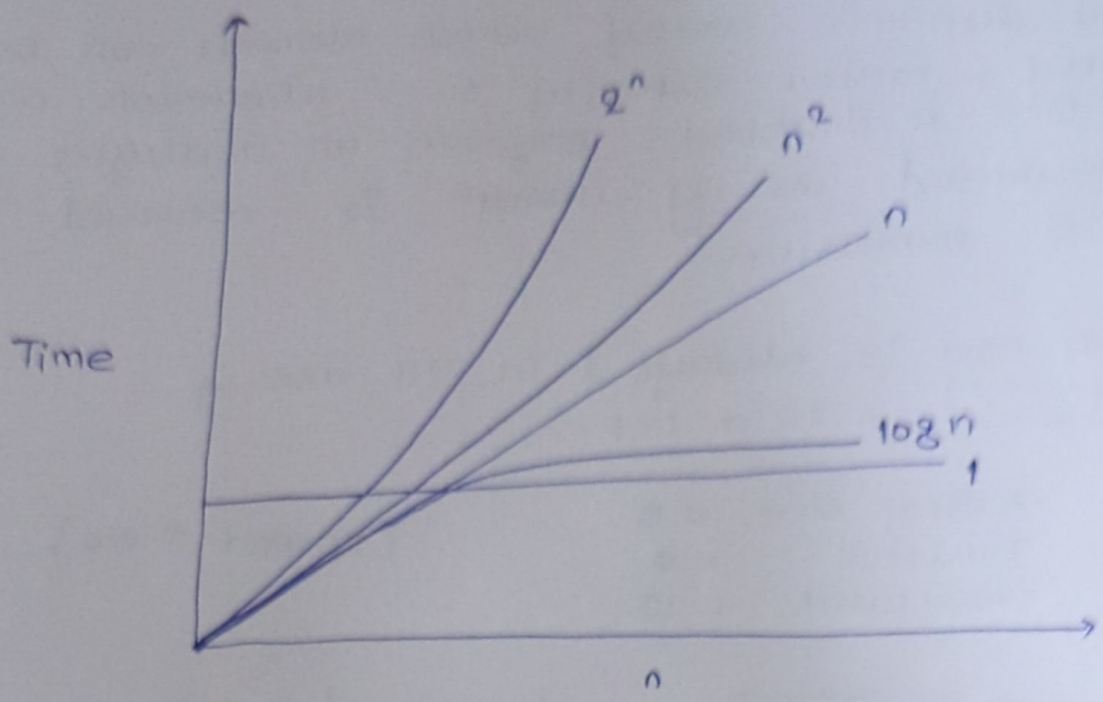


① Asymptotic Not (worst)

$n \geq n_0$

O Θ Ω

big O eqn - ① $f(n) \leq c \cdot g(n)$

$f(n) = O(g(n))$

② $c\,g(n) \leq f(n)$

③ $c_1 g(n) \leq f(n) \leq c_2 g(n)$

② Amortised Analysis (Realistic approach)

eg 499 items = ₹1 each

1 item = ₹500

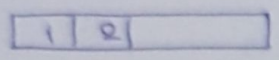
A (worst)	B (Realistic)
₹ 500 × 500 = 250 000	500 × 1 + 1 × 499 = ₹1000

Difinition - Is a method for analysing the time complexity of seq of operat^n which gives more realistic estimat for of performance of DSA.

where some op^n may be expensive but overall cost is balanced.

Implement dynamic array where elements can be added one by one. The array's initial capacity 5. whenever array becomes full its size is doubled. perform an analysis to show that amortised cost of insertn is constant eventhough resizing is expensive.

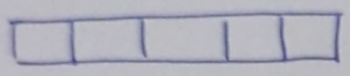
→ Insertion cost is always 1 in an array

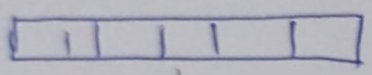


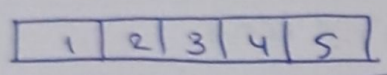
= TC = 1+1 ..

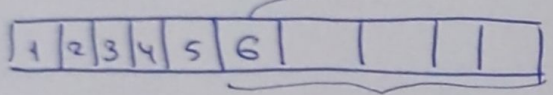
Given · Array size = 5
Total cost = 0
ele-count = 0

(Insert 8 ele)

Step 1 -  tc = 0 | ec = 0

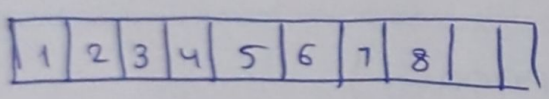
2) -  TC = 1 | EC = 1

5)  TC = 1+1+1+1+1 = 5

6)  TC = 5+5+1 =

doubled size

7)

8)  11+1+1 = 13

(This was Asymptotic)

Now Amortised cost

Amortised cost = $\frac{\text{Total cost}}{\text{No. of opn}} = \frac{13}{8} = \boxed{1.625}$

Avg cost for this = 1.625

Time Analysis of Recursive programs

(1) Back substitution

(2) Recursion Tree

(3) Masters Theorem

(1) Back substitution

$$\begin{cases} T(n) = 1 + T(n-1) & n>0 \\ T(n) = 1 & n=0 \end{cases} \quad \text{---(1)}$$

$T(n-1) = 1 + T(n-2)$ ---(2)

$T(n-2) = 1 + T(n-3)$ ---(3)

(2) in (1)

$T(n) = 1 + [1 + T(n-2)] = 2 + T(n-2)$

$T(n) = 2 + [1 + T(n-3)]$

$\boxed{T(n) = 3 + T(n-3)}$

∴ k times

$T(n) = k + T(n-k)$ for $n-k=0$, $k=n$

∴ $T(n) = n+1$

$= n$

$= O(n)$

Example 2 - 2d array $T(n) = n + T(n-1)$ $n>1$ ---(1)

$T(0) = 1$ $n=1$

$T(n-1) = (n-1) + T(n-2)$ ---(2)

$T(n-2) = (n-2) + T(n-3)$ ---(3)

$T(n) = n + [(n-1) + T(n-2)]$

$= 2n-1 + T(n-2)$

$T(n) = 2n-1 + [n-2 + T(n-3)] = [n + (n-1) + n-2 + T(n-3)]$

$= 3n-3 + T(n-3)$

∴ k times

$$T(n) = n + (n-1) + (n-2) + \cdots + (n-k) + T(n-(k+1))$$

$$n - k + 1 = 1$$

$$\therefore \boxed{k = n-2}$$

$$\therefore n + (n-1) + (n-2) + \cdots + [n-(n-2)] + T(1)$$

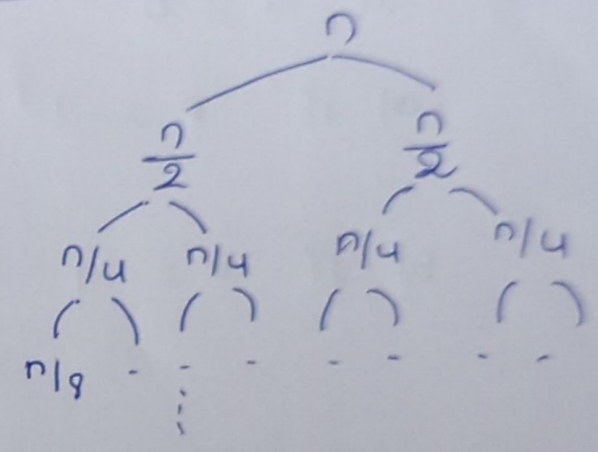
$$n + (n-1) + (n-2) + \cdots + 2 + 1$$

$$\frac{n(n+1)}{2} = O(n^2)$$

Recursive tree:

$$T(n) = 2T(n/2) + n \quad ; \quad n > 1$$

$$T(1) = 1 \quad ; \quad n = 1$$



$$\frac{n}{2^k} = 1$$

$$\therefore n = 2^k$$

$$k = \log_2 n$$

For every level $k = n \log_2 n$

∴ Final Ans: $n \log_2 n$

Master's Theorem:

