Name - Harsh Shah
Class - SE Comp-1
ERP No : 59
Teams Sr No - 25

DSA ASSIGNMENT - 1

Q1) What is hashing ? Explain different methods of hash function calculations.

→ Hashing is an effective way to reduce the number of comparisons. Hashing deals with the idea of providing the direct address of the record where the record is likely to store.

Different methods of hash function calculations are -

1) Division Method - The hash function depends upon the remainder of division. Typically the divisor is table length.
Ex - If the record 54, 72, 89, 37 is to be placed in a hash table of size 10, then

Hash Function
h(key) = record % size

4 = 54 % 10
2 = 72 % 10
9 = 89 % 10
7 = 37 % 10

0	
1	
2	72
3	
4	54
5	
6	
7	37
8	
9	89

2) Multiplicative Hash Function - The multiplicative hash function works in following steps:

i) Multiply the key 'k' by a constant A where A is in the range $0 < A < 1$. The extract the fractional part of kA.

ii) Multiply this fractional part by m and take the floor.

Ex - Let $k = 107$, $m = 50$ and $A = 0.6180339$

$h(k) = \lfloor m * 107 * 0.6180339 \rfloor$

$= 66.12$

↓

0.12 Fractional part

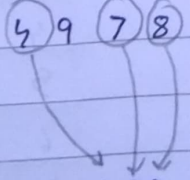
$\therefore h(k) = 50 \times 0.12$

$= 6$

$\therefore$ 107 will be placed at index 6.

3) Extraction - In this method, some digits are extracted from the key to form the address location in hash table.

Ex - Suppose 1st, 3rd and 4th digit from left is selected for hash key.



→ at location 478 in the hash table of 1000 the key is stored.

4) Mid Square - This method works in following steps
   i) Square the key
   ii) Extract middle part of result.

Ex - key = 3111

$\therefore (3111)^2$

↓

96 [783] 21

$\therefore$ For hash table of size 1000 ; H(3111) = 783

5) Folding - There are two folding techniques -

i) Fold shift - In this, the key is divided into separate parts. Then left and right parts are added to middle part.

ii) Fold boundary - In this, the leftmost and rightmost parts are folded on fixed boundary and added with middle part.

Ex -

key = 345678123

[345] 678 [123]

↓

345
+ 678
+ 123
(1)146

Discard ↵

$\therefore$ Index = 146

Fold shift

[345] 678 [123]

Reversed ↓ ↓ Reversed

543
+ 678
+ 321
(1)542

Discard ↑

$\therefore$ Index = 542

Fold boundary

6) Universal Hashing - It is a hashing technique in which randomized algorithm is used to select hash function at random, from family of hash functions with the help of certain mathematical properties. It guarantees lower number of collisions.

$U \rightarrow$ Universal key ; $H \rightarrow$ Finite collection of Hash function

$\therefore$ for $x, y \in U \ (x \neq y) \quad \{x \in H : h(x) = h(y)\} = \frac{|H|}{m}$

2) Construct hash table of size 15 and resolve collision using open addressing technique linear probing and use hash function $h(x) = x \bmod 15$.

35, 36, 25, 47, 2501, 129, 65, 29, 16, 14, 99

→

0	14
1	16
2	47
3	
4	
5	35
6	36
7	65
8	
9	129
10	25
11	2501
12	99
13	
14	29

35 % 15 = 5

36 % 15 = 6

25 % 15 = 10

47 % 15 = 2

2501 % 15 = 11

129 % 15 = 9

65 % 15 = 5 → collision

↓

6 → collision

↓

(7)

29 % 15 = 14

16 % 15 = 1

14 % 15 = 0 14 → collision

↓

0

99 % 15 = 9 → collision

↓

(12)

Name - Harsh Shah
Class - SE Comp-1
ERP No - 59
Teams Sr No - 25

DSA ASSIGNMENT - 2

1) What is binary tree ? Construct binary tree from the given traversals

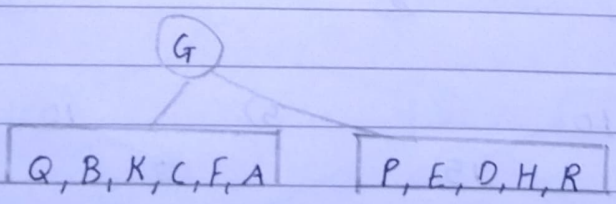
Pre - order - G, B, Q, A, C, K, F, P, D, E, R, H
In - order - Q, B, K, C, F, A, G, P, E, D, H, R.

→ A binary tree is a finite set of nodes which is either empty or consists of a root and two disjoint binary trees called the left - subtree and right sub - tree.

