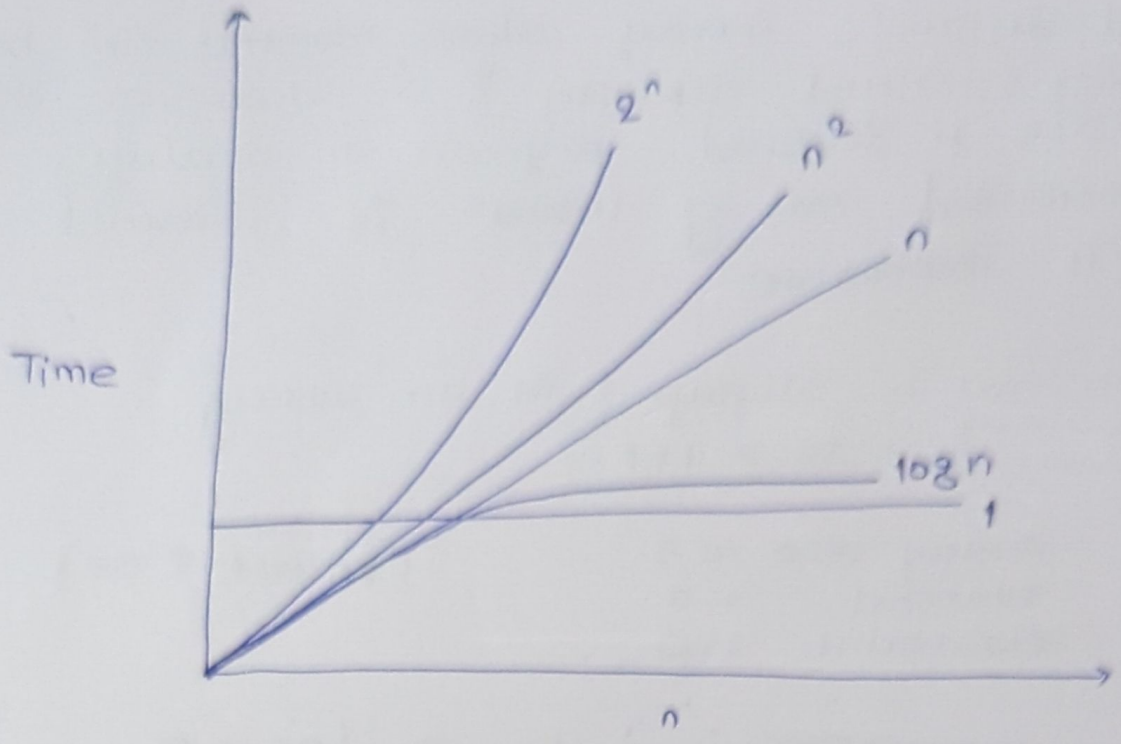


① Asymptotic Not (worst)

$n \geq n_0$

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

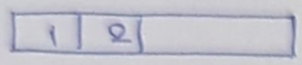
A (worst)	B (Realistic)
₹ 500 × 500 = 250 000	500 × 1 + 1 × 499 = ₹1000

Difinition - Is a method for analysing the time complexity of seq of operat^n which gives more realistic estimat~~e~~ 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 insertion is constant eventhough resizing is expensive.

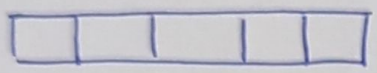
→ Insertion cost is always 1 in an array.

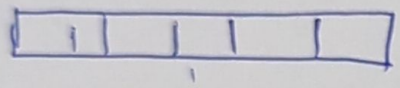


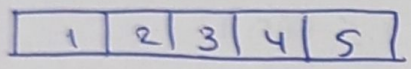
= TC = 1+1 ..

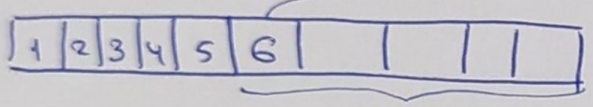
Given · Array size = 5
Total cost = 0
ele-count = 0

(Insert 8 ele)

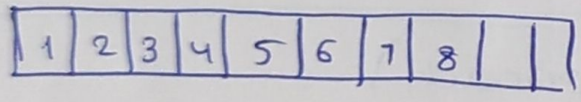
Step 1 -  tc = 0 | ec = 0

2) -  TC = 1 | EC = 1

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

6)  TC = 5+5+1 =

7)

8)  11+1+1 = 13

(This was Asymptotic)

Now Amortised cost

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

Avg. cost for this = 1.625

Time Analysis of Recursive programs:

① Back substitution

② Recursive Tree

③ Masters Theorem.

① Back substn

$$\boxed{\begin{array}{l} T(n) = 1 + T(n-1) \quad ; \quad n>0 \\ T(0) = 1 \qquad\qquad\quad n=0 \end{array}} \quad \text{---} ①$$

$$T(n-1) = 1 + T(n-2) \quad \text{---} ②$$

$$T(n-2) = 1 + T(n-3) \quad \text{---} ③$$

② − 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 $n - K = 0$, $K = n$

$$\therefore T(n) = n+1 = n = O(n)$$

Example 2 - 24 carat

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

$$T(n-1) = (n-1) + T(n-2) \text{ ----- } ②$$

$$T(n-2) = (n-2) + T(n-3) \text{ ----- } ③$$

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

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

$\therefore$ k times

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

$$n - k + 1 = 1$$

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

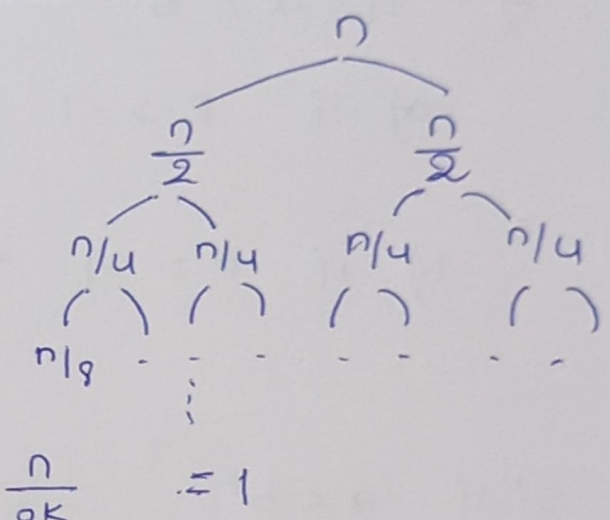
$$\therefore \quad n + (n-1) + (n-2) + \cdots + [n-(n-2)] + T(1)$$

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

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

Reccursive tree:

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



$\therefore$ $n = 2^k$

$k = \log_2 n$

for every level $k = n \log_2 n$

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

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

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

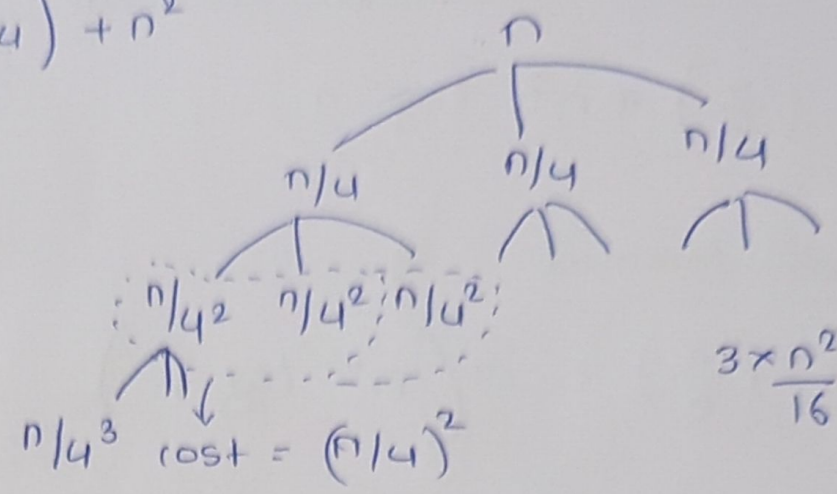
3) if $a < b^k$

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

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

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

$T(1) = 1$



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

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

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

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

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

$k = \log_4 n$

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

$n^2\left[1 + \frac{3}{16} + \frac{9}{16} \cdots\right]$ ------ 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 format:

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

∴ Here,

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

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

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

3rd case:

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

$= n^2$

$= O(n^2)$

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

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

$b^k = 2^2 = 4$

$\therefore \boxed{a = b}$

2nd case: (a)

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

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

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

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

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

$= \Theta(n^{\log_4 2} \log_n^{p+1})$

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

$a < b^k$

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

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

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

$\Theta(n^{0.51})$

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

$a < b^k$

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

$n^k \log 2^$

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 }

$\log_n 2$

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

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

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

$T(n) = \Theta(n^{\log_a b}) = \Theta(n^{\log_{\sqrt{2}} 2})$

$\Theta(\sqrt{n})$

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

$a > b^k$

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

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

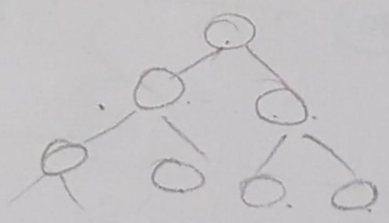
$= \sqrt{n}$

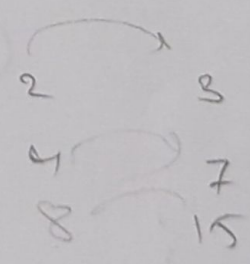
Tree Revisions

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

→ (2 × leaf nodes) − 1

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





8, 11, 15
4, 5, 7
2, 1, 3

(2 * leaf) − 1

Unit 2 - Search Trees

→ Why trees?
organised
logarithmic time

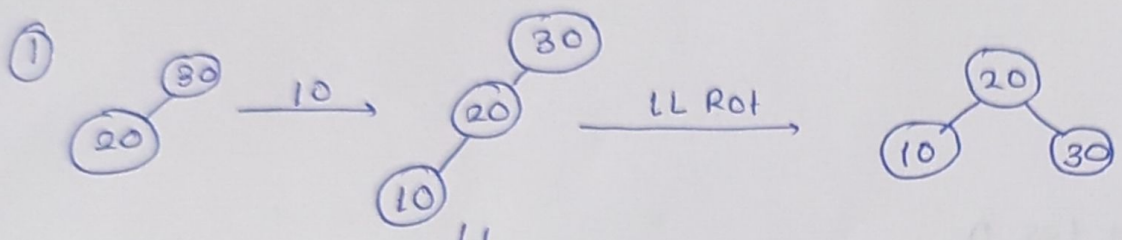
① AVL

Balance factor = $|H_L - H_R| \leq 1$

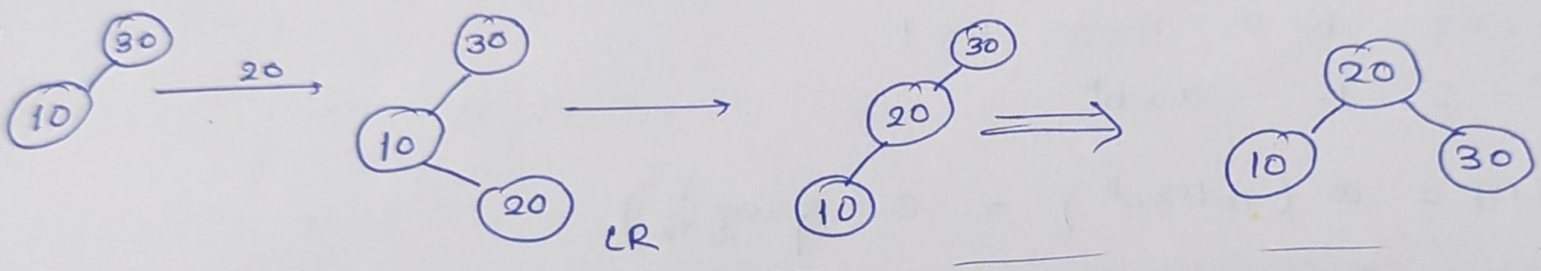
Ht of left & right children of every node is differed by ± 1

