


Delete 28

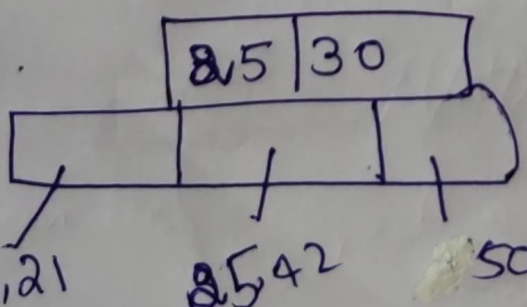


(or)

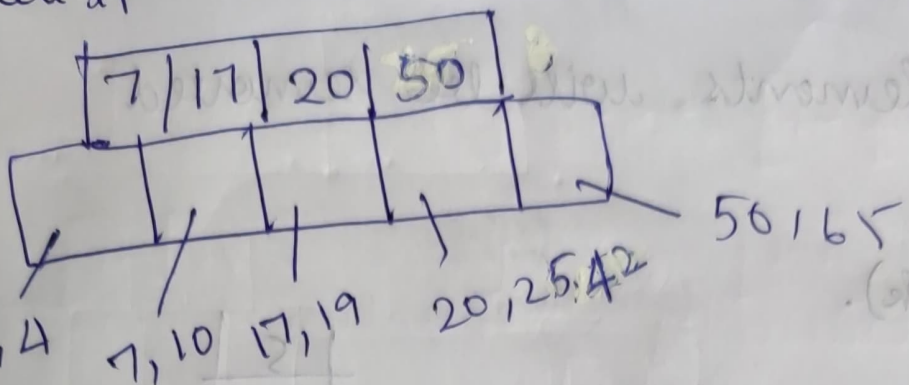
Both possible.



delete 31.



delete 21



delete 42. don't affect.

i) B+ tree of order 5

Q, F, T, B, K, O, X

From tree

again insert P, I, N, S, M, P, O K

Amortized Analysis

Average time required for sequence of operation.

Average cost per operation is small.

i) Aggregate Analysis

sequence of n operation where average time is $\frac{T(n)}{n}$. (Worst case of n operation is T(n))

1. Stack operation

Push (S, x) $\Rightarrow O(1)$, for n operation $\Rightarrow O(n)$

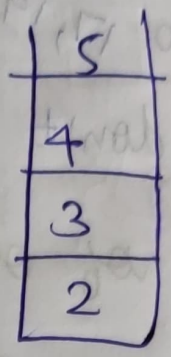
IIly Pop.

Multipop

Top k elements will be removed from stack.

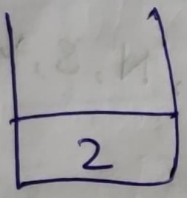
Multipop (S, k).

Multipop (S, 3).



⇓

(5 4 3) removed.



Total cost = min (S, k) → no of elements to be removed.

↓

current no of elements in stack.

n Push, Pop, Multipop.

$\underbrace{O(n) \quad O(n) \quad O(n)}_{O(n)}$

Amortised cost = $\frac{O(n)}{n} = O(1)$

~~$O(n^2) \Rightarrow$ Cost.~~

Each object can be popped only once per time

Binary counter

A[0] ⇒ LSB

0 0 0 0

↓

A[3]

(B bits).

MSB = $\frac{n}{4}$ flip.

LSB = n flip

b/w MSB and LSB (1 bit get flip) = $\frac{n}{2}$ flip.

Cost of increment = Θ(no of bits flipped).

i^{th} bit = $\left\lfloor \frac{n}{2^i} \right\rfloor$ flip.

Total no flips = 2n

n ⇒ # of operation, K bits flip.

Accounting method, Potential method $O(nk)$.

Assign different charges to different operation

* Some operation are more than actual cost
* Some are charged less.

Amortised charge = amount we charge

when,

amortised cost > actual cost.

store the difference on specific object in the data structure as credit.

actual cost > amortized cost.

use credit later to pay for operation.

Let

$c_i \Rightarrow$ actual cost of i^{th} operation.

$\hat{c}_i \Rightarrow$ amortised cost

$$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$$

- The difference of amortized cost and actual cost will always be greater than 0.

Actual cost of Multipop min(K, S).

The actual cost for pushing will be 1, but once an element is pushed it should be poped amortized cost 1+1 = 2

The amortized cost will always be greater than or equal to actual cost, so the credit will never go negative.

If there are n operation then amortized cost would be- $\sum_{i=1}^{n} \hat{c}_i$

Totally n push operations ⇒ so, amortised operations is 2n.

So, amortized cost = O(n).

Binary Counter

Flipping $0 \rightarrow 1 \rightarrow 2$

$1 \rightarrow 0 \rightarrow 1$

When ever the flipping occur $0 \rightarrow 1$ is always greater than $1 \rightarrow 0$. so it will always be positive.

For n operations, amortized cost = O(n)

Divide and Conquer

Divide the problem into small instances then get the solution. then combine the solution of the instances together.

eg: mergesort.

Mergesort

Set of elements divided until u get individual element.

Mergesort (A[0 ..... n-1])

1) copy A[0 ... $\lfloor n/2 \rfloor - 1$] to B[0 ... $\lfloor n/2 \rfloor - 1$]

2) copy A[$\lfloor n/2 \rfloor$ ... n-1] to C[0 ... $\lceil n/2 \rceil - 1$]

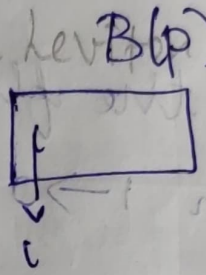
Mergesort (B[0 ... $\lfloor n/2 \rfloor - 1$]

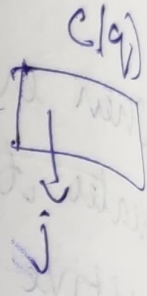
Mergesort (C[0 ... $\lceil n/2 \rceil - 1$]

Merge (B, C, A)

i = 0, j = 0, k = 0

↓ B(1st element) ↓ C ↓ Merged.





while i < p and j < q do

if B[i] ≤ C[j]

A[k] ← B[i] i ← i+1

else A[k] ← C[j] j ← j+1.

k ← k+1.

if i = p

copy C[j ... q-1] to A[k ... p+q+1].

else copy B[i ... p-1] to A[k ... p+q+1]