$$T(n) = a\,T(n/b) + \Theta\left(n^k \log^p n\right)$$

cond^n $a \geq 1$ $\quad b \geq 1$ $\quad k \geq 0$ $\quad$ & p is real

1) If $a > b^k$ then

$$T(n) = \Theta\left(n^{\log_b a}\right)$$

2) if $a = b^k$

a) if $p > -1 \Rightarrow T(n) = \Theta\left(n^{\log_b a} \log_n^{p+1}\right)$

b) if $p = -1$ then $T(n) = \Theta\left(n^{\log_b a} \log\log n\right)$

$$T(n) = \Theta\left(n^{\log_b a}\right)$$

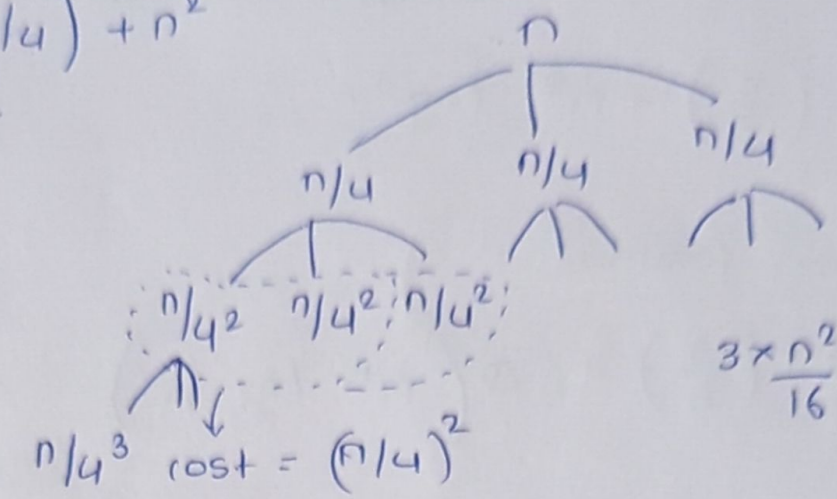
3) if $a < b^k$

if $p \geq 0 \quad T(n) = \Theta\left(n^k \log_n^a\right)$

if $p < 0 \quad T(n) = O(n^k)$

Q. $T(n) = 3T(n/4) + n^2$

$T(1) = 1$



$\Rightarrow n^2$

$3 \times \frac{n^2}{16}$ cost of this level

$$T(n/4^2) = 3T\left(\frac{n}{4^3}\right) + \left(\frac{n}{4}\right)^2$$

1st level - $\frac{n^2}{16^0}$

2nd - $\frac{n^2}{16^1}$

3rd - $\frac{n^2}{16^2}$ --- $\frac{n^2}{16^k}$

$\therefore \frac{n}{4^k} = 1$

$k = \log_4 n$

$\Rightarrow n^2 + \frac{3}{16} n^2 + \frac{9}{16^2} n^2$

$n^2 \left[1 + \frac{3}{16} + \frac{9}{16} \ldots\right]$ ------ This is GP

$$S = \frac{1}{1 - \frac{3}{16}}$$

$$= \frac{16}{13}$$

$$\boxed{n^2 = \frac{16}{13}}$$

Masters Theorem

Q $T(n) = 3T\left(\frac{n}{2}\right) + n^2 \log^0 n$

Masters theorem formal:

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta\left(n^k \log^p n\right)$$

∴ Here,

$a = 3, \quad b = 2, \quad k = 2$

$a = 3 \quad b^k = 4$

∴ $\boxed{a < b^k}$ ------ Now check case to fit.

3rd case:

$T(n) = n^k \log^p n$

$= n^2$

$= O(n^2)$

Q.2 $T(n) = 4T(n/2) + n^2$

$a = 4 \quad b = 2 \quad k = 2 \quad p = 0$

$b^k = 2^2 = 4$

∴ $\boxed{a = b}$

2nd case: (a)

$T(n) = \Theta\left(n^{\log_b^a} \log^{p+1}\right) =$

$= \Theta\left(n^{\log \frac{4}{2}}\right)$

$= \Theta\left(n^2 \log n\right)$

21/1/25

Q. 1) $T(n) = 16T\left(\frac{n}{4}\right) + n$

$a = 16 \quad b = 4 \quad k = 1 \quad p = 0$

$b^k = 4^1 = 4$

$\therefore \boxed{a > b^k}$

$\therefore$ cond^n 1

$T(n) = \Theta\left(n^{\log_b^a}\right)$

$T(n) = \Theta\left(n^{\log_4^{16}}\right)$;?

$T(n) = \Theta(n^2)$ ←

$a > b^k \quad T(n) = \Theta(n^{\log_b a})$

$\Theta(n^2)$

Q. 2) $T(n) = 2T\left(\frac{n}{2}\right) + n\log n$

$a = 2 \quad b = 2 \quad k = 1 \quad p = 1$

$\therefore b^k = 2^1 = 2$

$\therefore \boxed{a = b^k}$

2nd cond^n (a) $p > -1$

$T(n) = \Theta\left(n^{\log_b^a} \log^{p+1} n\right)$

$= \Theta\left(n^{\log_2^2} \log^2 n\right)$

$\boxed{T(n) = \Theta(n \log^2 n)}$

$a = 2 \quad b = 2 \quad k = 1 \quad p = 1$

$a = b^k$

$T(n) = \Theta(n \log^2 n)$

Q. 3) $T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$

$a = 2 \quad b = 2 \quad k = 1 \quad p = -1$

$\therefore b^k = 2$

$\therefore \boxed{a = b}$

case 2 (b)

$T(n) = \Theta\left(n^{\log_a^b} \log\log n\right)$

$= \Theta\left(n^{\log_2^2} \log\log n\right)$

$= \boxed{\Theta(n \log\log n)}$

$a = 2 \quad b = 2 \quad n = 1 \quad p = -1$

$T(n) = \Theta(n \log\log n)$

$T(n) = \Theta\left(n \log_2^2\right)$

$T(n) = \Theta(n)$

Q.4) $T(n) = 2T\left(\frac{n}{4}\right) + n^{0.51}$

$a=2 \quad b=4 \quad k=0.51 \quad p=0$

$b^k = 4^{0.51} = 2$

$\therefore$ **$a=b$** case 2(a) ✗ wrong

$T(n) = \Theta(n^{\log_a b} \log_n^{p+1})$

$= \Theta(n^{\log_2^4} \log_n^{p+1})$

$= \Theta(n^2 \log n)$

$a < b^k$

$T(n) = \Theta(n^k \log_n^a)$

$= \Theta(n^{0.51} \log_n^2)$

$= \Theta(n^{0.51})$

$\Theta(n^{0.51})$

Q.5) $T(n) = 0.5T(n/2) + 1/n$

→ $a=0.5 \quad b=2 \quad k=-1 \quad p=0$

$b^k = 2^{-1} = 0.5$

$a=b$

$\therefore T(n) = \Theta(n^{\log_b} \log_n^{p+1})$

$= \Theta(n^{\log_{0.5}^{0.5}} \log n)$

$= \Theta(n \log n)$

Not applicable ? How

{ $K \geq 0$ for Masters, Here not }

Q.6 $T(n) = 6T(n/3) + n^2 \log n$

→ $a=6 \quad b=3 \quad k=2 \quad p=1$

$b^k = 3^2 = 9$

$a < b^k$ case 3 (a)

$T(n) = \Theta(n^k \log_n^p)$

$= \Theta(n^2 \log_n^6)$

$= \boxed{\Theta(n^2 \log n)}$

$a=6 \quad b=3 \quad k=2 \quad p=1$

$a < b^k$

$T(n) = \Theta(n^k \log^p n)$

$= \Theta(n^2 \log n)$

Q.7 $T(n) = 64T(n/8) - (n^2 \log n)$

$a = 64 \quad b = 8 \quad k =$

Not Applicable why?

Q.8 $T(n) = \sqrt{2}\, T(n/2) + \log n$

$a = \sqrt{2} \quad b = 2 \quad k = 0 \quad P = 1$

$b^k = 2^0 = 1 \qquad a > b^k$

$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_{\sqrt{2}} 2})$

$\Theta(\sqrt{n})$

$a = 2^{1/2} \quad b = 2 \quad k = 0 \quad P = 1$

$a > b^k$

$T(n) = \Theta(n^{\log_b a})$

$= n^{\log_2 2^{1/2}}$

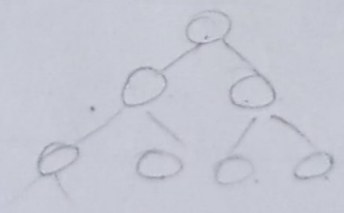
$= \sqrt{n}$

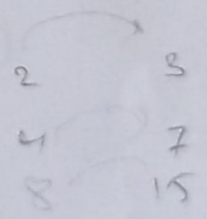
Tree Revisions

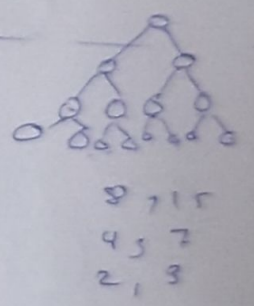
Q In BT 20 leaf nodes, How many total node?