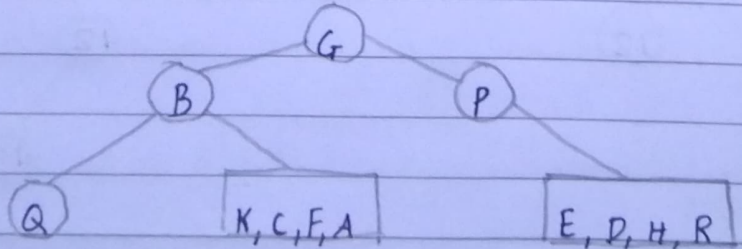
construction :

Preorder - Root, left, right
Inorder - Left, root, right

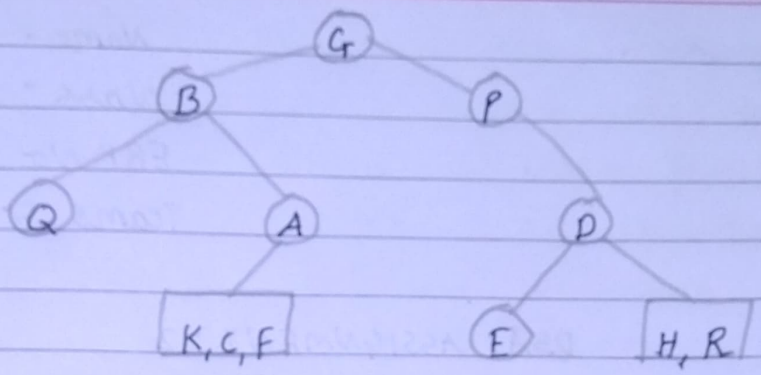
Step 1 -



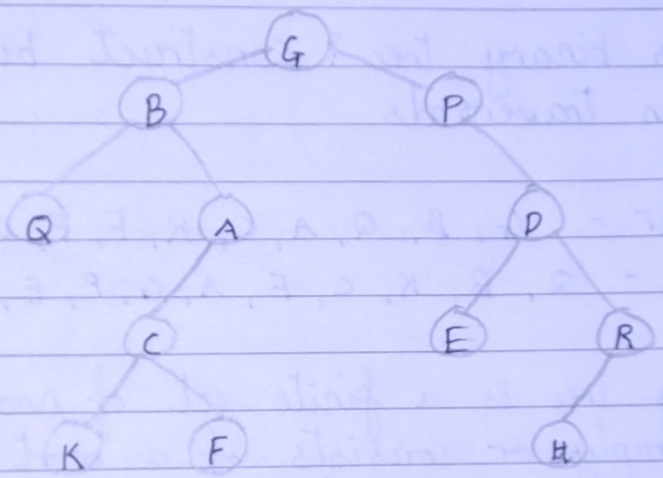
Step 2 -



step 3 -



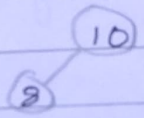
step 4 -

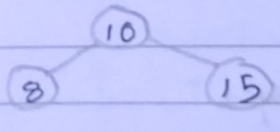


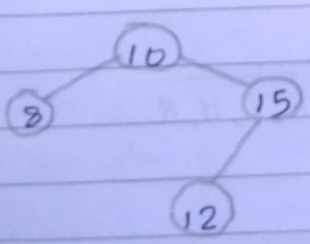
2) construct binary search tree for the following data:
10, 8, 15, 12, 13, 7, 9, 17, 20, 18, 4, 5

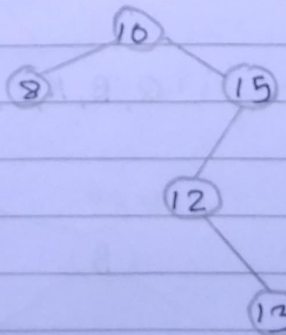
→ construction -

1) 

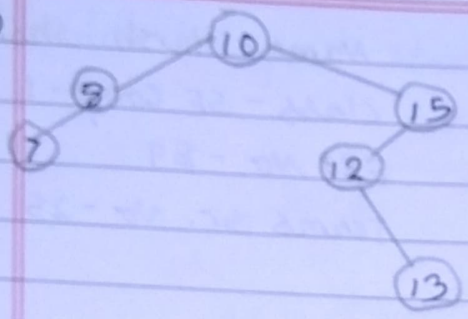
2) 

3) 