* Insertion

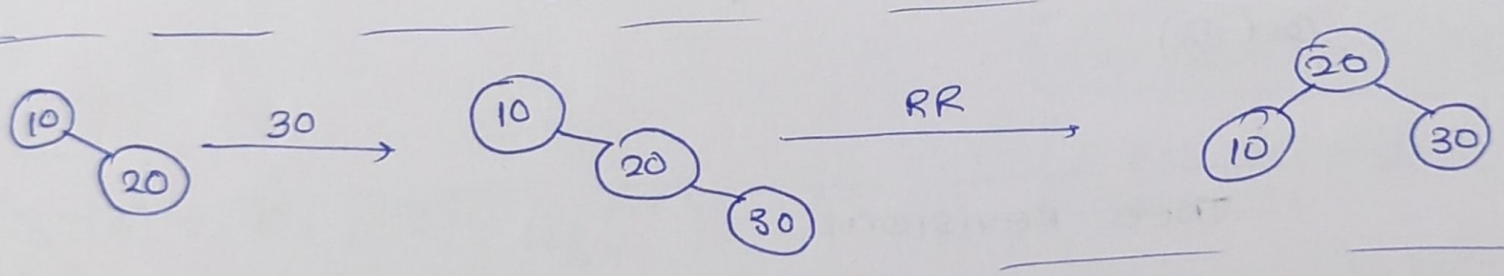
① LL



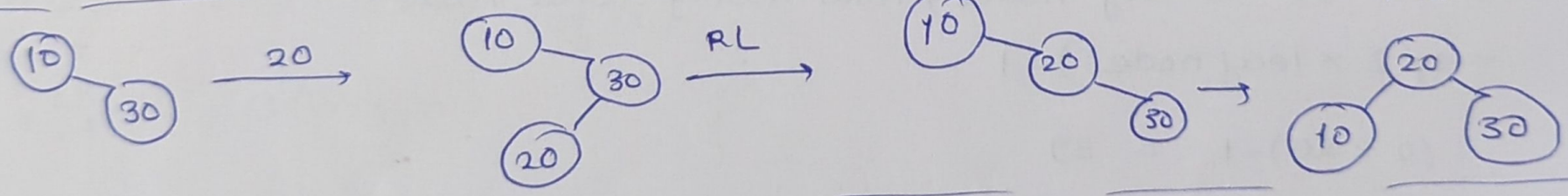
② LR



③ RR



④ RL



27 Jan 2025

② Red Black Tree

colours are req to maintain self balance property

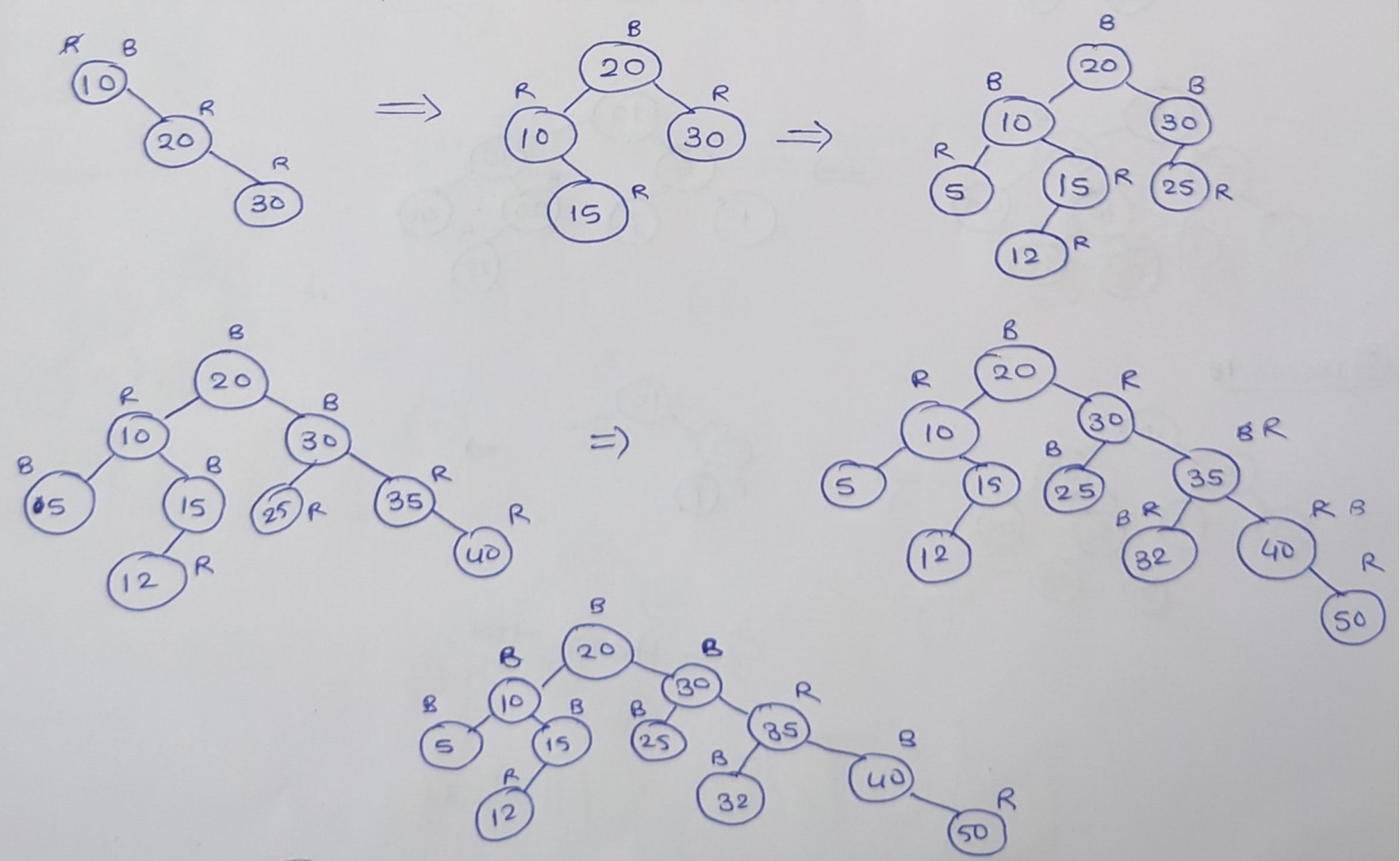
Properties :

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
      - peyoam rotat° & recolour the nodes.

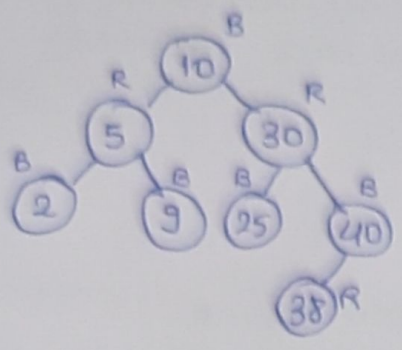
EX. - 10, 20, 30, 15, 25, 5, 12, 35, 40, 32, 50

