## Tree Traversal

To state info we required ordered rooted tree and some action procedure for visiting each vertex. The process of visiting and computation at each vertex called searching, tree search, walking, traversing

In ordered rooted tree the listing of vertices used to represent various types of arithmetic expressions

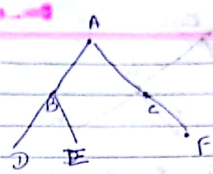
There are three recursive methods (traversal algo) are available for searching (traversing) a positional binary tree / ordered rooted tree
1. Preorder traversal / search
2. Inorder
3. Postorder

I Preorder search -
visit root, search left subtree $T_1$ in preorder, search right subtree $T_2$ in preorder


II Inorder search -
search $T_1$ in preorder, visit root, search $T_2$ in preorder

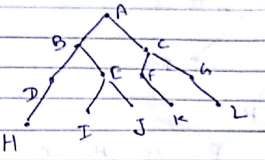
III Postorder -
search $T_1$ in postorder, search $T_2$ in postorder, visit root

Qu



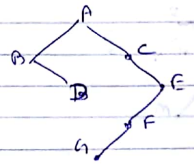

| | | | | | | |
|---|---|---|---|---|---|---|
| Pre | A | B | D | E | C | F |
| In | D | B | E | A | C | F |
| Post | D | E | B | F | C | A |

Qu



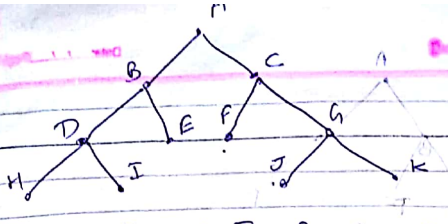

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Pre | A | B | D | H | E | I | J | C | F | K | G | L |
| In | H | D | B | I | E | J | A | F | K | C | G | L |
| Post | H | D | I | J | E | B | K | F | L | G | C | A |

Qu



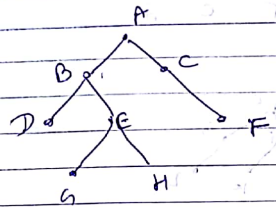

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Pre | A | B | D | C | E | F | G |
| In | B | D | A | C | G | F | E |
| Post | D | B | G | F | E | C | A |

Qu



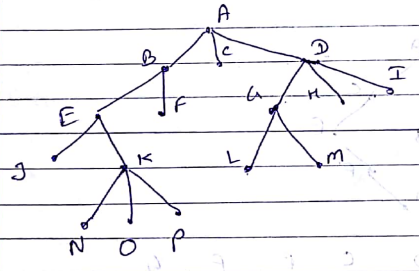

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Pre | A | B | D | H | I | E | C | F | G | J | K |
| In | H | D | I | B | E | A | F | C | J | G | K |
| Post | H | I | D | E | B | F | J | K | G | C | A |

Qu.



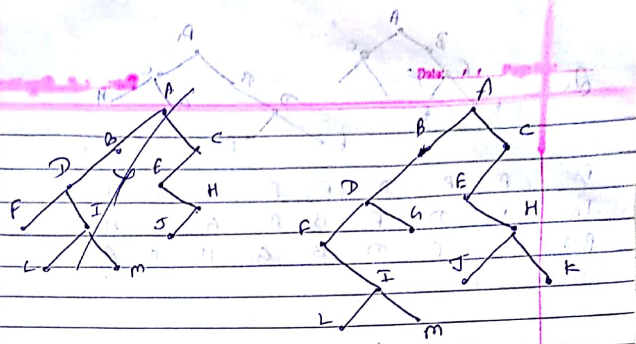

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Pre | A | B | D | E | G | H | C | F |
| In | D | B | G | E | H | A | C | F |
| Post | D | G | H | E | B | F | G | A |

Qu



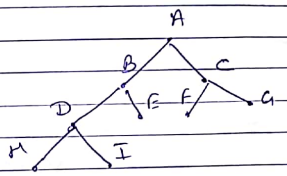

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Pre | A | B | E | J | K | N | O | P | F | G | D | G | L | M | H I |
| In | J | E | N | K | O | P | B | F | A | C | L | G | M | D | H I |
| Post | J | N | O | P | K | E | F | B | C | L | M | G | H | I | D A |