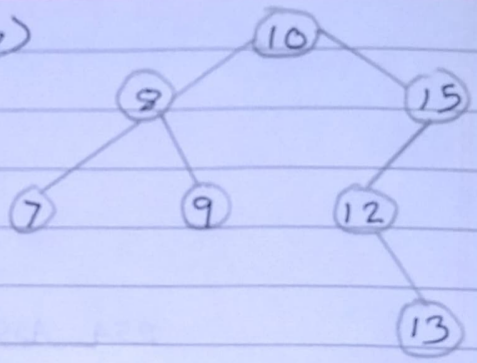
4) 

5) 

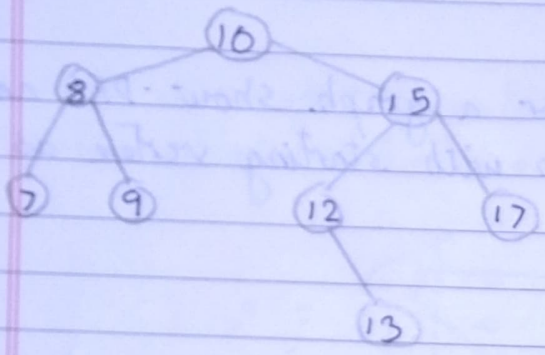
6)


10
8
15
7
12
13

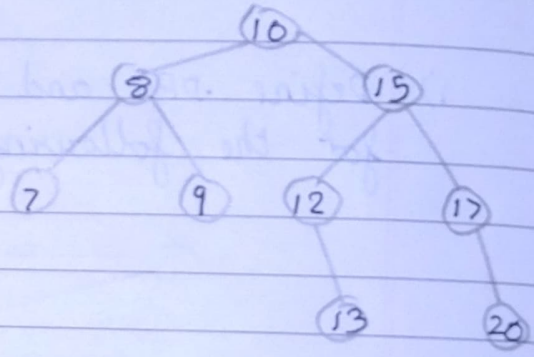
7)


10
8
15
7
9
12
13

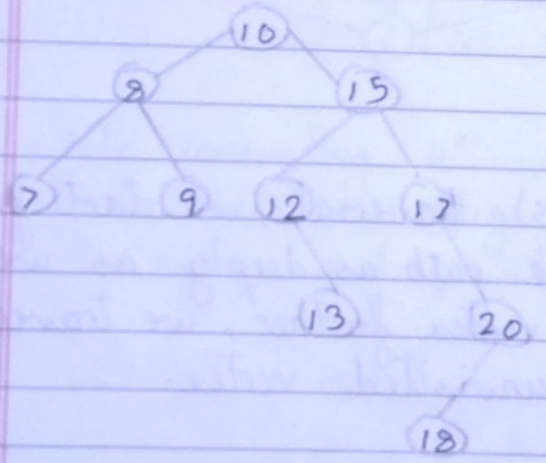
8)


10
8
15
7
9
12
17
13

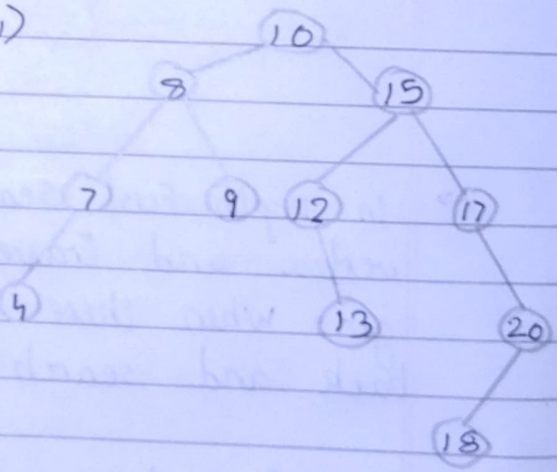
9)


10
8
15
7
9
12
17
13
20

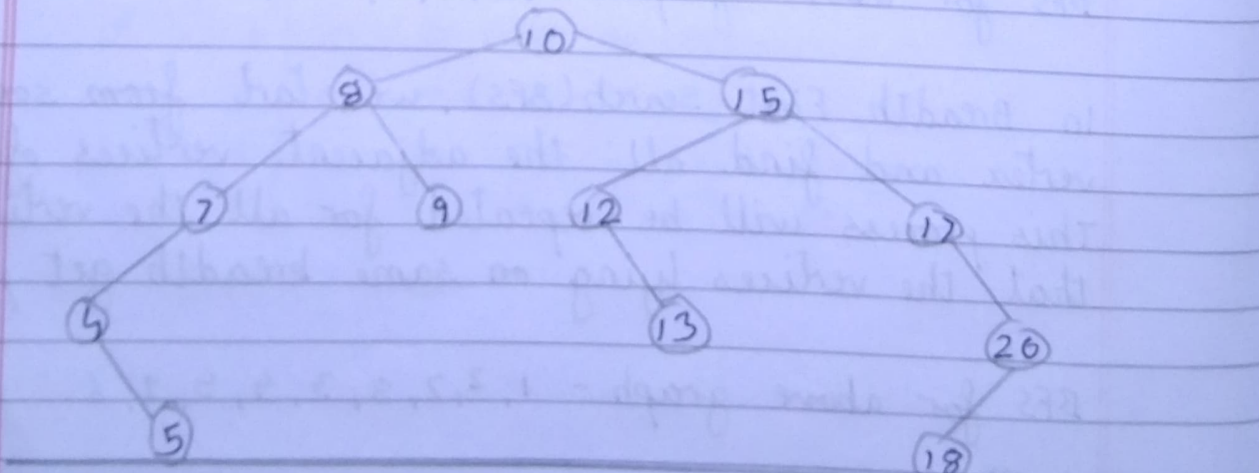
10)