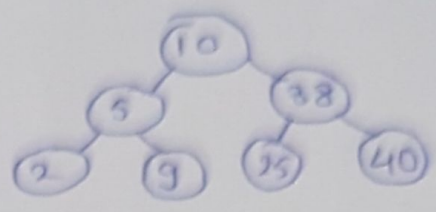


completely fair schedule

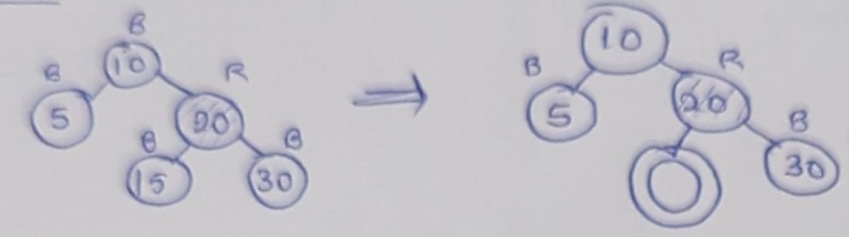
Deletion : 5 cases

① Delete 30

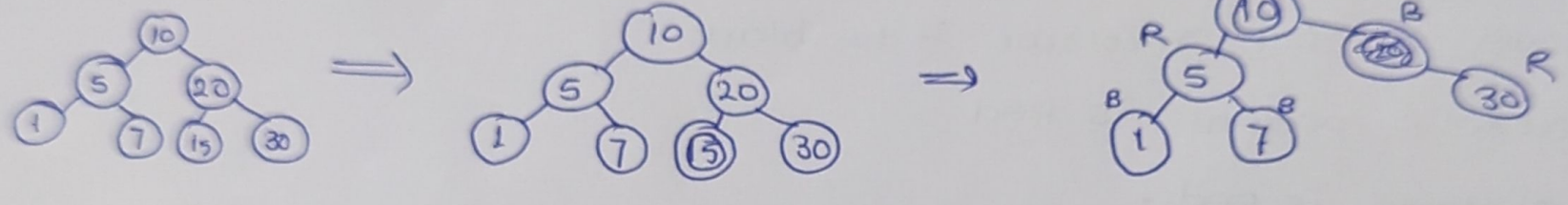




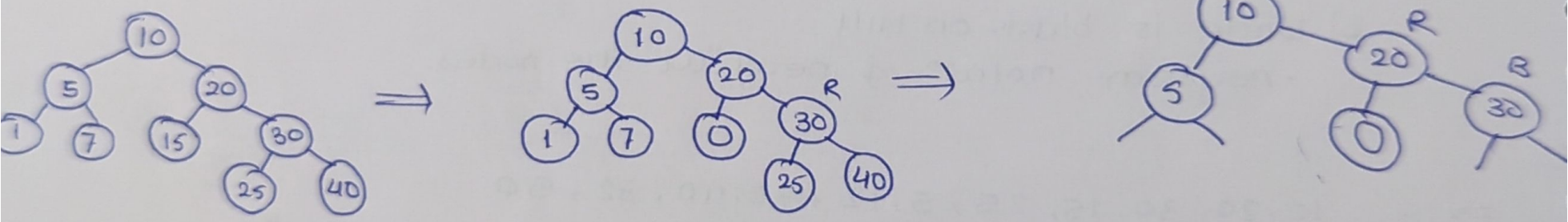
② Delete 15

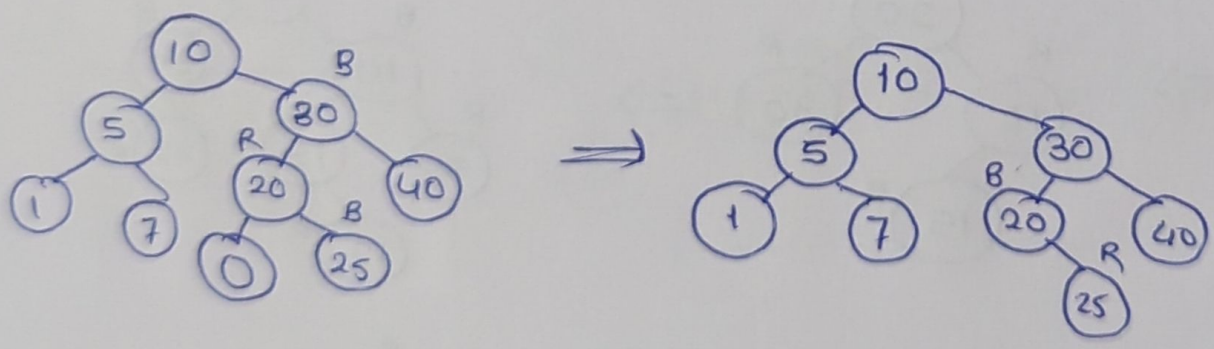


③ Delete 15

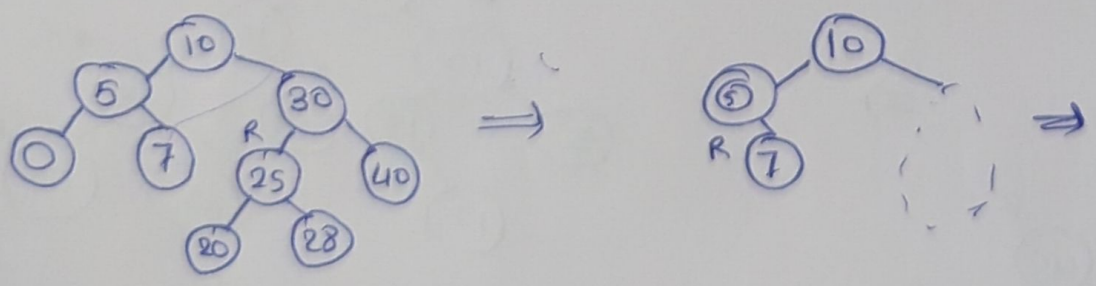


④ Delete 15





⑤ Delete 10



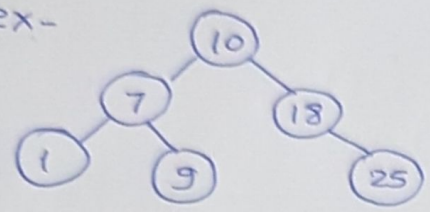
See there from

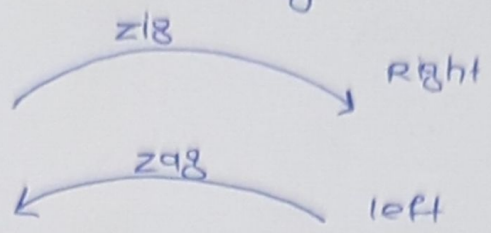
NOT in syllabus

③ Splay Tree

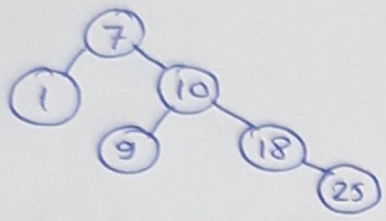
- The primery goal of splay tree is to optimised excess freq used elements by keeping then near the root of the tree.

ex-

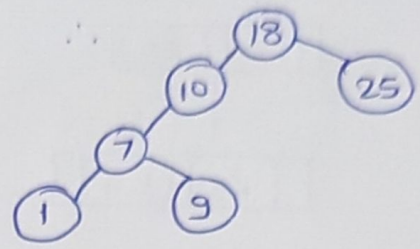




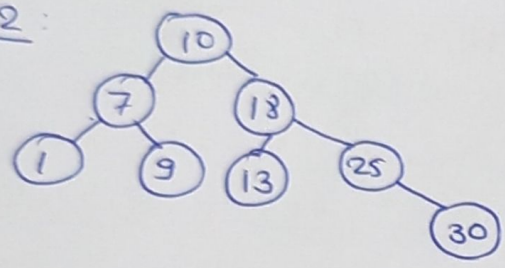
search ⑦ Hence ⑦ should be root hence do zig



search (18)



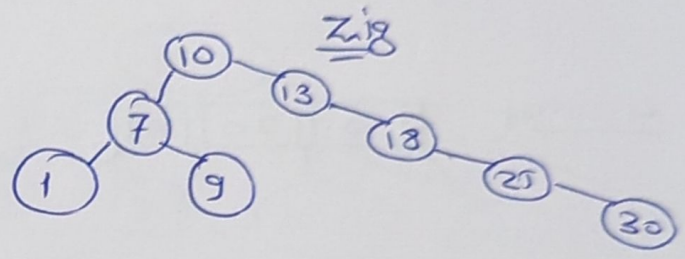
Ex2:



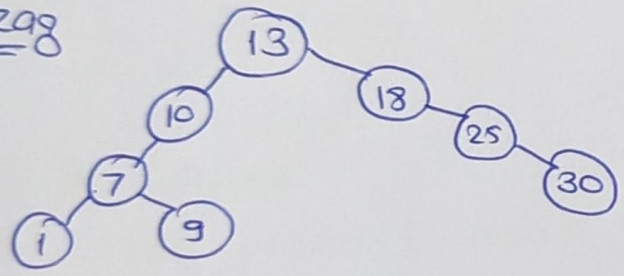
search 13

zig - zag

Zig

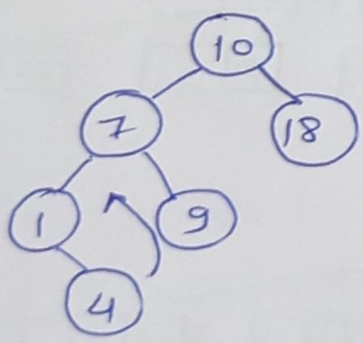


Zag

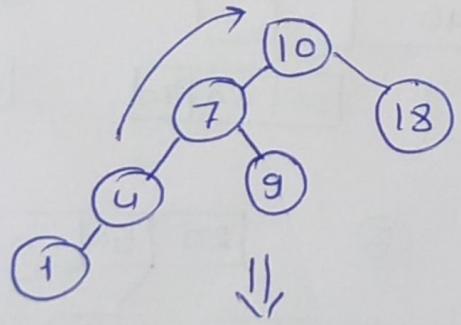


Insertion of splayed tree

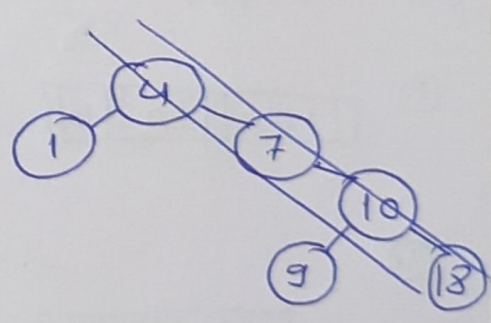
Insert 4

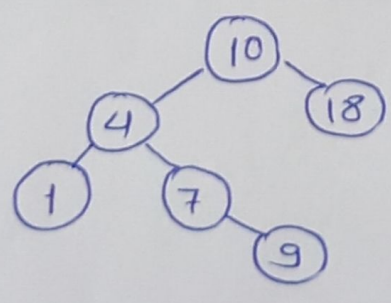


≡

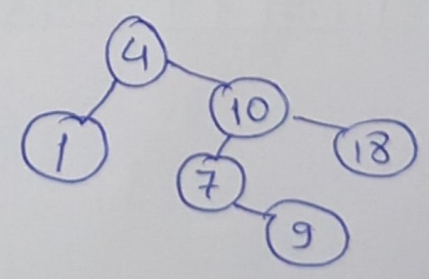


≡





⟹

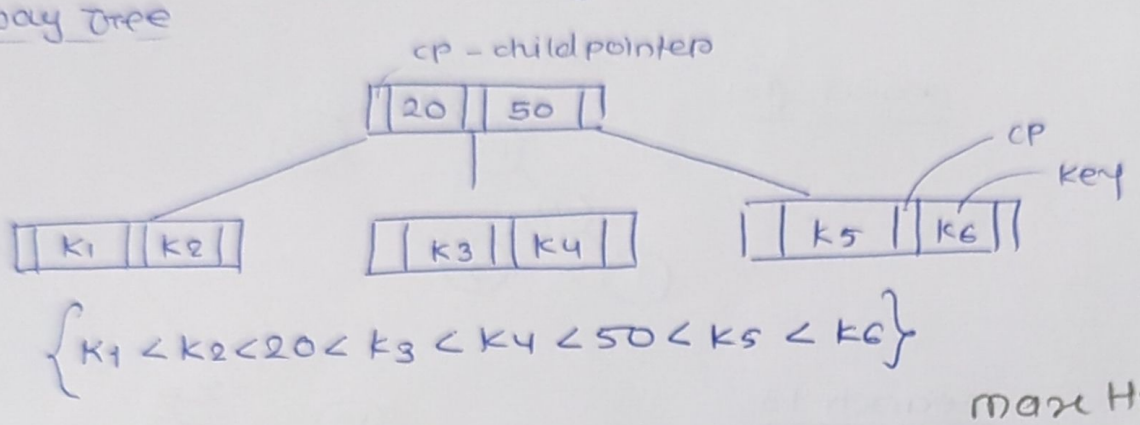


Insert -2

Multi way search Trees (M-ways)

M → No of children

3-way Tree



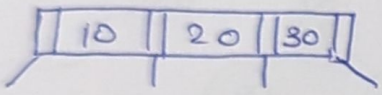
$\{K_1 < K_2 < 20 < K_3 < K_4 < 50 < K_5 < K_6\}$

n-ele

max Ht = n

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

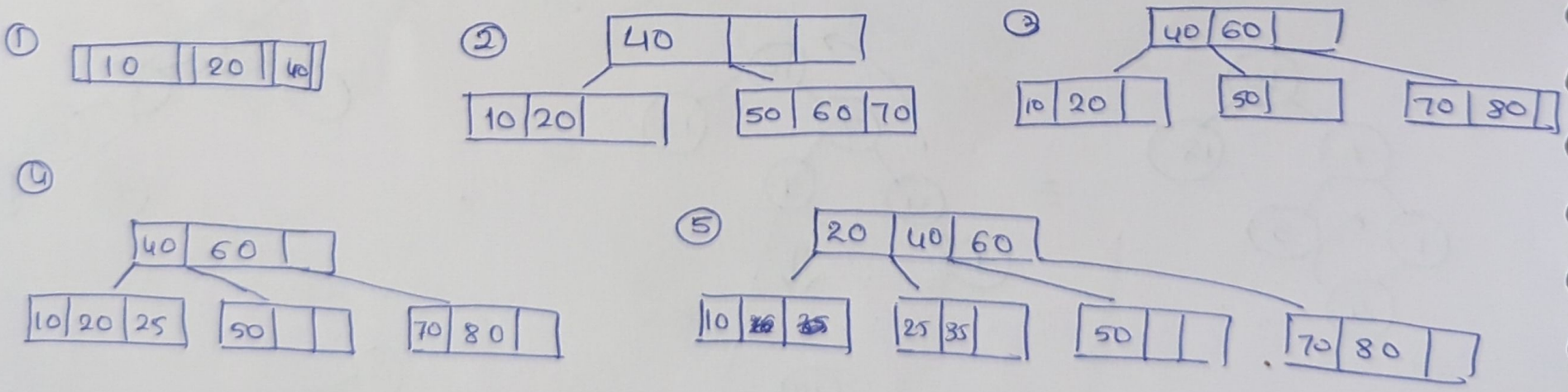
4-way



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.