→ $(2 \times \text{leaf nodes}) - 1$

$(2 \times 20) - 1 = 39$







$(2 + \text{leaf}) - 1$

Unit 2 - Search Trees

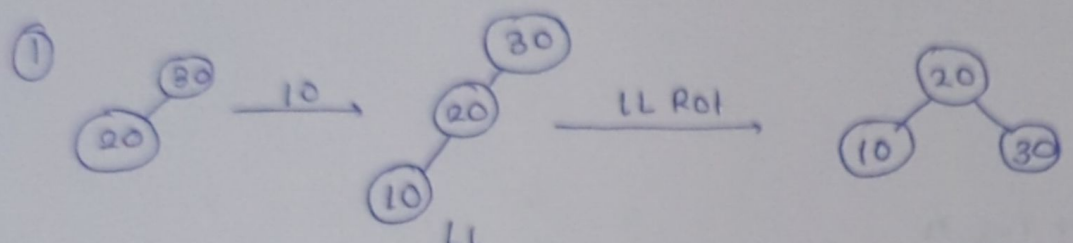
— Why trees?
organised
logarithmic time

① AVL

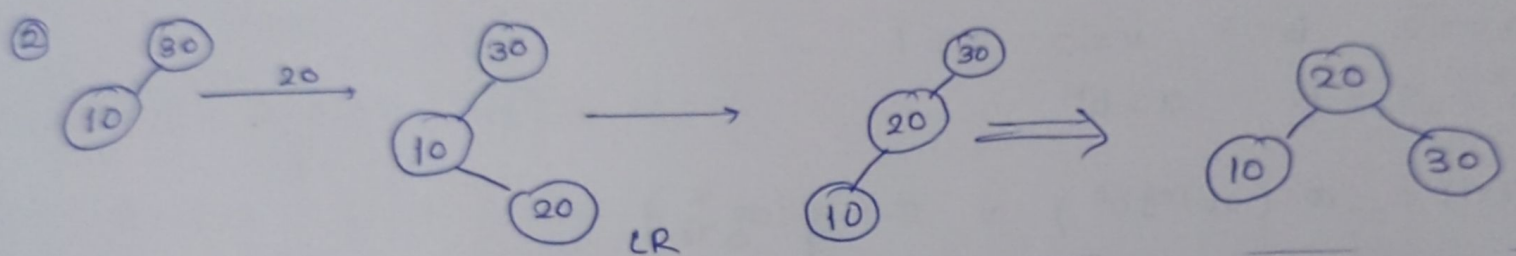
Balance factor = $|H_L - H_R| \leq 1$
Ht of left & right children of every node is differed by ± 1

* Insertion

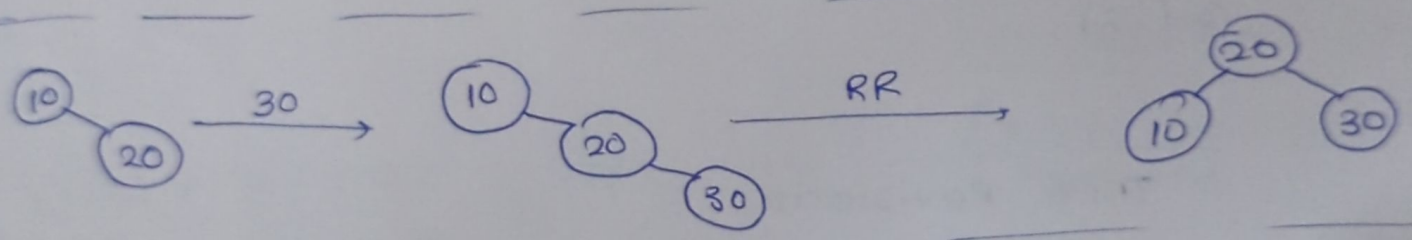
① LL



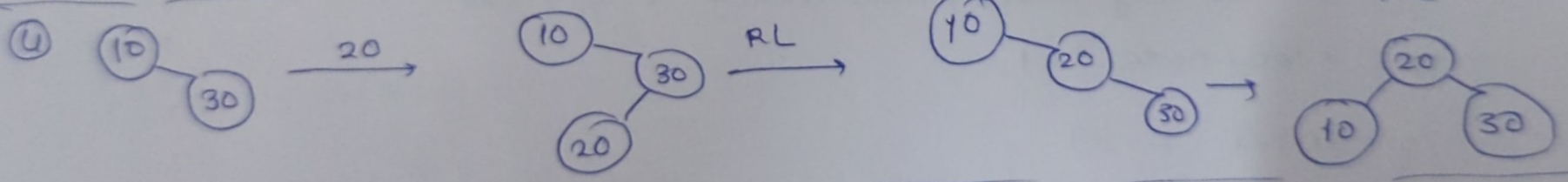
② LR



③ RR



④ RL



27 Jan 2025

② Red Black Tree

colours are req to maintain self balance property

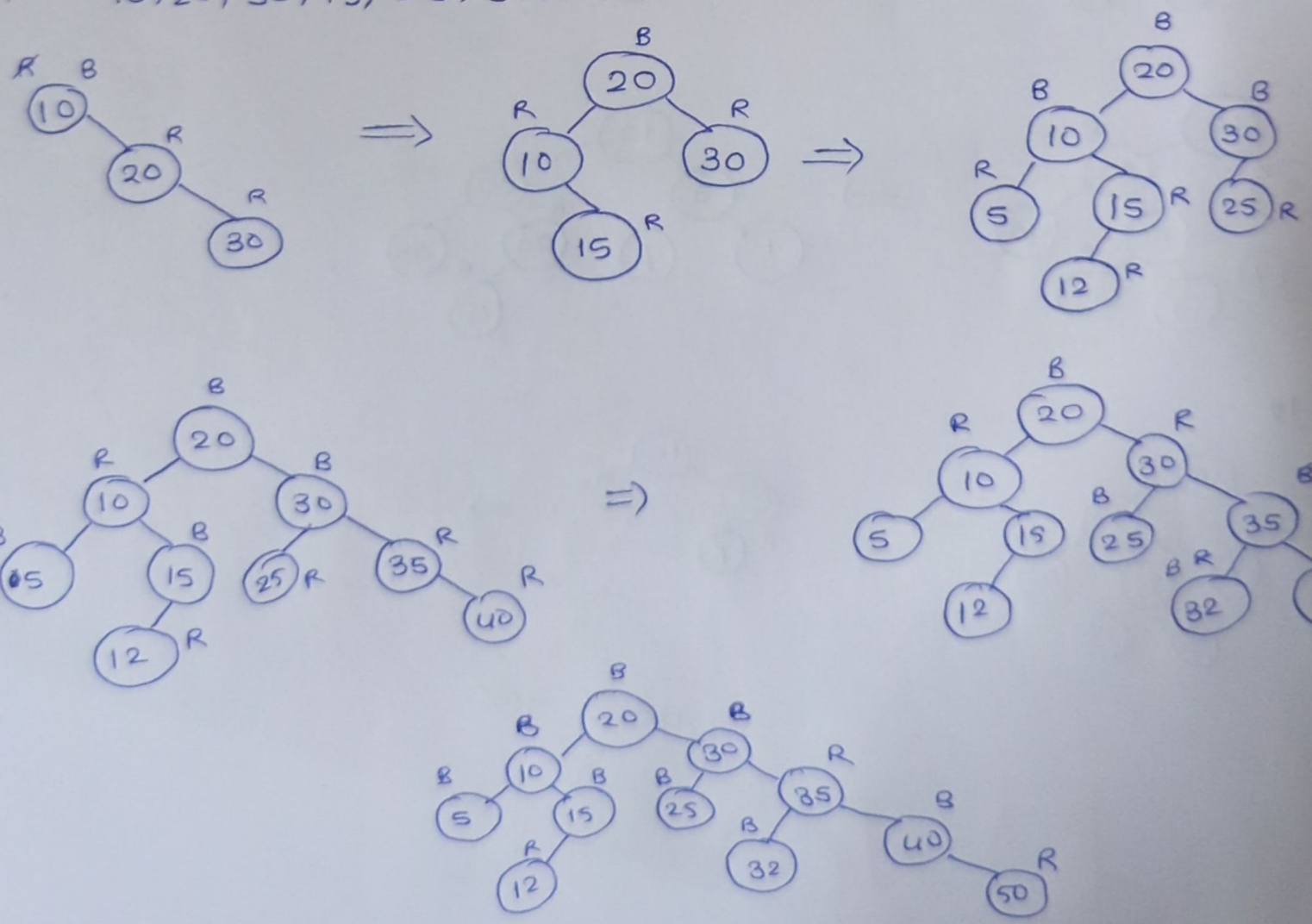
Properties:

1) RBT is a BBST
2) Root Node & N leaf should be Black.
3) If a node is a red then both its children are black.
4) For each node, all paths from the node to leaf contains the same number of black Node.

Insertion properties:

1) Insert new node then colour it Red.
2) If node is root recolour it to black.
3) If Node's parent is Red:
   a) Uncle is Red -
      - Reclolour parent and Uncle to black & GP to Red
   b) Uncle is black or Nill
      - peyorm rotatn & recolour the nodes.