10
8
15
7
9
12
17
13
20
18

11)


10
8
15
7
9
12
17
4
13
20
18

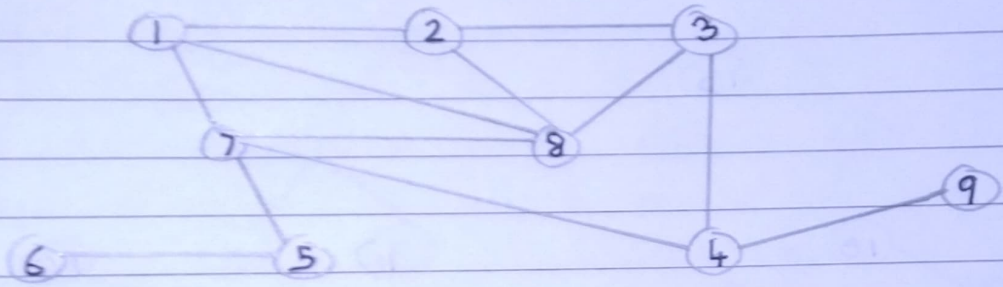
12)


10
8
15
7
9
12
17
4
13
20
5
18

Name - Harsh Shah
Class - SE Comp - 1
ERP No - 59
Teams Sr. No - 25

DSA ASSIGNMENT - 3

1) Define DFS and BFS for a graph. Show DFS and BFS for the following graph with starting vertex as 1.



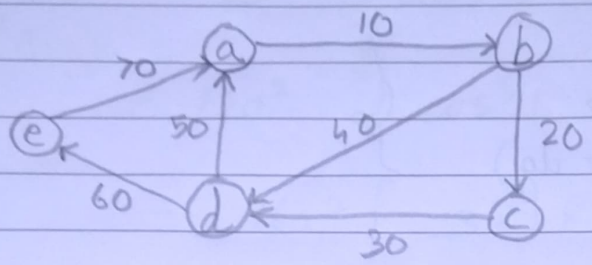
→ In Depth First Search (DFS) traversal, we start from one vertex and traverse the path as deeply as we can go. When there is no vertex further, we traverse back and search for unvisited vertex.

DFS for above graph - 1, 2, 3, 4, 7, 5, 6, 9, 8

In Breadth First Search (BFS), we start from some vertex and find all the adjacent vertices of it. This process will be repeated for all the vertices so that the vertices lying on same breadth get printed.

BFS for above graph - 1, 2, 7, 8, 3, 4, 5, 9, 6.

2) Solve all pair shortest path problem using diagraph.



→ First we will compute weighted matrix with no intermediate vertex i.e D^0

$$D^0 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 0 & 0 & \infty & \infty & \infty \\ \infty & 0 & 20 & 40 & \infty \\ \infty & \infty & 0 & 30 & \infty \\ 50 & \infty & \infty & 0 & 60 \\ 70 & \infty & \infty & \infty & 0 \end{bmatrix} \end{array}$$

while computing D^k, in each matrix D^k is the shortest distance dig has to be computed between vertices v_i and v_j where intermediate vertex is k

$$D[i,j] = \min [D[i,j] , D[i,k] + D[k,j]]$$

$$D^1 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 0 & 10 & \infty & \infty & \infty \\ \infty & 0 & 20 & 40 & \infty \\ \infty & \infty & 0 & 30 & \infty \\ 50 & 60 & \infty & 0 & 60 \\ 70 & 80 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

PRABHAT
PAGE NO. :
DATE : / /

$D^2 =$

	a	b	c	d	e
a	0	10	30	60	∞
b	∞	0	20	40	∞
c	∞	∞	0	30	∞
d	50	60	80	0	60
e	70	80	100	120	0

$D^3 =$

	a	b	c	d	e
a	0	10	30	60	∞
b	∞	0	20	40	∞
c	∞	∞	0	30	∞
d	50	60	80	0	60
e	70	80	100	120	0

$D^4 =$

	a	b	c	d	e
a	0	10	30	60	120
b	∞	0	20	40	100
c	∞	90	0	30	90
d	50	60	80	0	60
e	70	80	100	120	0

∴ The above matrix shows all pair shortest path.