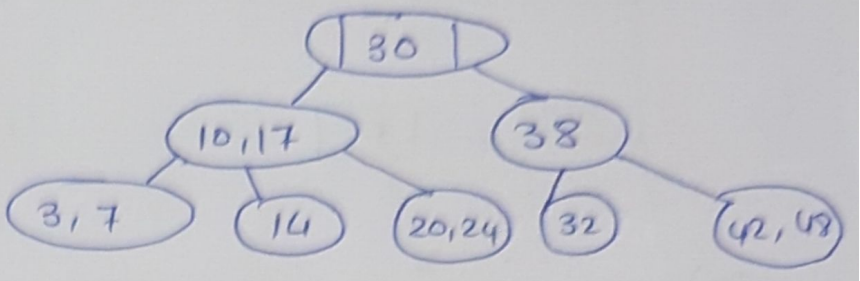
Ex- keys - 10, 20, 40, 50, 60, 70, 80, 25, 35 m = 4



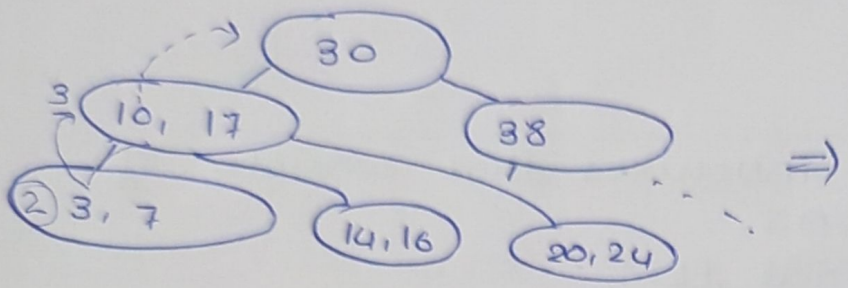
check it!

2-3 Tree

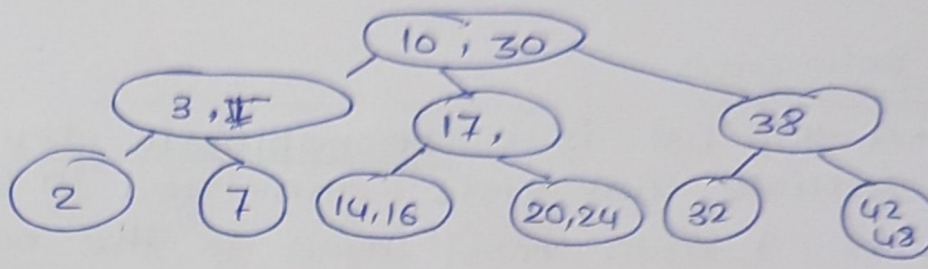
1) 2-3 Trees are special case of B-tree where each node has a maximum of 2 keys and either 2 or 3 children.
2) 2 children ⇒ 1 key
3) 3 children ⇒ 2 keys



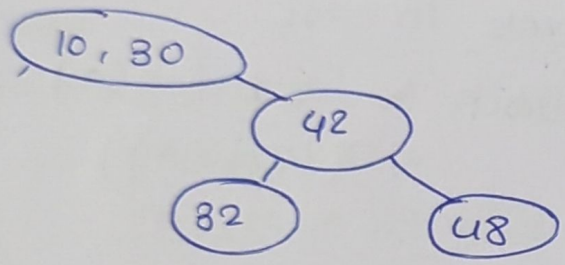
Insert 16 & 2



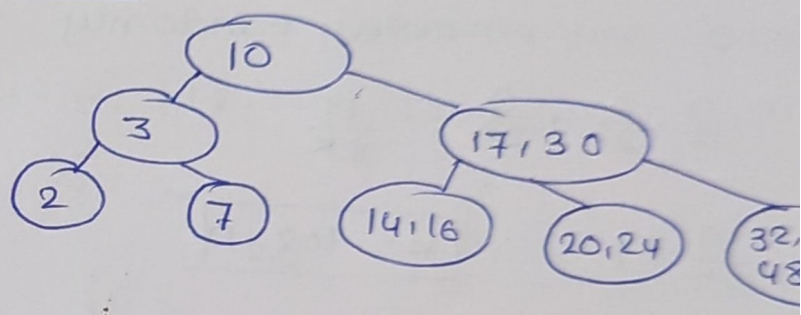
⇒



Delete 38



Delete 42

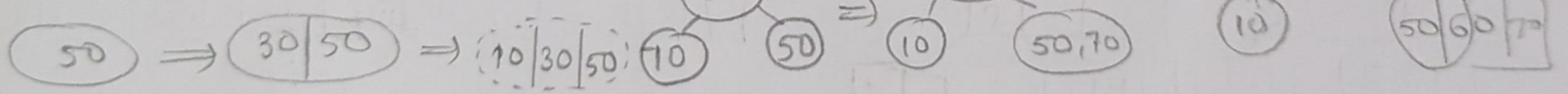


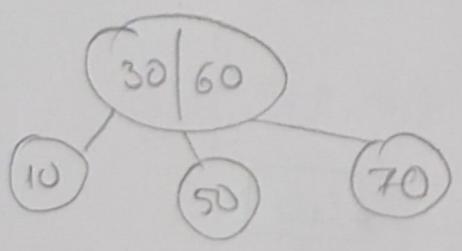
Delete 20 - Directly

Delete -24

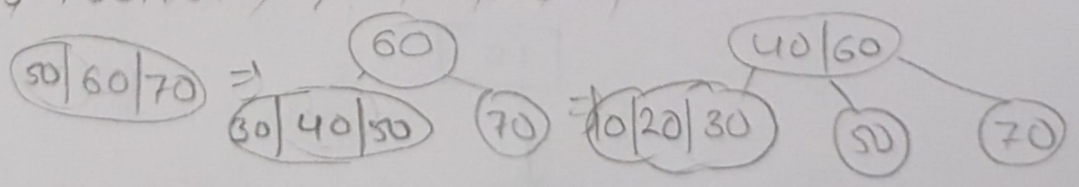
1) Balanced
2) Every internal node is 2node or 3-node
3) Samelevel
4) Data is stored in sorted manner.

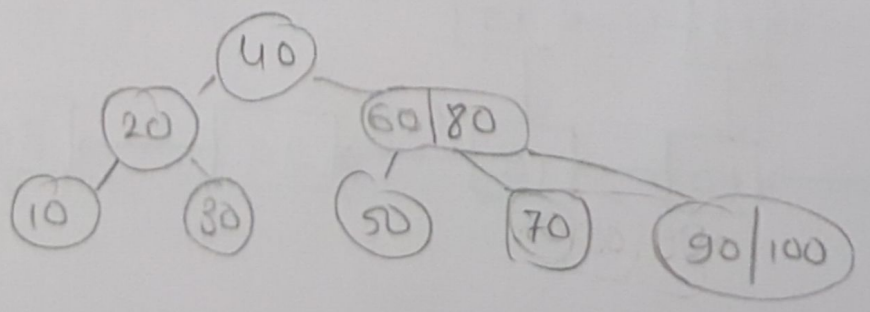
50, 30, 10, 70, 60





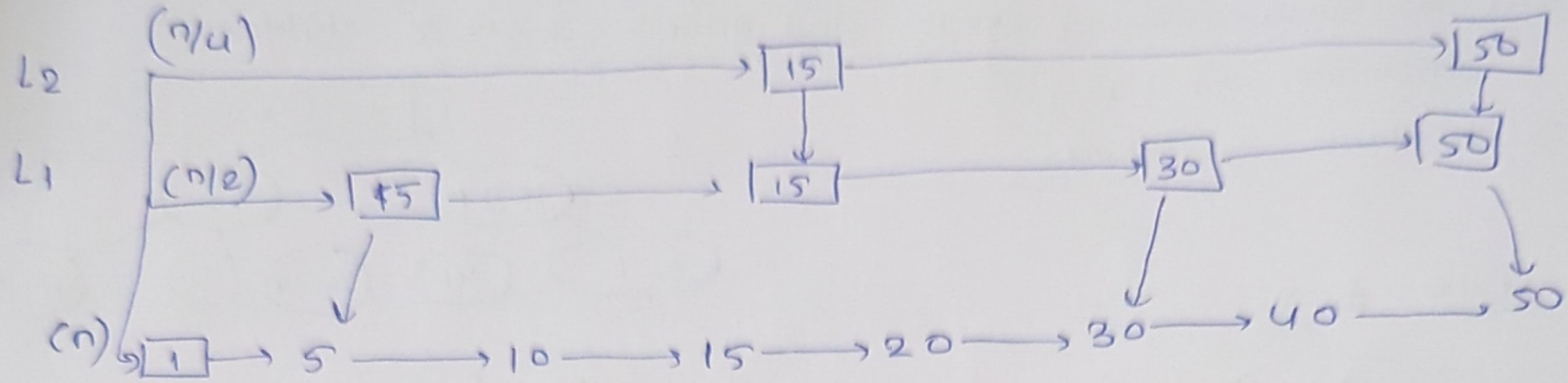
96, 60, 70, 40, 30, 20, 10, 80, 90, 100





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.

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

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

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

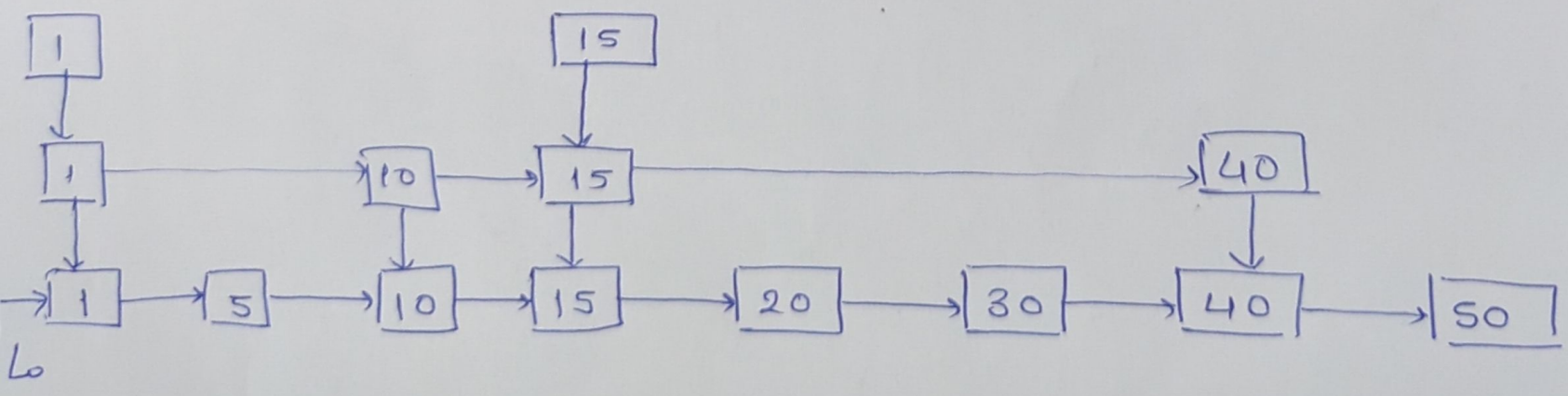
TC for search $= 2 * \log(n)$
$= O(\log_2(n))$