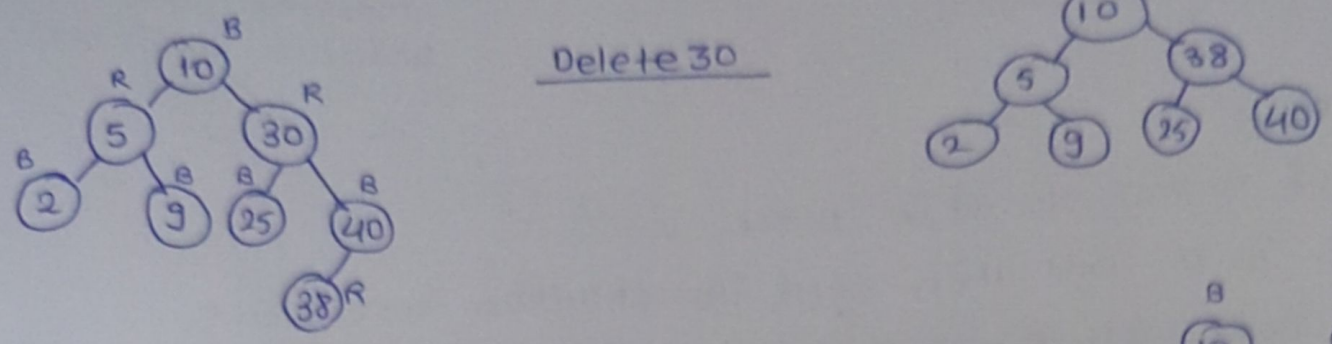
EX. - 10, 20, 30, 15, 25, 5, 12, 35, 40, 32, 50



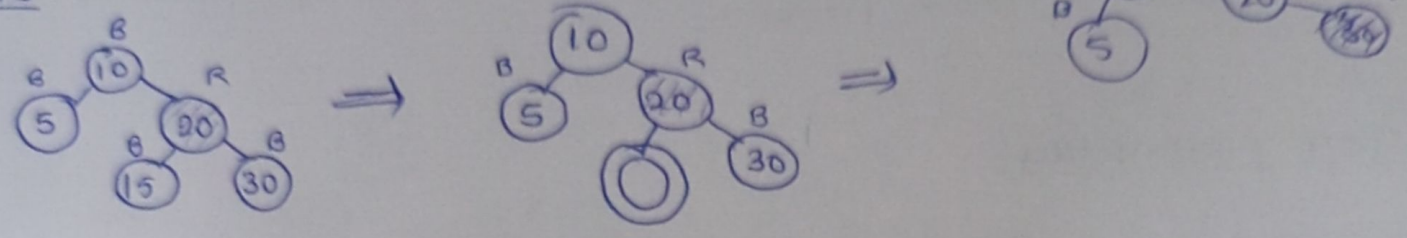
Completely fair schedule

Deletion : 5S of RULES

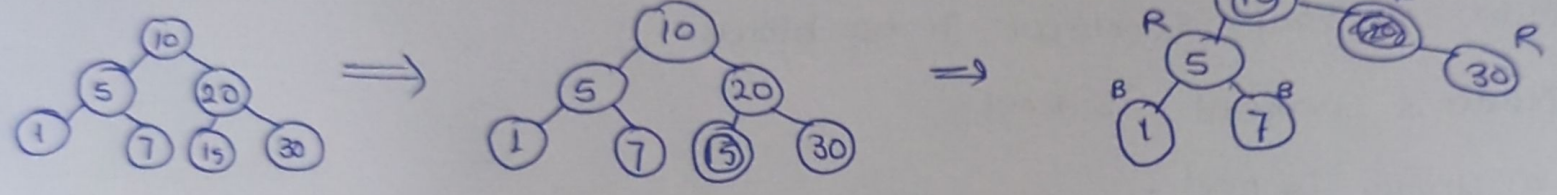
① Delete 30



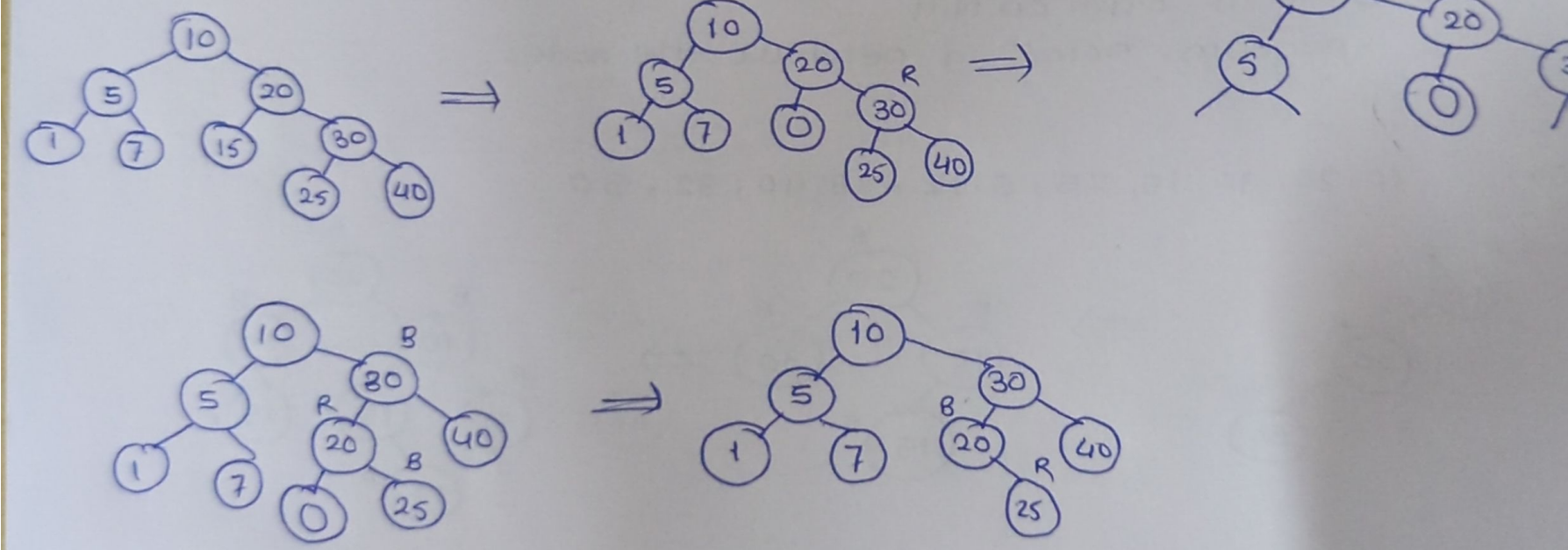
② Delete 15



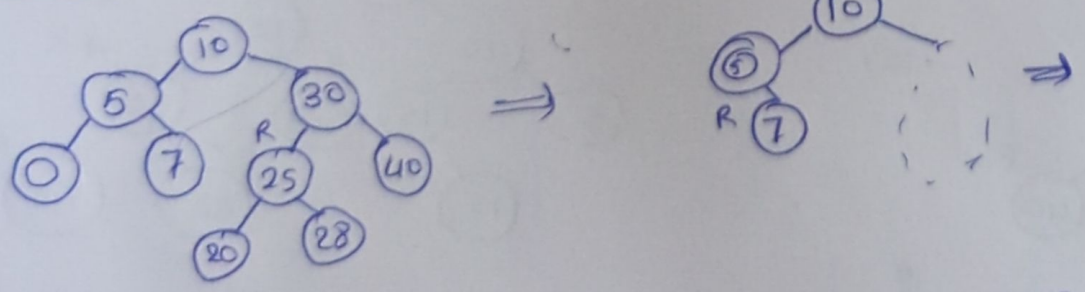
③ Delete 15



④ Delete 15



⑤ Delete 10



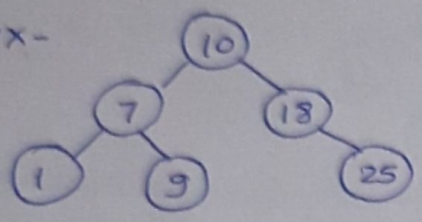
See there from

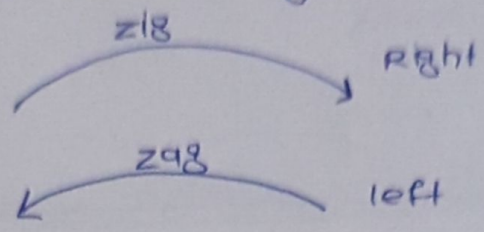
NOT in syllabus

(3) Splay Tree

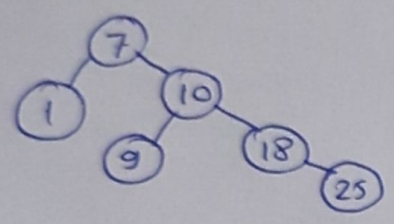
- The primery goal of splay tree is to optimised excess freq used elements by keeping then near the root of the tree.

ex-

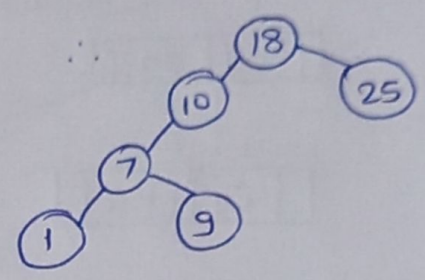




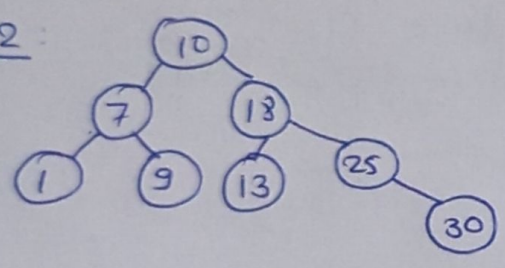
search (7) Hence (7) should be root hence do zig



search (18)