PRABHAT
PAGE NO. :
DATE : / /

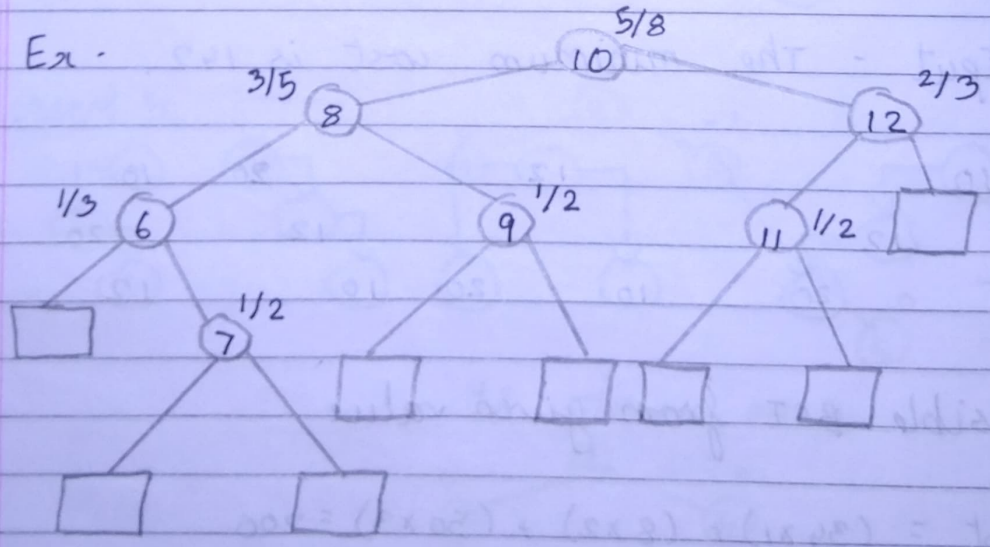
Name - Harsh Shah
Class - SE Comp - 1
ERP No - 59
Teams Sr No - 25

DSA ASSIGNMENT - 4.

1) Define weight balanced tree and optimal binary search tree with example.

→ - The weight balance tree is a binary search tree, whose balance is based on the sizes of the subtrees, in each node.
- It is a binary tree in which the number of nodes in the left subtree is atleast half and at most twice the number of nodes in right subtree.
- Balance factor at each node = $\frac{\text{no. of ext. node in left subtree}}{\text{Tot no. of ext. node of tree}}$

Ex.



- The depth of weight balance tree is O(log n)

PRABHAT
PAGE NO. :
DATE : / /

Optimal Binary Search Tree (OBST)

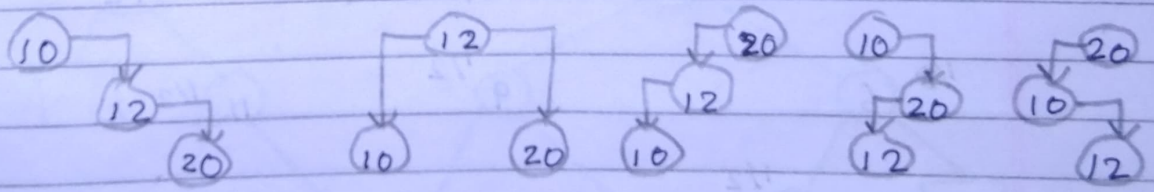
- The element having more probability of appearance should be placed nearer to the root of binary search tree and the element that appear for least no. of time i.e the element with lesser probability should be placed away from roots.
- The BST created with such kind of arrangement is called OBST.

Ex -

- A set of integer are given in sorted order and another array 'freq' for frequency count. Our task is to find minimum cost of all searches.
- An auxiliary array cost[n] is created to solve and store solution of sub programs. cost matrix will hold data to solve problem in bottom up.

Input - key value as node and freq.
key = {10, 12, 20} freq = {34, 8, 50}

output - The minimum cost is 142.

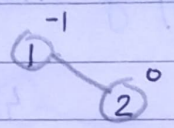


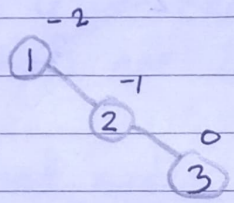
Possible BST from given value

a) cost = (34 x 1) + (8 x 2) + (50 x 3) = 206
b) cost = (8 x 1) + (34 x 2) + (50 x 3) = 176
c) cost = (50 x 1) + (34 x 2) + (8 x 3) = 142 (min).