Qu



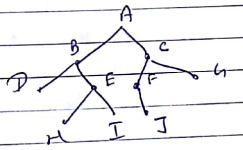

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Pre | A | B | D | F | I | L | M | G | C | E | H | J | K |
| In | F | L | I | M | D | G | B | A | E | J | H | K | C |
| Post | L | M | I | F | G | D | B | J | K | H | E | C | A |

Qu



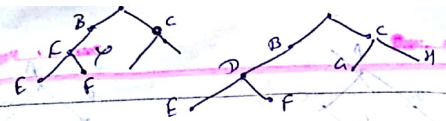

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Pre | A | B | D | H | I | E | C | F | G |
| In | H | D | I | B | E | A | F | C | G |
| Post | H | I | D | E | B | F | G | C | A |

Qu




| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Pre | A | B | D | E | H | I | C | F | J | G |
| In | D | B | H | E | I | A | J | F | C | G |
| Post | D | H | I | E | B | J | F | G | C | A |

Q.




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Pre | A | B | D | E | F | C | G | H |
| In | E | D | F | B | A | G | C | H |
| Post | E | F | D | B | G | H | C | A |

Binary Search tree – / Ordered or sorted binary tree

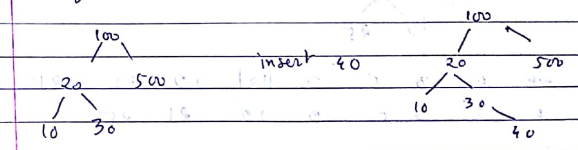
It formulate the problem of searching for an element in an ordered list. and provide a sequence of comparison. This comparison continues for successive branch nodes until value is reached.
Generally the problem of searching done in an ordered list.

A binary search tree is a node-based binary tree, where every vertex v satisy following properties

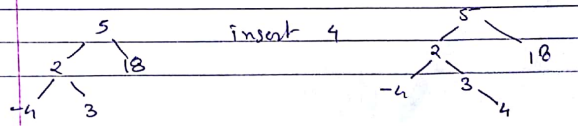
1) v > for every vertex in its left subtree.
2) v ≤ (x) – " – right "
3) Left & right subtree each must also be a binary search tree.
(There must be no duplicate nodes)

Insertion of a key –
A new key is also inserted at leaf. we start searching a key from root till we hit a leaf node. Once a leaf node is found, the new node is added as a child of the leaf node

a)



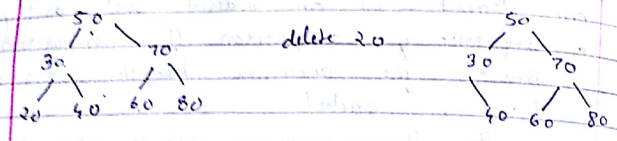

b)



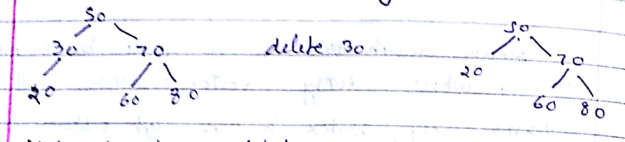
Deletion of a key –

When we delete a node, 3 possibilities arise

1). Node to be deleted is leaf

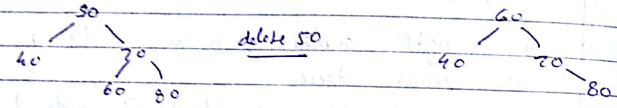



2). Node to be deleted has only one child




3) Node to be deleted has two children

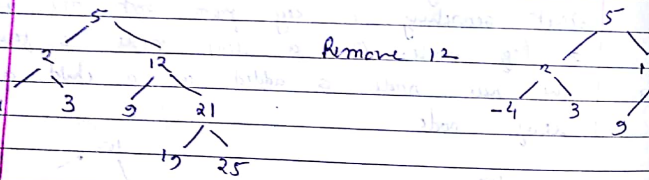
a)




Inorder 40 [50] 60 70 80

After 40 [60] 70 80

b)




Inorder -4 2 3 5 9 [12] 19 21 25

After -4 2 3 5 9 19 21 25

Ques Make binary search tree rooted with a vertex labeled with 42.

Inorder is 24 35 37 [42] 46 52 62 79