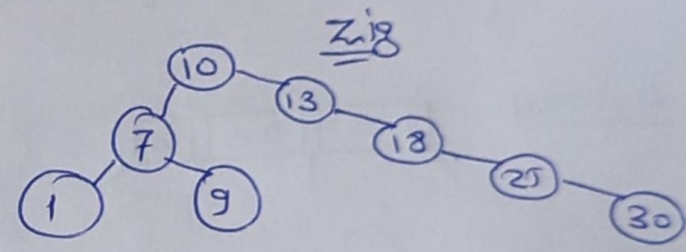
Ex2



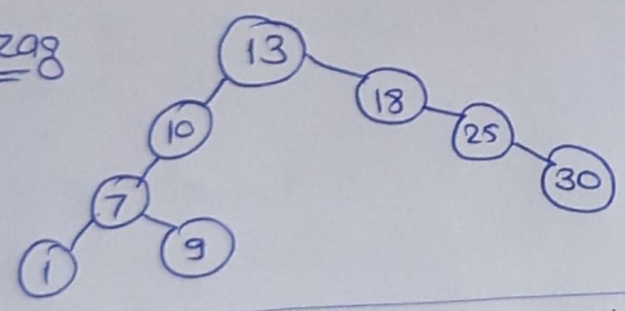
search 13

zig - zag

Zig

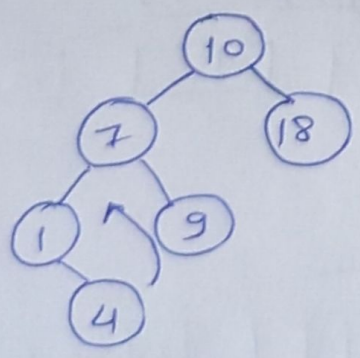


Zag

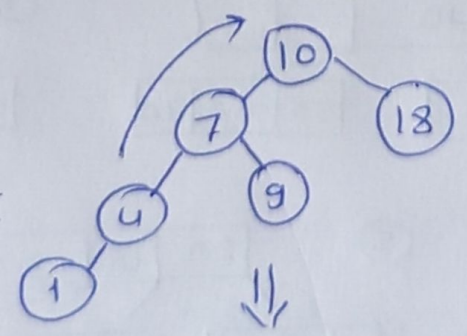


Insertion of splayed tree

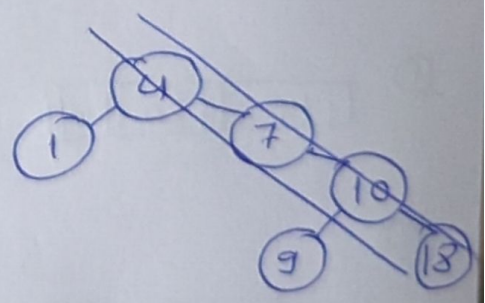
Insert 4

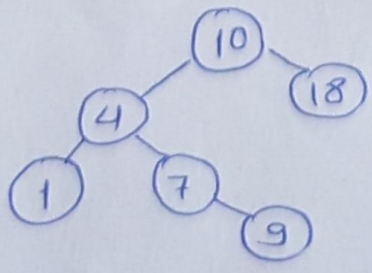


≡

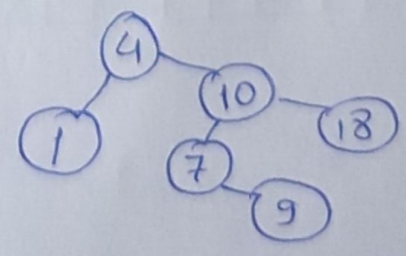


≡

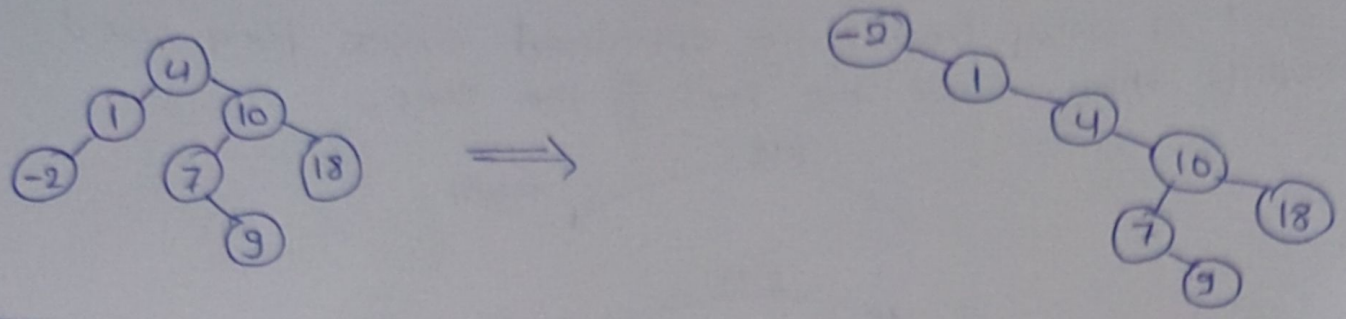




⟹



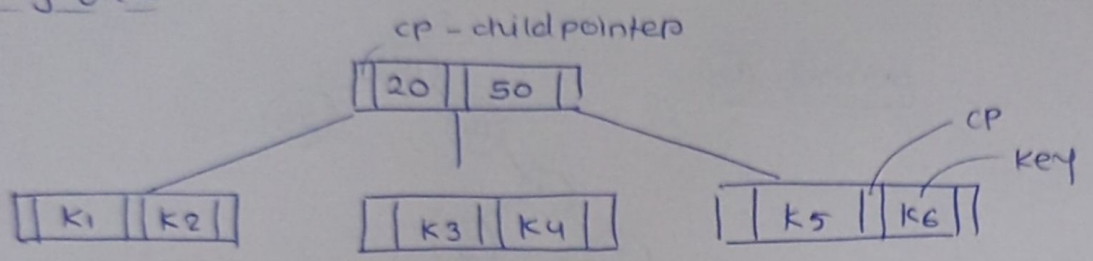
Insert -2



Multi way search Trees (M-ways)

M → No of children

3-way tree



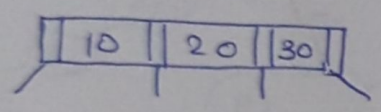
$\{K_1 < K_2 < 20 < K_3 < K_4 < 50 < K_5 < K_6\}$

n-ele

max Ht = n

min Ht = $\log_m (n+1)$

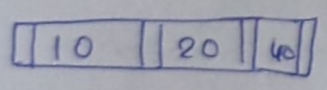
4-way

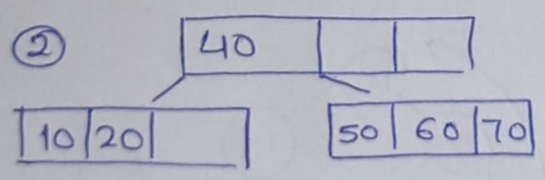


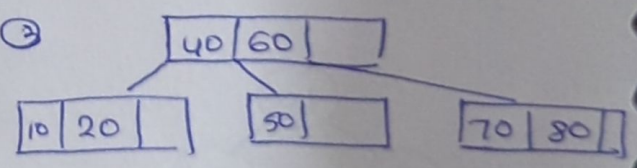
B-Tree properties:

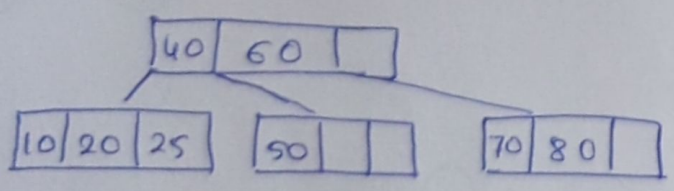
1) Internal node must have $\lceil \frac{m}{2} \rceil$ children.
2) Root can have minimum 2 children.
3) All leaf at same level.
4) Follow Bottom up Approach.