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

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

Randomised SL

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



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

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

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

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

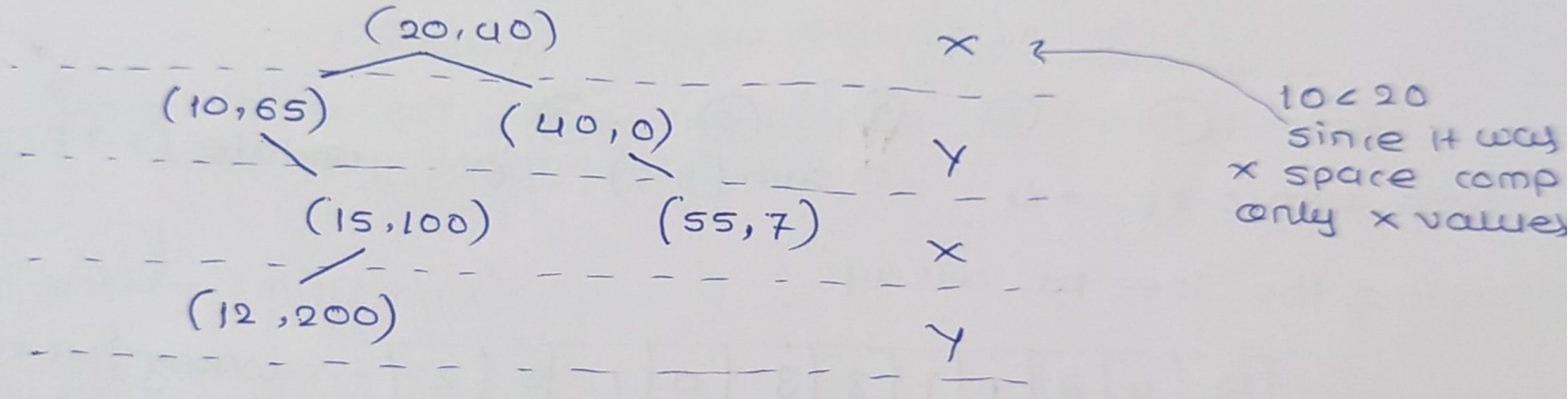
K-D Tree - (K-dimentional tree)

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

ex. k=2

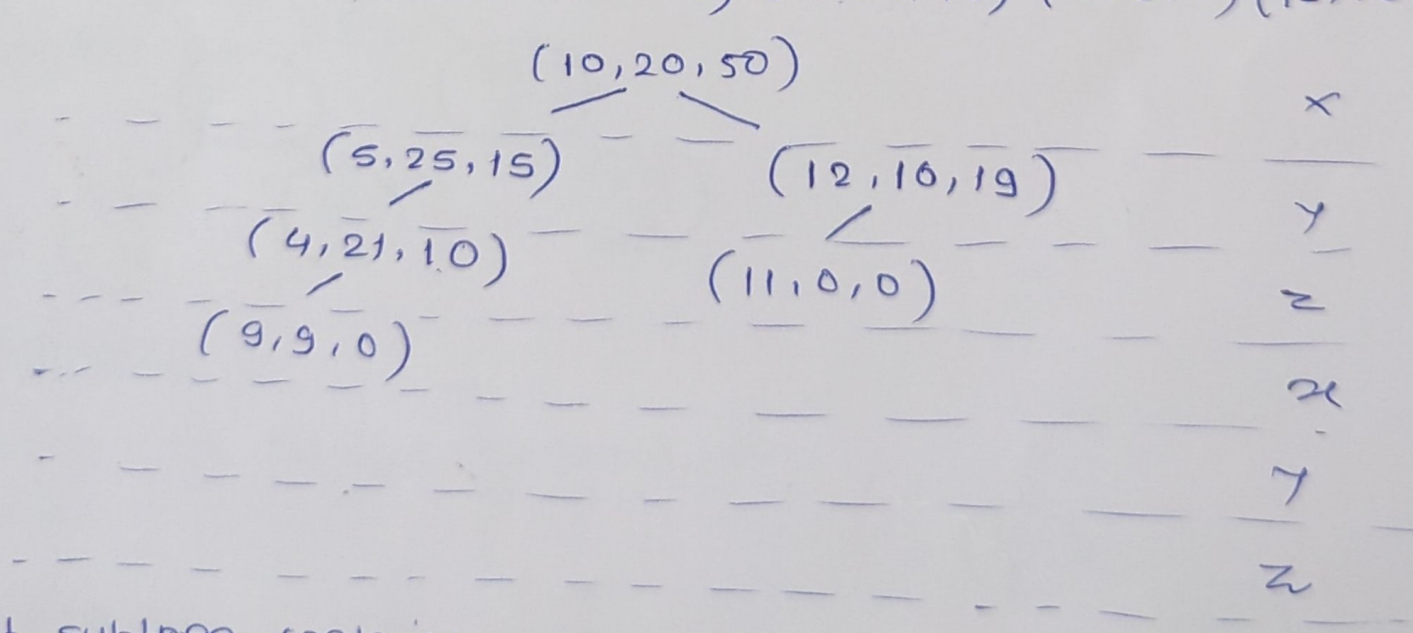
∴ 2D Tree

Data points: (20,40) (10,65) (15,100) (40,0) (12,200) (55,7)

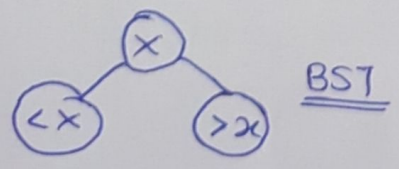


Ex-2 k=3

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



- 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

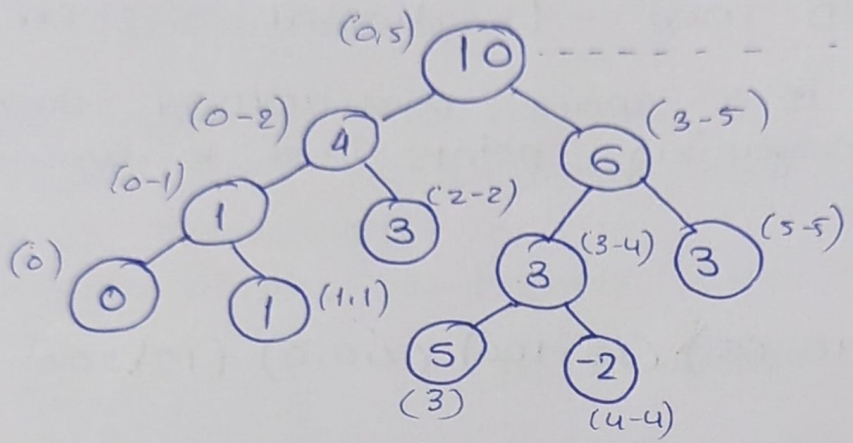
sum(0,2) = 0+1+3 = 4
sum(3,5) = 5-2+3 = 6
sum

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 → (0) 0 (1) 1 (2) 3 (3) 5 (4) -2 (5) 3



0 1 3 5 -2 3

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

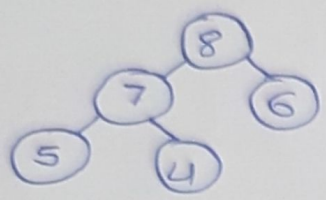
---wrong, no Null values are mentioned

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.

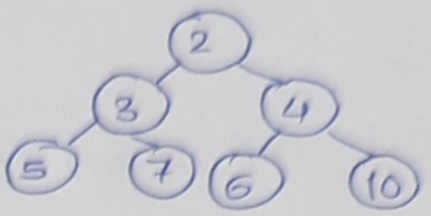
① Max Heap

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

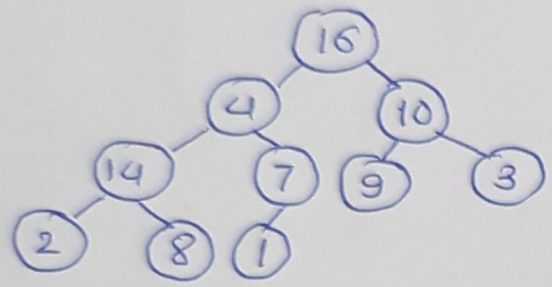


② Min Heap

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

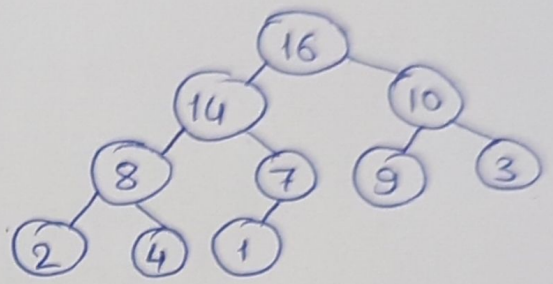


Example



Max-Heapify ( ) ⟶

TC : O(log n).

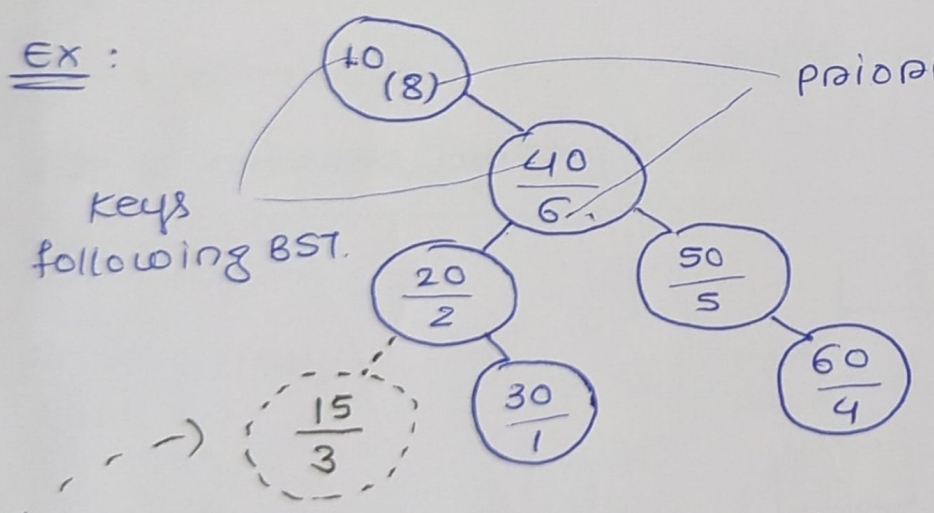


Treaps (Tree + Heap) unit 2 part

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

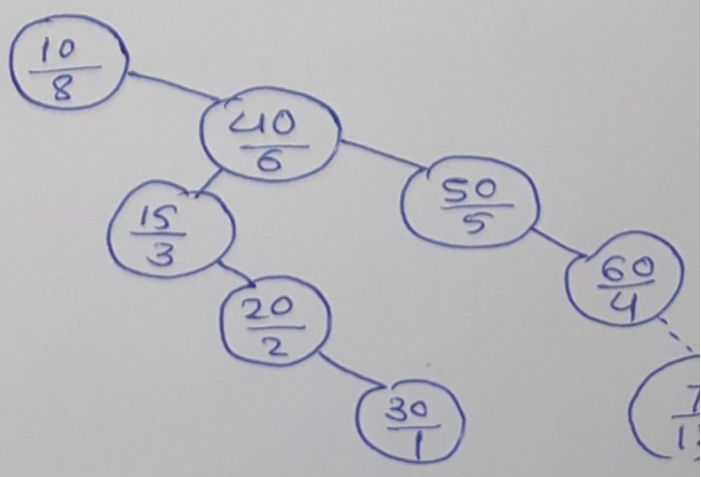
keys following BST.