2) Construct an AVL tree by inserting numbers from 1 to 8.

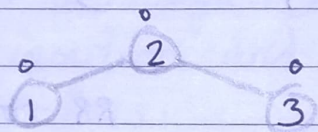
→ Insert 1 → 

Insert 2 → 

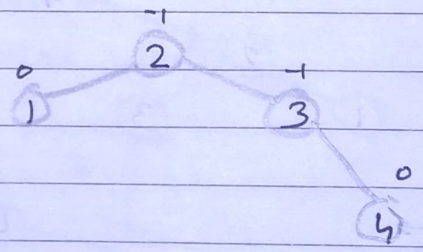
Insert 3 → 

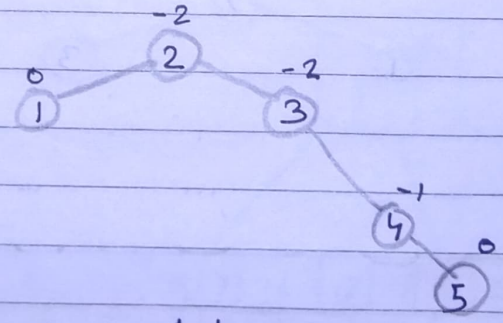
→ unbalanced tree

Apply RR rotation



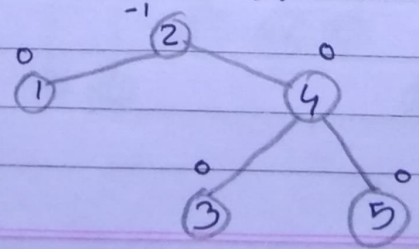
balanced

Insert 4 → 

Insert 5 → 

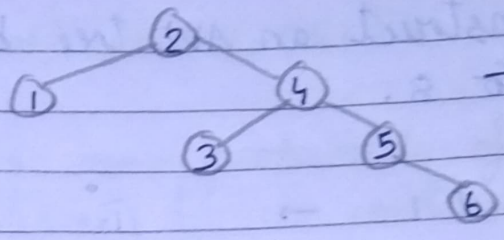
→ Unbalanced

RR rotation on 3



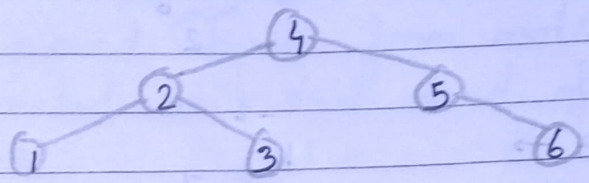
→ balanced

Insert 6 →

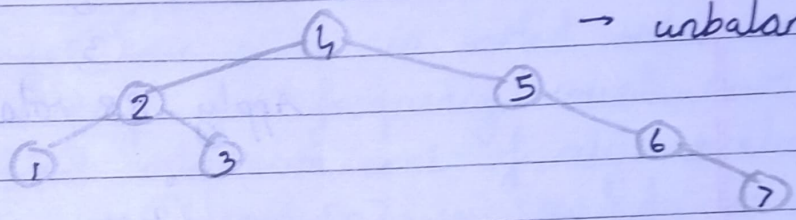


→ unbalanced

Apply RR rotation on 2

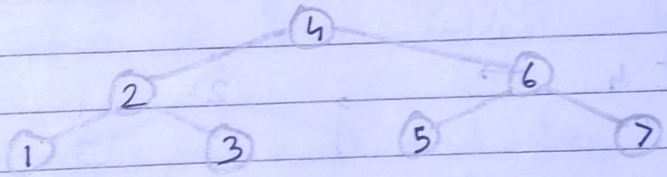


Insert 7 →



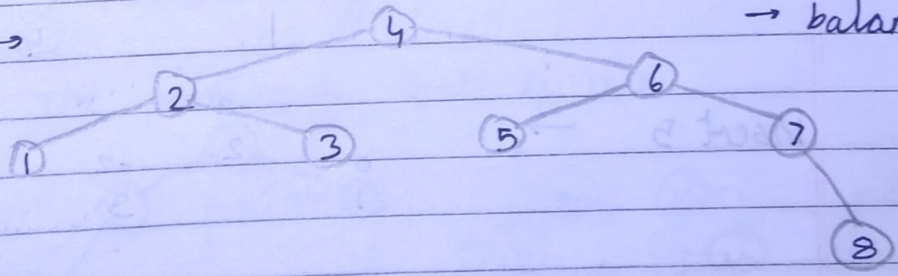
→ unbalanced

RR rotation on 5



→ balanced

Insert 8 →



→ balanced

PRABHAT
PAGE NO. :
DATE : / /

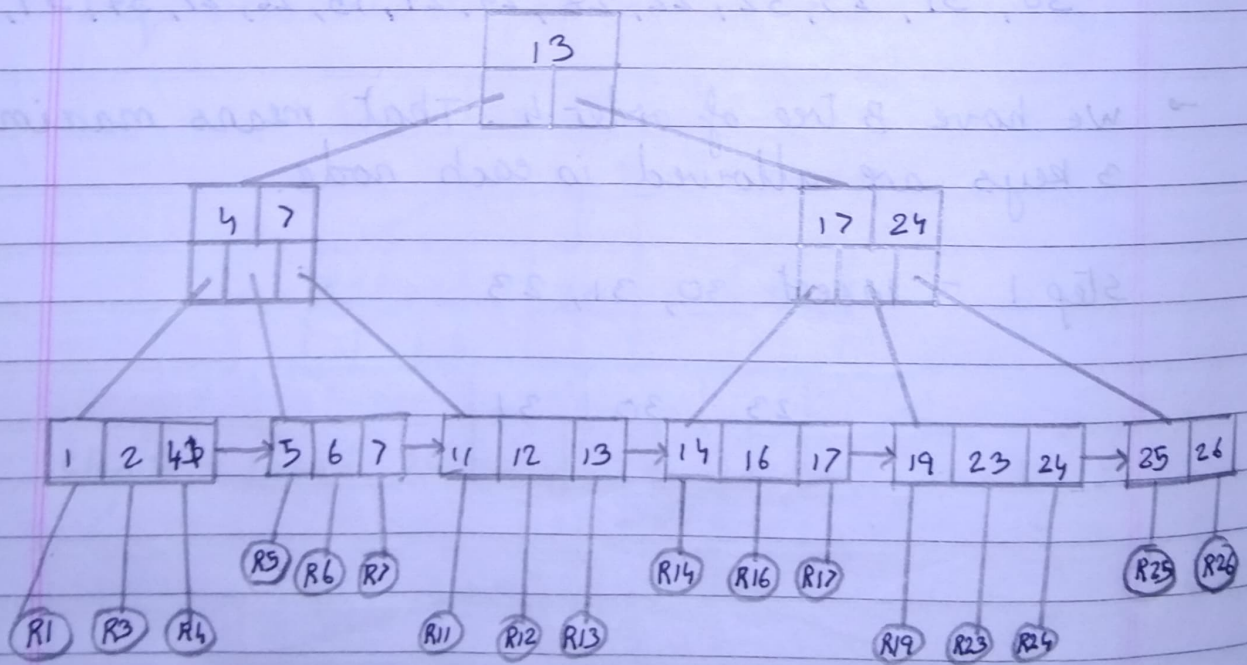
Name - Harsh Shah
Class - SE Comp - 1
ERP No - 59
Teams Sr No - 25

DSA ASSIGNMENT - 5

1) What is B+ tree? Give the structure of its internal node. What are the variations of B tree?