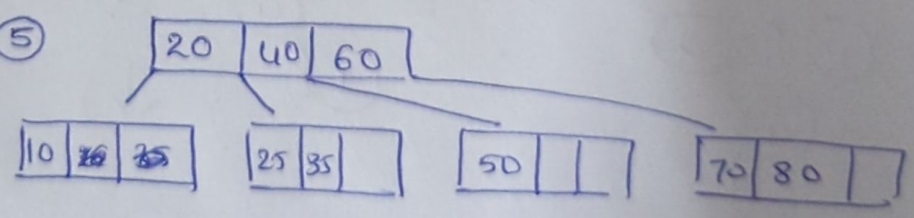
Ex- keys - 10, 20, 40, 50, 60, 70, 80, 25, 35 m = 4

① 

② 

③ 

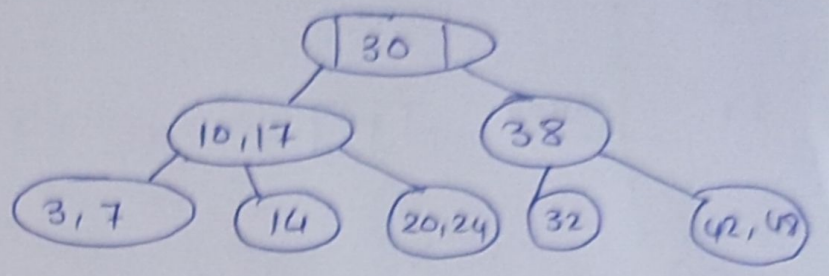
④ 

⑤ 

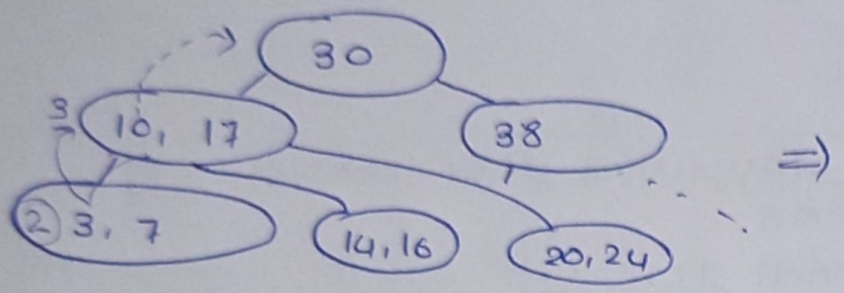
check it!

2-3 Tree

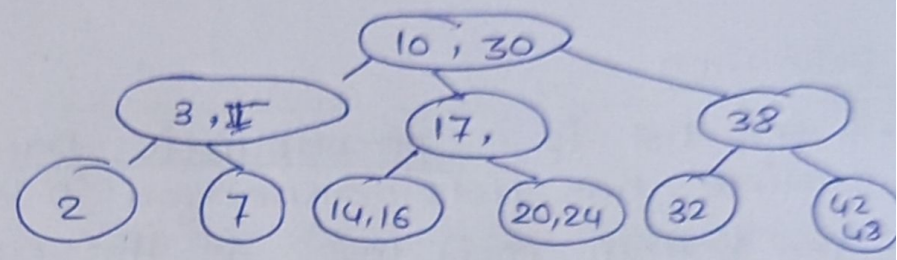
1) 2-3 Trees are special case of B-tree where each node has a maximum of 2 keys and either 2 or 3 children.
2) 2 children ⇒ 1 key
3) 3 children ⟶ 2 keys



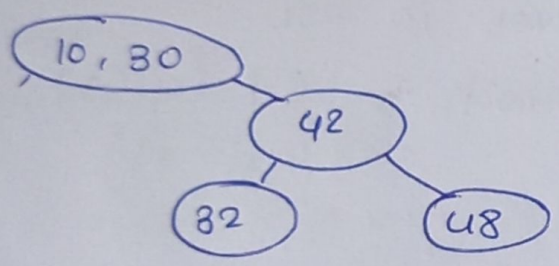
Insert 16 & 2



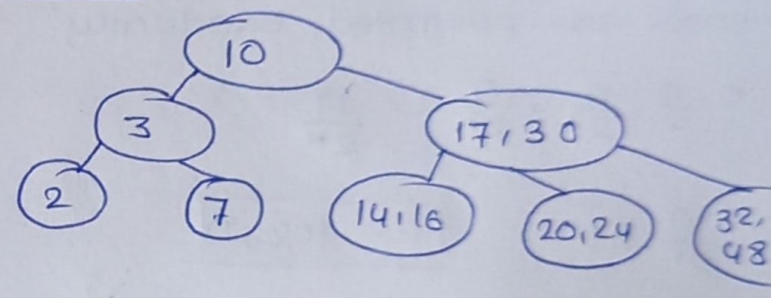
⇒



Delete 38



Delete 42

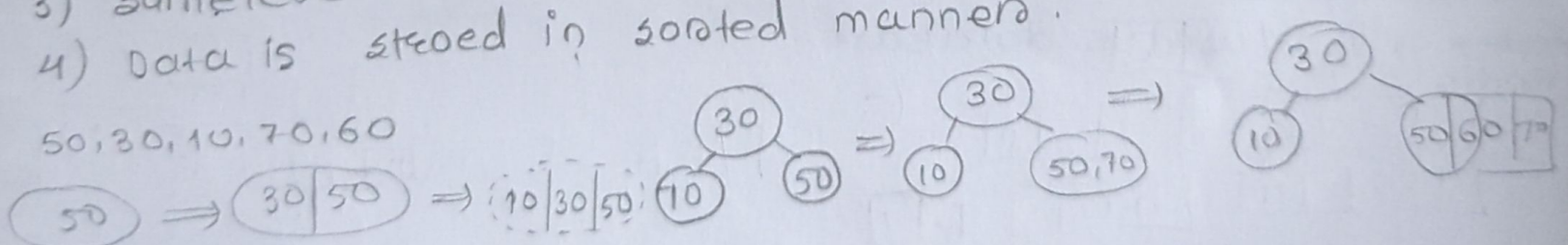


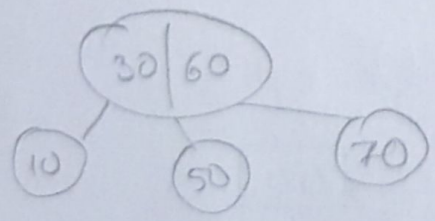
Delete 20 - Directly

Delete -24

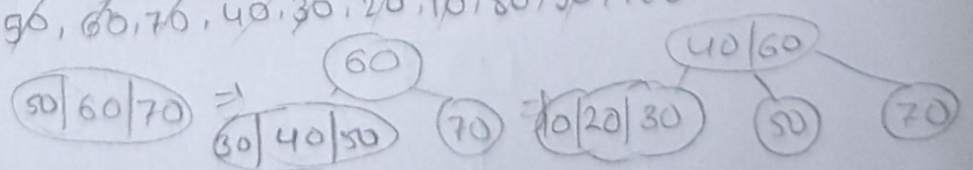
1) Balanced
2) Every internal node is 2node or 3-node
3) Samelevel
4) Data is stored in sorted manner.

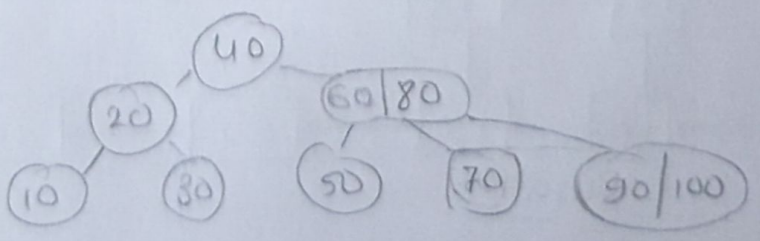
50, 30, 10, 70, 60





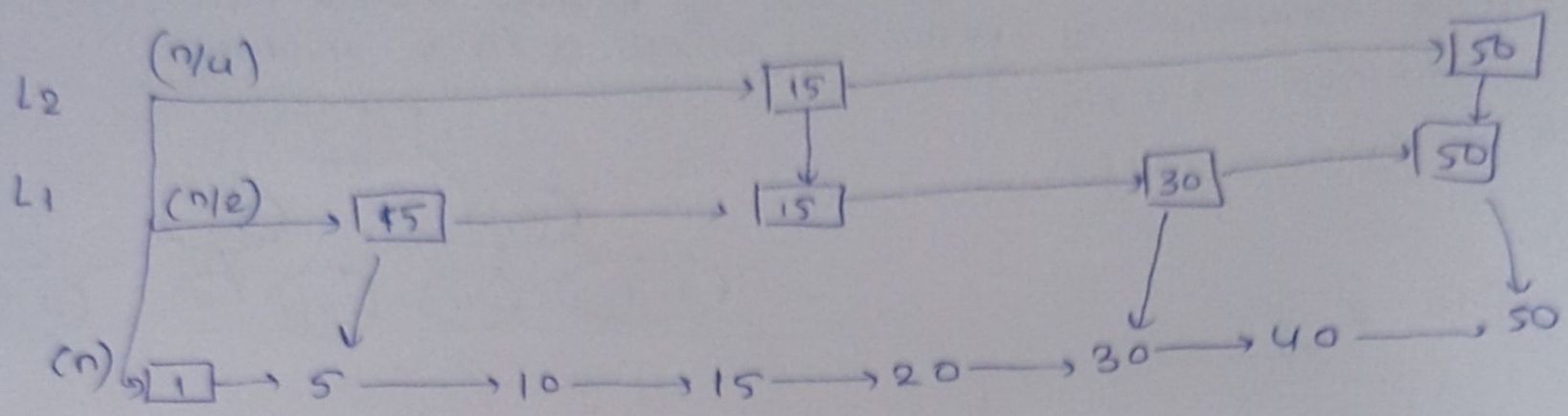
90, 60, 70, 40, 30, 20, 10, 80, 90, 100





31 Jan 26

Skip List



Definition:

- A skip list is a probabilistic Data structures that consists of multiple link lists arranged in layers.
- The bottom most layer is the original LL.
- Each higher layer acts as an express lane, skipping over nodes to speed up operations.
- nodes are promoted randomly to higher levels in RSL.