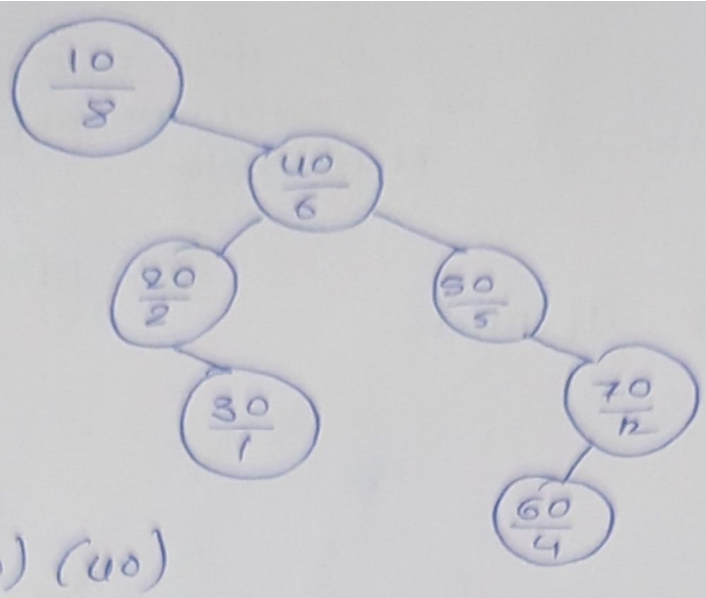
Operations

1) search ⟶ follows normal BST
2) Insert

Insert (15, 3)

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





solve it ....

① Insertion

② Deletion. (60) (20) (40)

3. Heap

① minHeap
② MaxHeap

* Insertion } O(logn)
  Deletion }

- Applications
① Huffman coding
② priority
③ scheduling
④ Memory managment →

In MM: ① Stack ② Heap

→ Stack: when local variables
Heap: Dynamic mem. eg (malloc)
Global variables

search - O(n)

optimal solution - O(n)

Applications:

- Huff

* LeftIst Heap n → rank = 1

Insert, delete : O(log n)
Merge /meld : O(log n)

or: if n → left = null
    if n → right = null

n → rank = 1 + min (n → left → rank, n → right → rank)
if n → left ≠ null & n → right = null

* properties:

The leftist Heap is characterised by that the shortest path on the left child is atleast as long as that on the rightside.

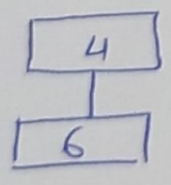
n → left → rank ≥ n → right → rank

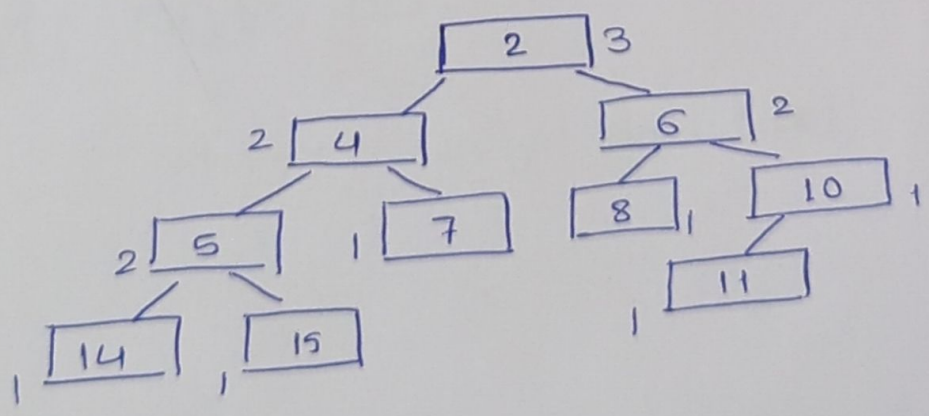
Not a complete binary tree

Insertion

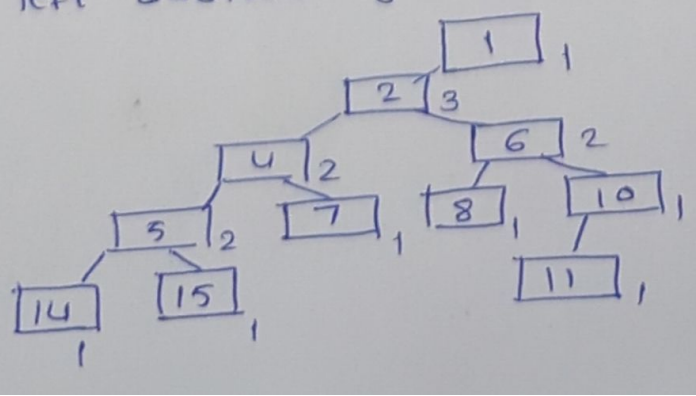
Insert: 1

- Empty Heap
rank = 0

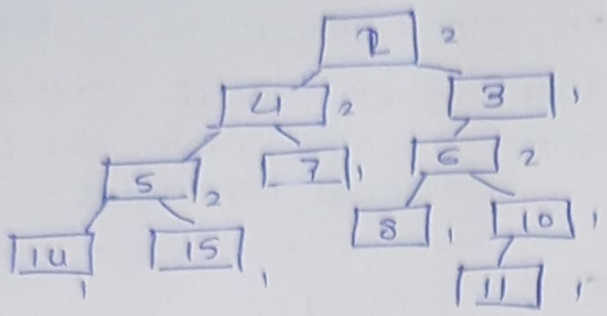




while adding newkey in Heap, if key < Heapkey then insert heap as left subtree of newkey (root)

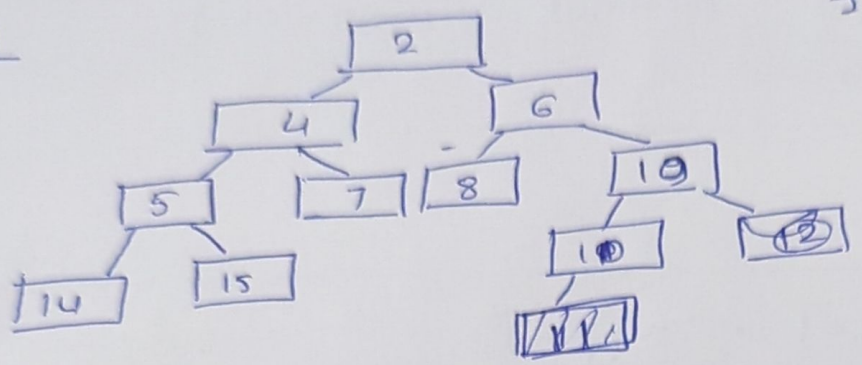


Insert element 3

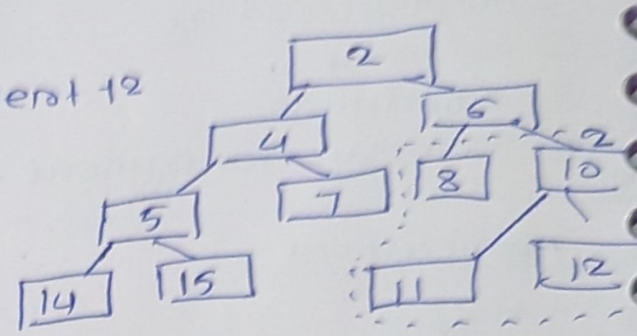


If newkey > Heap key (root) then search right side of heap & insert new key.

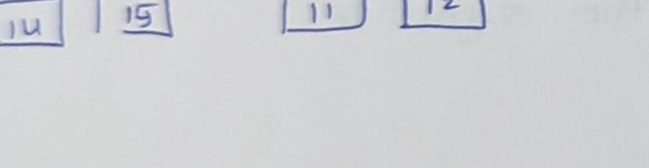
Insert 9



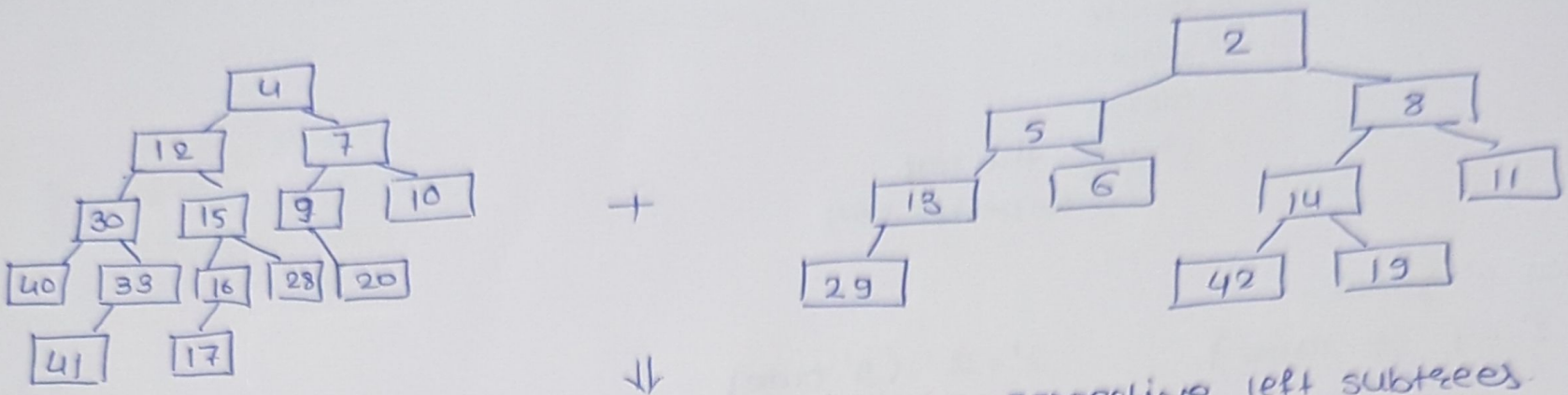
Insert 12



Here right sibling has rank greater than left one hence swap them.