→ In B+ tree, from leaf nodes, reference to any other node can be possible. The leaves in B+ tree form a linked list which is useful in scanning the nodes sequentially. The insertion and deletion operations are similar to B-trees.

Structure of internal nodes.



- From leaf node only, any key can be accessed of entire tree. There is no need to traverse the tree in inorder fashion. Thus B+ trees gives faster access to any key.

There are two variants of B-trees:

1) The B+ tree - It is a B-tree in which data is stored only in the leaf nodes due to which efficient data access is possible.

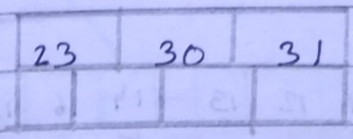
2) The B* tree - It is a B-tree in which each node except root node is at least 2/3 full rather than just half full.

---

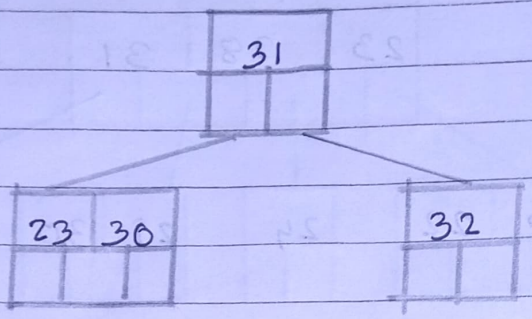
2) Build B tree of order 4 for following data:
30, 31, 23, 32, 22, 28, 24, 29, 15, 26, 27, 34, 39, 36

→ We have B tree of order 4. That means maximum 3 keys are allowed in each node.