$n, \frac{n}{2}, \frac{n}{2^2}, \frac{n}{2^3} \cdots \frac{n}{2^k}$

$\therefore \frac{n}{2^k} = 1 \quad \therefore \boxed{k = \log_2 n}$

TC for search $= 2 * \log(n)$
$= \Theta(\log_2(n))$

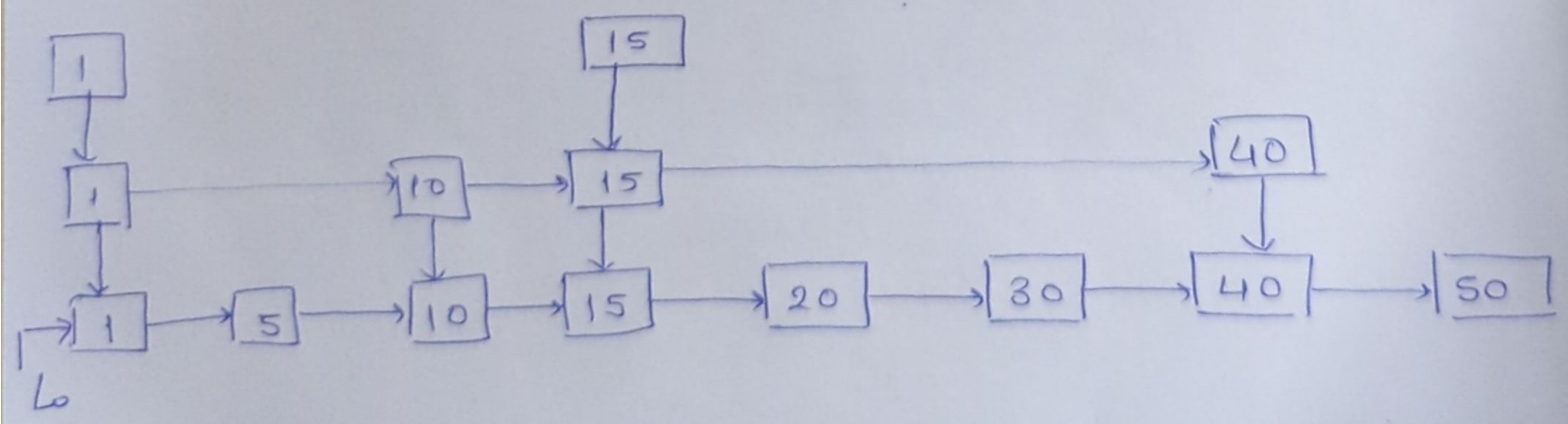
No. of levels in PSL $= \log_2(n)$

$\therefore$ No. of levels in SL $= O(\log_2 n)$

- perfect skiplist ---- Difficult to implement in real life some drawback If you do insertn deletn then structure collapse Restructure it.

Randomised SL

App: ~~Redex~~ Redis, It is used is database.



Randomised approaches ≡ Tossing of coin. eg if H → update to upper level
T → Dont

warm up lemma - No. of levels in an "n" elements skip list is O(log n) with high probability.

Space complexity -

Total nodes - $n + \frac{n}{2} + \frac{n}{2^2} + \frac{n}{2^3} + \cdots + \frac{n}{2^k}$

$\therefore n\left[\frac{1}{2} + \frac{1}{2^2} + \cdots \frac{1}{2^k}\right]$

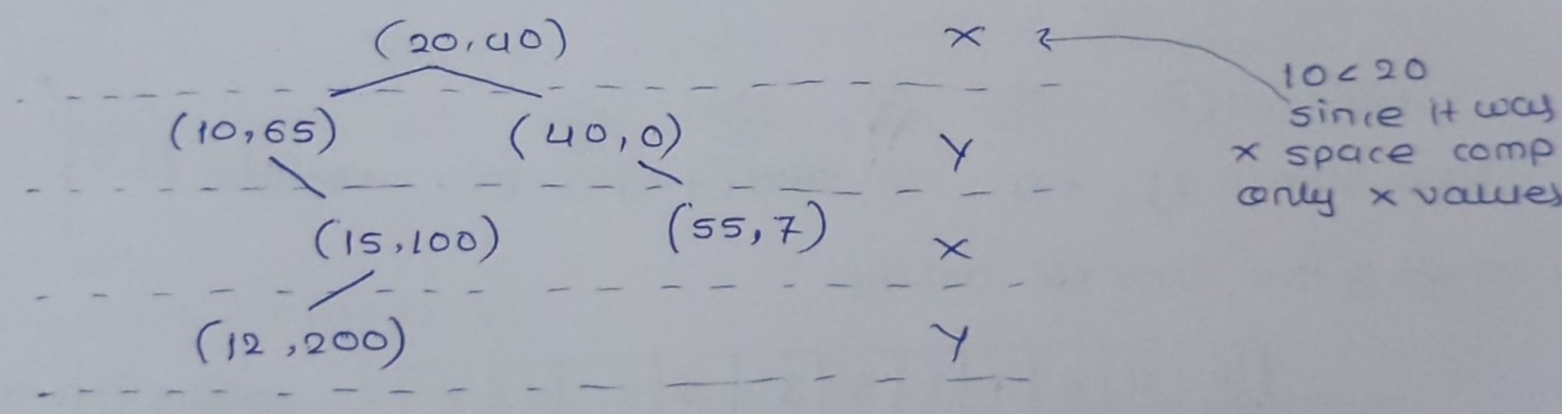
$\therefore n\left[\frac{1}{1-\frac{1}{2}}\right] = \underline{2n}$

$\therefore$ **O(n)**

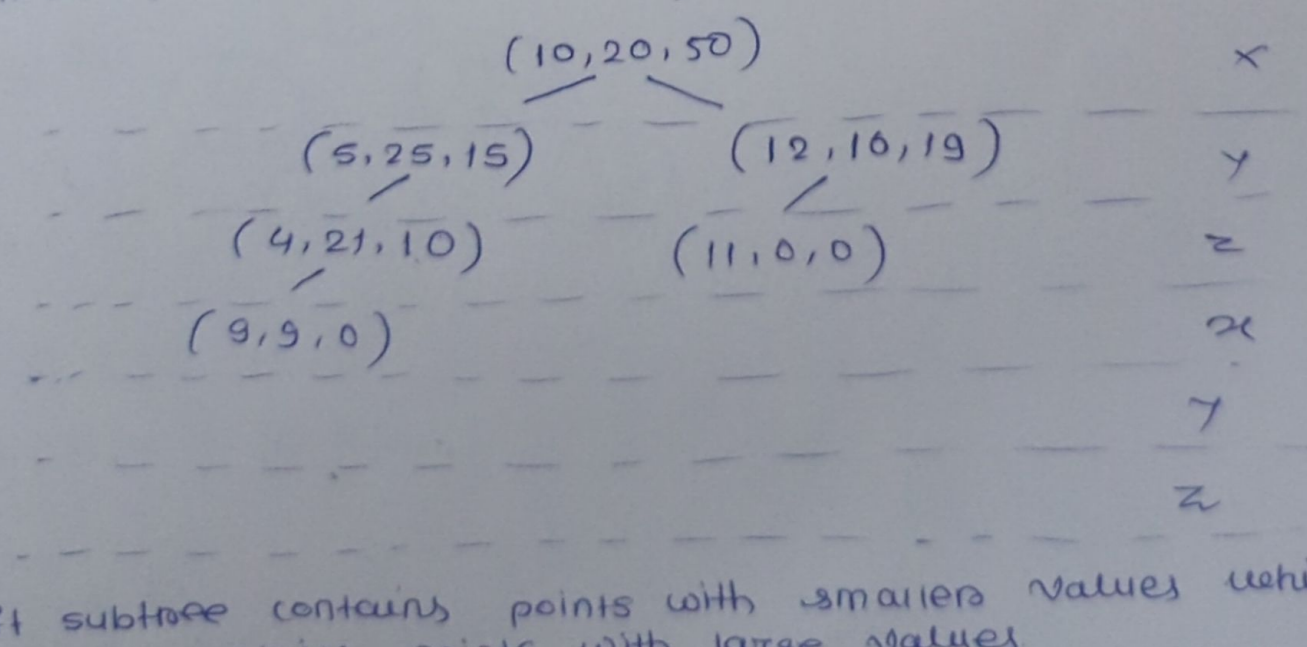
K-D Tree - (K-dimentional tree)

Definition - A KD Tree is a space partitioning data structure used for organizing points in a k-diminsional space.