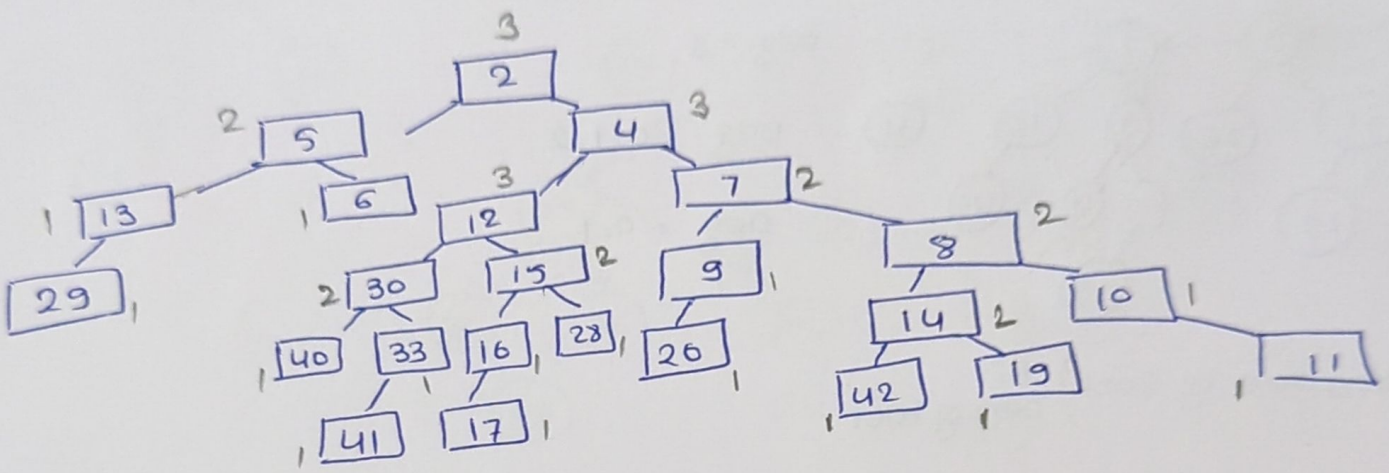


Merging property :

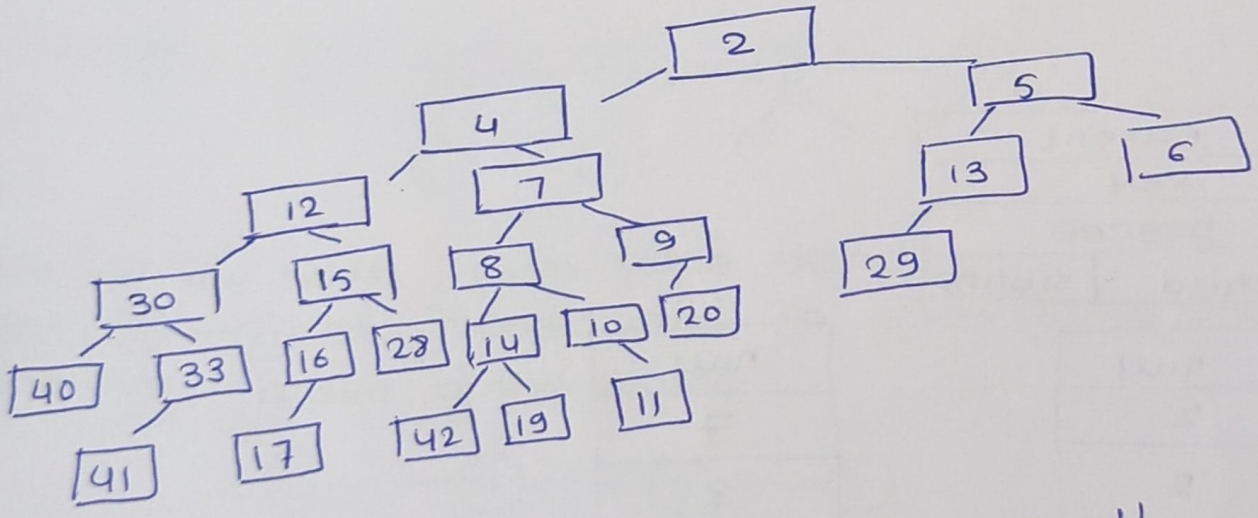


⇓

Do right side only then add these respective left subtrees



⇓



Yayyyy ! We got final tree!!

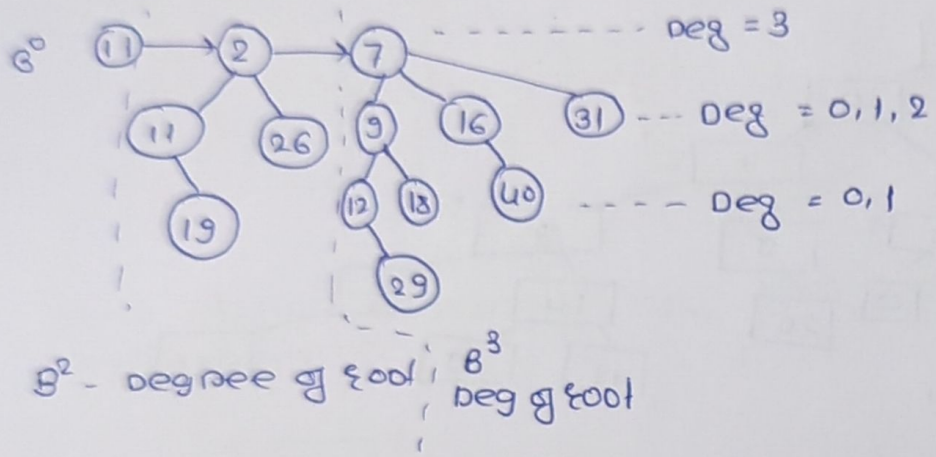
10/08/25

* Binomial Heap:
  - Insert
  - Delete
  - Find min
  - Merge
  - change the key
    - Decrease key

- Has 2^k nodes

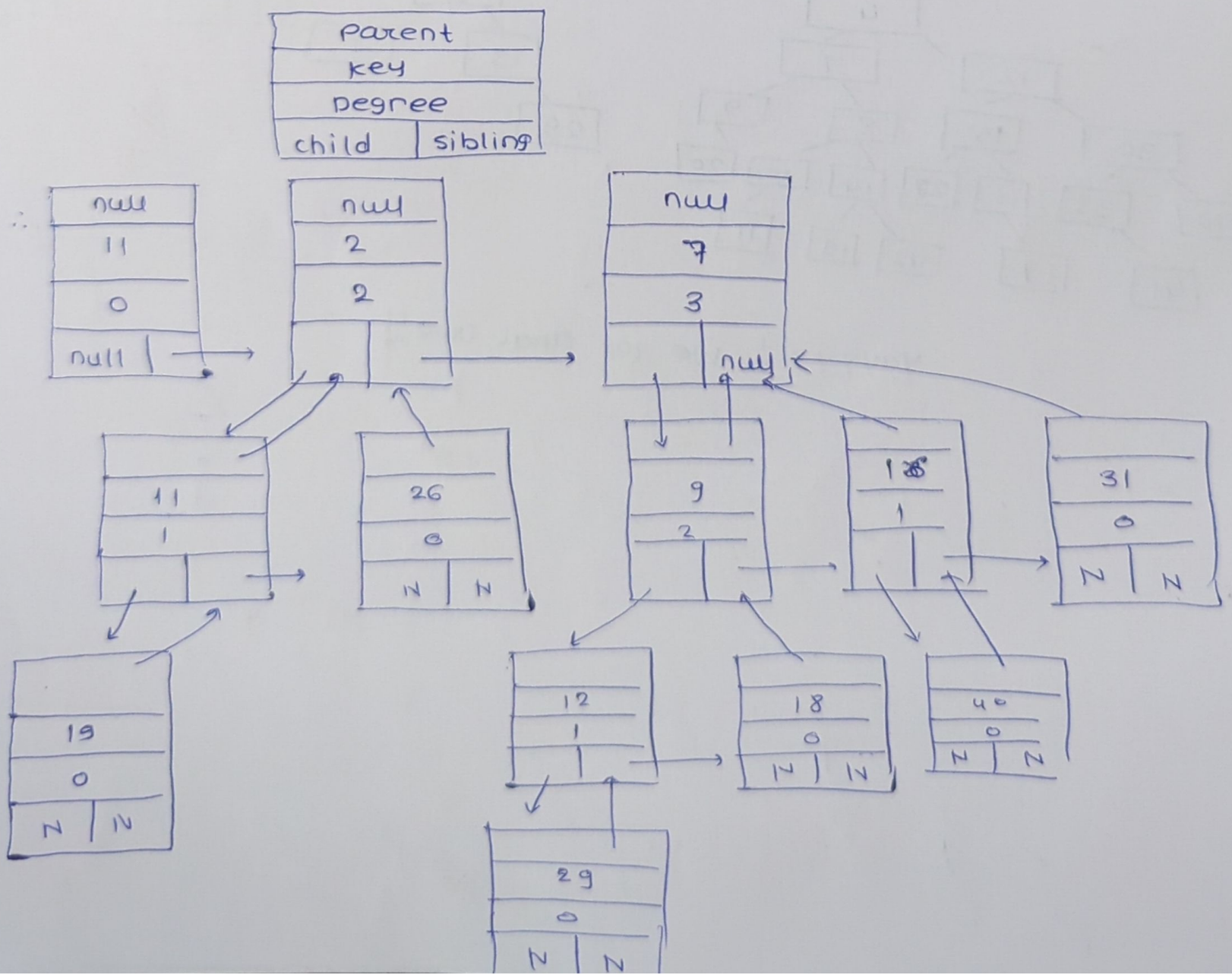
  $2^0 = 1$ (B-Tree) $2^1 = 2$ (B' tree) ---

* Follows min Heap property:



Roots are connected via singly linked list.

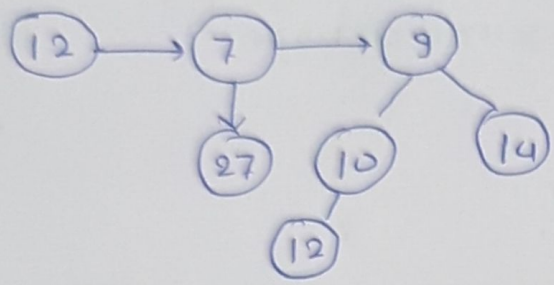
* Binomial node:



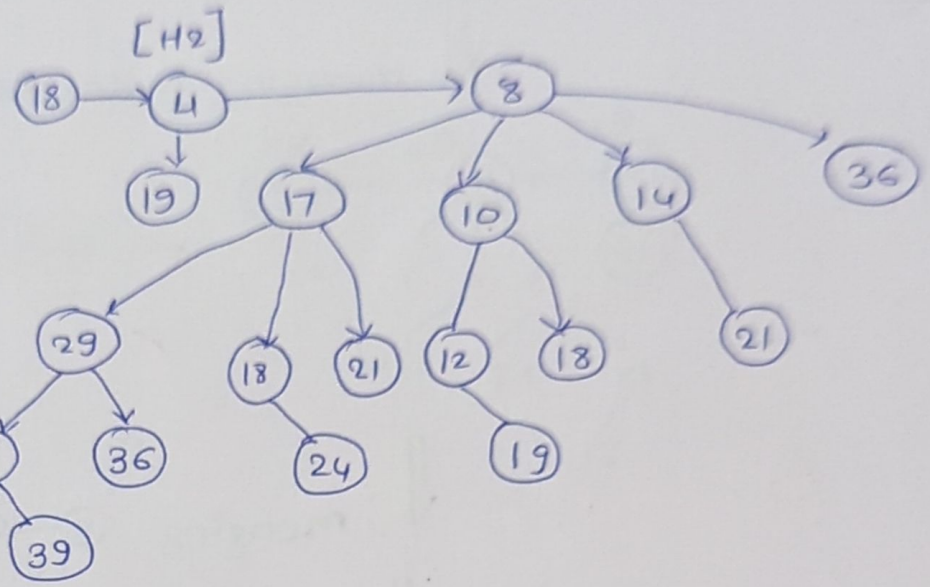
Heap operations

1) create B Heap
2) find minimum key (linear search LL)
3) Uniting two B Heaps
4) Insert Node
5) Extract Min key
6) Decrease key
7) Delete key

[H1]