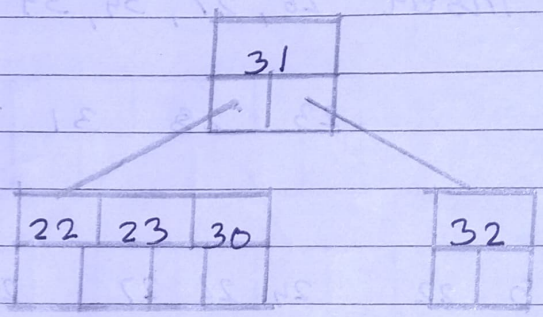
Step 1 - Insert 30, 31, 23



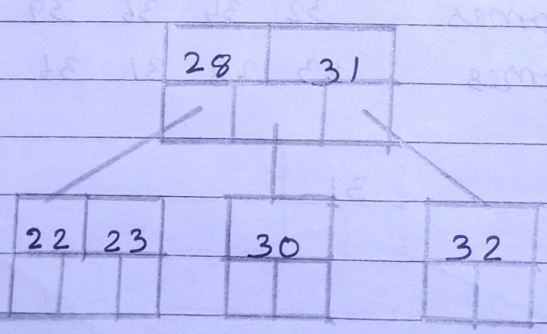
Step 2 - Insert 32
Seq. becomes 23 30 31 32 ; 31 push up



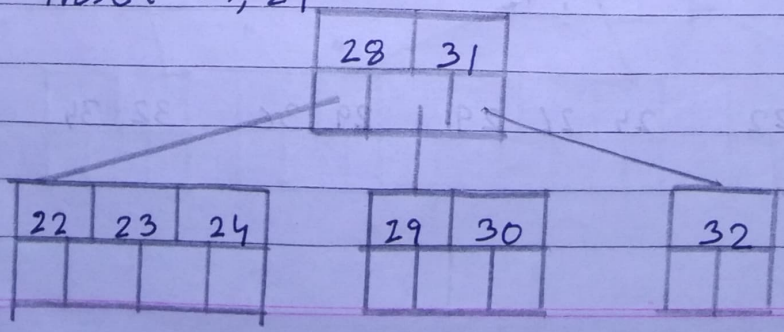
Step 3 -
Insert 22



Step 4 - Insert 28 push up
Seq. becomes 22 23 28 30

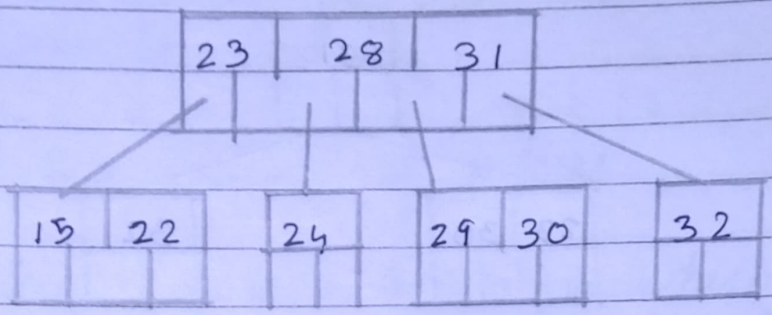


Step 5 - Insert 24, 29

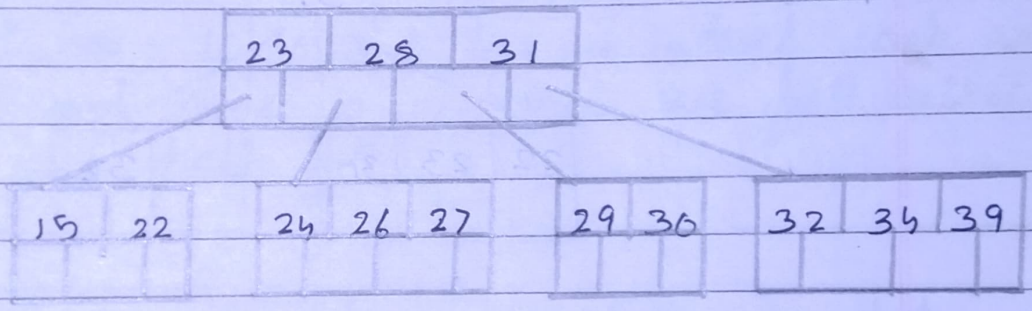


PRABHAT
PAGE NO. :
DATE : / /

Step 6 - Insert 15
seq. becomes 15 22 23 24 23 push up



Step 7 - Insert 26, 27, 34, 39



Step 8 - Insert 36
seq. becomes 32 34 36 39 36 push up
seq becomes 23 28 31 36 31 push up