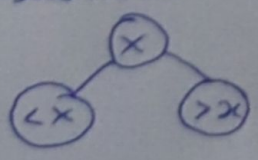
ex. k=2
∴ 2D Tree
Data points: (20,40) (10,65) (15,100) (40,0) (12,200) (55,7)



Ex-2 k=3
∴ 3-D Tree (10,20,50) (5,25,15) (4,21,10) (9,9,0) (12,10,19) (11,0,0



- left subtree contains points with smaller values while right subtree contains points with large values



BST

Segment Tree

- To use range queries easily and in faster way.

Array :

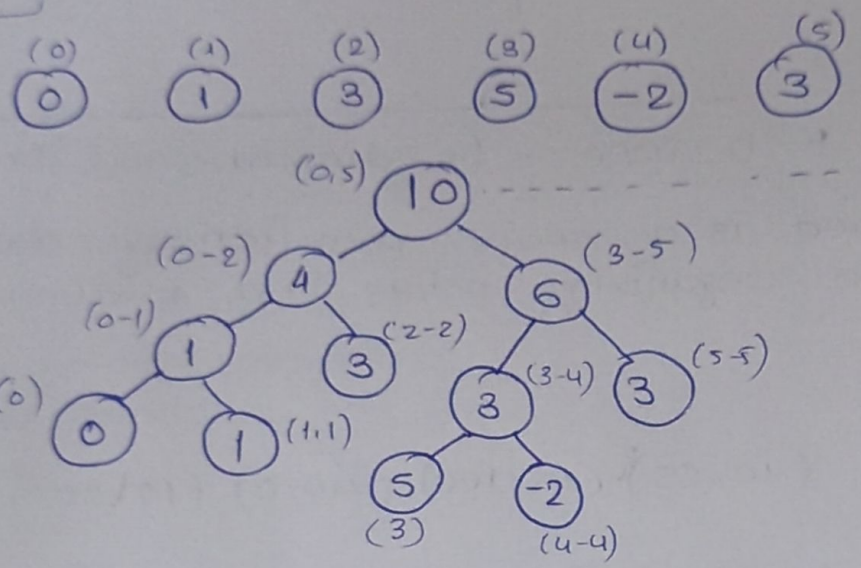
0	1	2	3	4	5
0	1	3	5	-2	3

sum(0,2) = 0+1+3 = 4
sum(3,5) = 5-2+3 = 6
sum

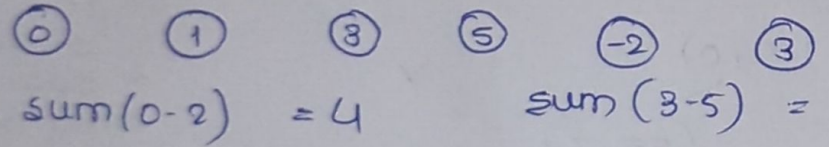
update(2,7) ≡ pos = 2 Value = 7
sum(0,2) = 0+1+7 = 8

Sum = O(n) update = O(1)	BUT in ST we get O(log n) only.

N-nodes →



--- Add° of all



Now, sum(0-2) = 4 sum(3-5) = 6 update(2,7) =

length of the tree ka array:

10	4	6	1	7	3	3	0	1	5	2

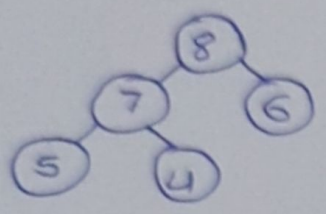
---wrong, NO Null values are mentioned

Unit 3 - Heaps

Definition: A Heap is a complete BT that satisfies the heap property, that is, the value of each parent node is greater or smaller than its child node.

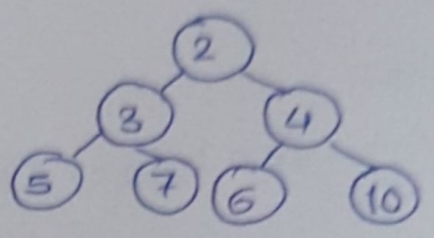
① Max Heap

$A[parent] \geq A[child]$

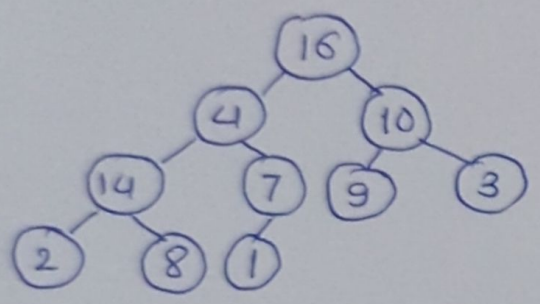


② Min Heap

$A[parent] \leq A[child]$

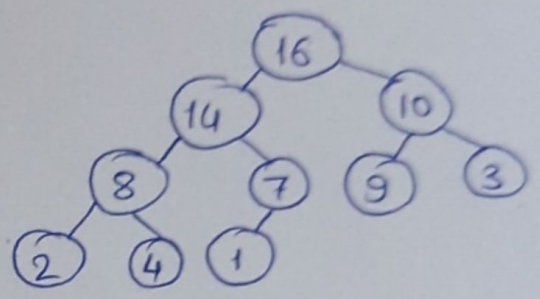


Example



Max-Heapify ( ) ⟶

TC : O(log n)

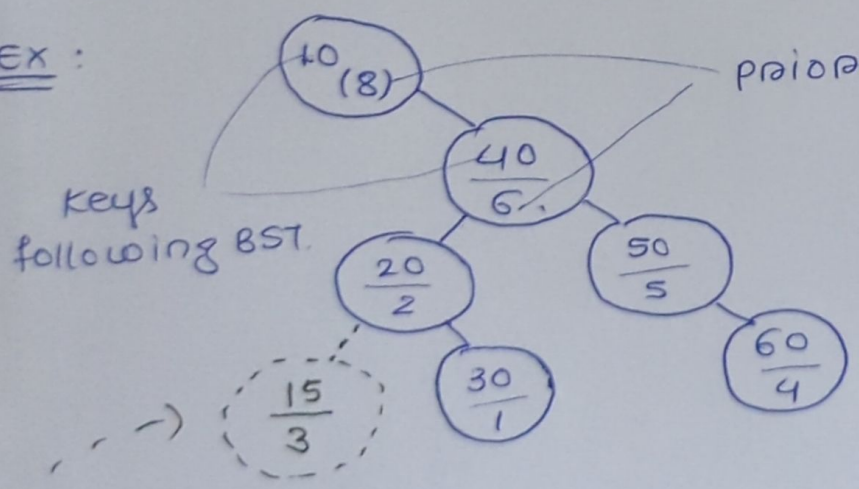


Treaps (Tree + Heap) unit 2 part

Def^n: A treap is a BST that maintains a heap property using randomly assigned priorities.

1) BST property
2) Heap property (min/max)

Ex:



priorities following. Heap property

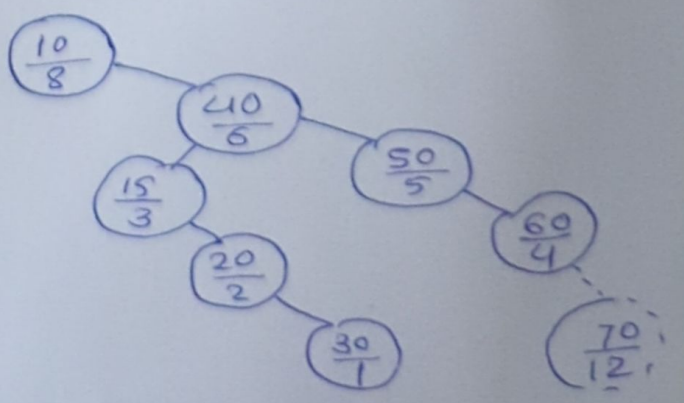
Keys following BST.

Operations

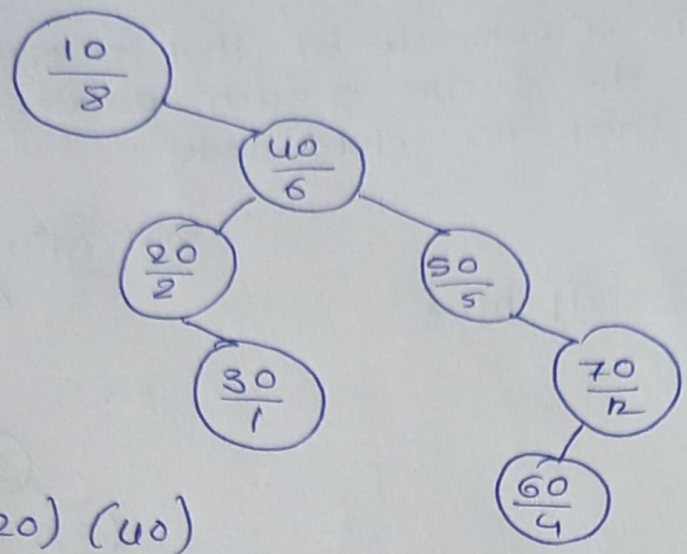
1) search → follows normal BST
2) Insert

Insert (15, 3)

1) Insert like BST
2) Rot^n for heap prop. ⟹



Insert (70, 12)



solve it ....

① Insertion

② Deletion. (60) (20) (40)