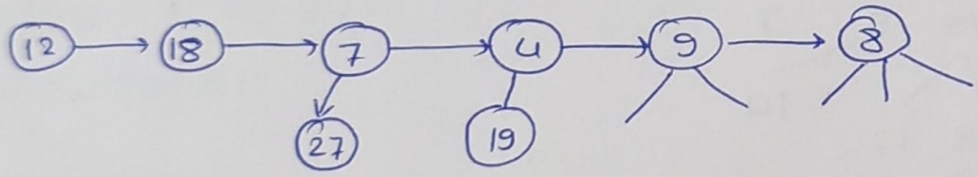


[H2]

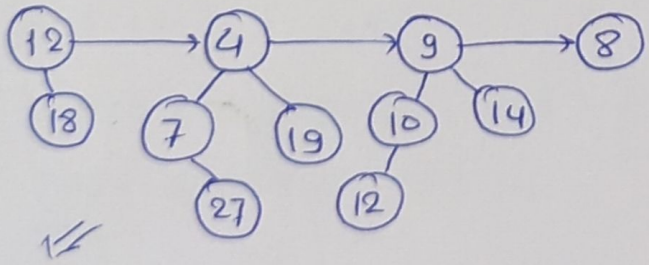


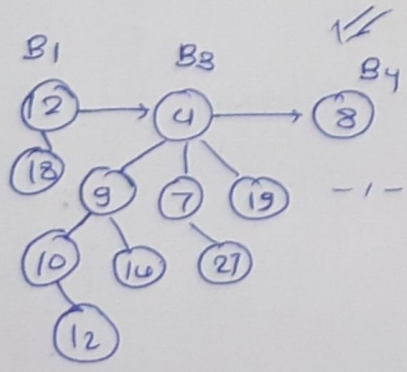
→ Merge H1 and H2 :

- merge the roots (i.e. LL) based on the degree (Ascending order)

∴ 

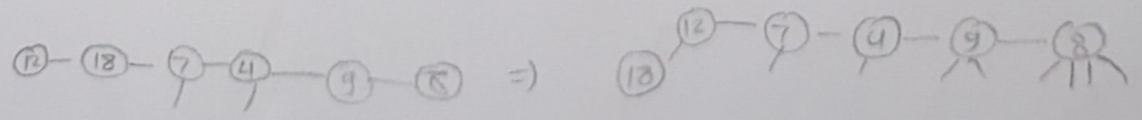
- combine the nodes with same degree.
  Here (7) → (4) are merged and not (12) → (7) to maintain the ascending order of the degree.

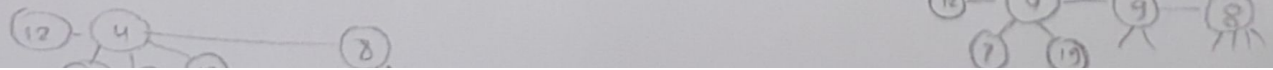




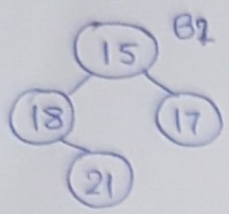
① Make the smallest node the root.
② create left child and add the other root.

merging heaps

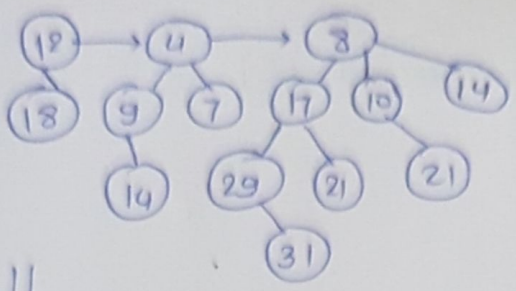




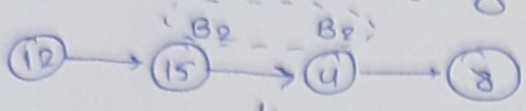
[H1]



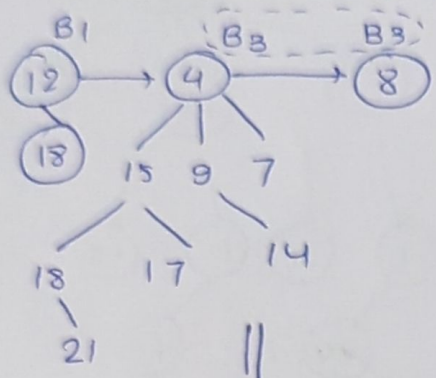
[H2]



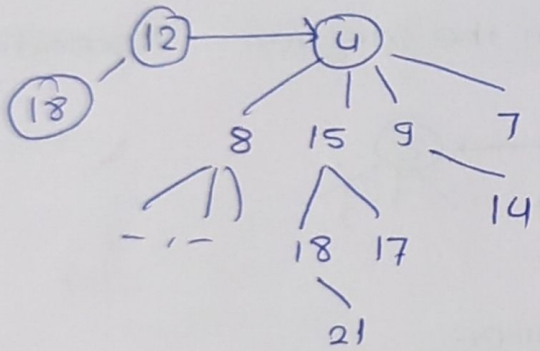
⇓ Merging trees in asc order



⇓ merging nodes withe same degree

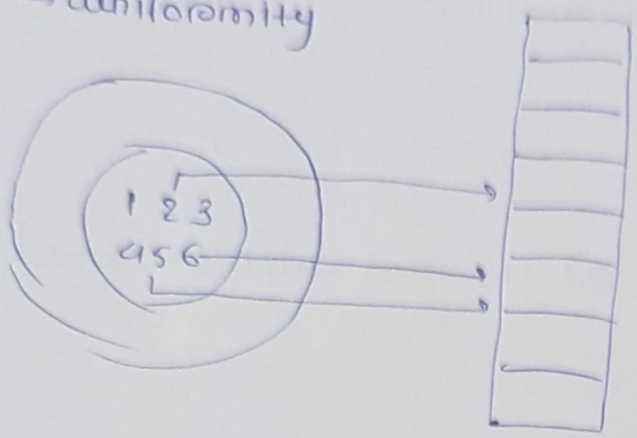


⇓ Merging (4)—(8)



Hash Table

Key → low cost
→ Determinism
→ uniformity



if same no
if no. pointing to same loc.
collism.

Hash funct^n

1) Division Method
2) Multiplicat^n Method
3) Midsquare Method
4) Folding Method.

① Division

$h(x) = x \bmod m$ m is prime NO.

key = 1234 m = 97

$= 1234 \bmod 97$

$h(1234) = 70$

② Multiplicat^n

constant A, $0 < A < 1$, size HT 1000, map key = 12345 let A = 0.6

$h(k) = m\,(kA \bmod 1)$

$h(12345) = 1000\,(12345 * 0.618033 \bmod 1)$

$= 1000\,(\underbrace{7629}_{\text{neglect}}.617385 \bmod 1)$

$= 1000\,(0.617385)$

$= 617.385$

$h(12345) \approx 617$

key = 1234

$(key)^2$ = 1522756 .... consider 3rd and 4th bit from right. & store at that loc.

= 27

at 27th loc.

④ Folding Method :

1234 = 12 & 34 $\quad$ 12+34 = 46 at 46 this will be stored

1234 = 12+34 = 46

12345 = 12+34+5 = 51 loc.

Collision : when same hash no. for 2 or more numbers.

* collision resolution Techniques

- open addressing / closed Hashing
  - → linear probing
  - → Quadratic probing
  - → Double Hashing
  - → Rehashing.
- chaining
  - ↳ Linking

LINEAR

① Insert the keys : consider a hash table of size 10

72, 27, 36, 24, 63, 81, 92

$h(k,i) = (h'(k) + i) \bmod m$ $\quad$ $h'(k) = k \bmod m$

m = 10

i = probe 0 to m-1

⟹

0	1	2	3	4	5	6	7	8	9
	81	72	63	24	92	36	27	101	
-1	-1	-1	- - -	-	-	-	-	-1	

Initially

$h(72,0) = ((72 \bmod 10)) \bmod 10$

$= 2 \bmod 10$

$= 2$

$h(92,0) = ((92 \bmod 10)) \bmod 10$

$= 2 \bmod 10$

$= 2$ — collision

Now inc i

$$h(99,1) = ((99 \bmod 10) + 1) \bmod 10$$
$$= 3$$

Again keep inc

$$\therefore h(99,3) = ((99 \bmod 10) + 3) \bmod 10 = 5 \bmod 10$$
$$= \boxed{5}$$

Ist may take more time since incrementing i....
There is problem of clustering

② Quadratic

$$h(k,i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$$
$c_1, c_2 \neq 0$

$m = 10$

$i = 0$ to $m-1$

consider $c_1 = 1$ and $c_2 = 3$

0	1	2	3	4	5	6	7	8
	81	72	63	24	92	36	27	

for key 72

$$h(72,0) = ((72 \bmod 10) + 1\times0 + 3\times0^2) \bmod 10$$
$$= 2 + 0 + 0 \bmod 10$$
$$= \boxed{2}$$

for 92

$$h(92,0) = ((92 \bmod 10) + 1\times0 + 3\times0^2) \bmod m$$
$$= (2 + 0 + 0) \bmod 10$$
$$=$$

16
3
48

2 + 12
16

5 + 75
80 82.

Double Hashing Method :

$$h(k,i) = (h_1(k) + i h_2(k)) \bmod m$$

$h_1(k) = k \bmod m$

$h_2(k) = k \bmod m'$

$m' < m$

if collision occur i will be inc. and this will be used.

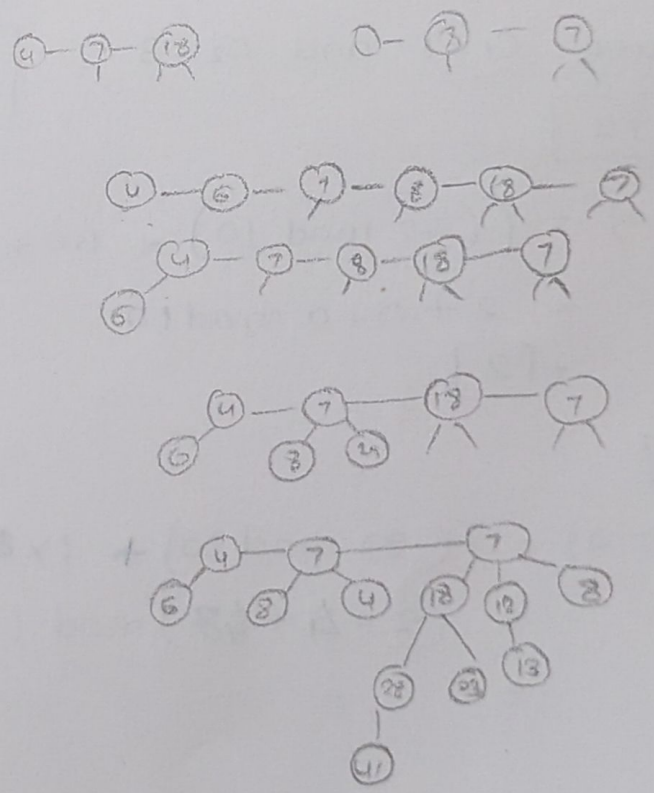
0	1	2	3	4	5	6	7	8	9
92	81	72	63	24		56	27		

$$h(92,1) = ((92 \bmod 10) + 1 \times (92 \bmod 8)) \bmod 10$$

$$= (2 + 2 \times 4) \bmod 10$$

$$= 10 \bmod 10$$

$$= \boxed{0}$$



ADS