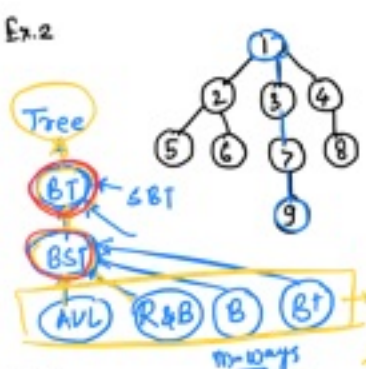
Ex1

Degree (tree) = 2

0, 1, 2 ⇒ BT

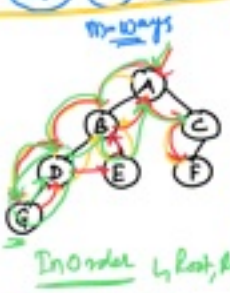
Ex.2



Incomplete BT ?

C F S In

Ex.3



— BT

Preorder Root, Left

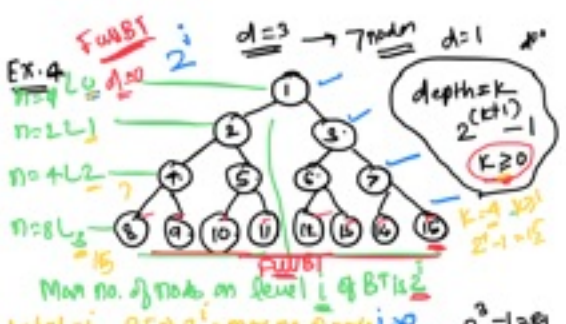
A B D G E C F

Postorder L, R, Root

Inorder L, Root, R G D E B F C A

G D B E A F C

Ex.4



Max no. of nodes on level i of BT is 2^i

level = i, BT ⇒ 2^i = max no. of nodes, i≥0

i≥1, BT ⇒ 2^{i-1}

depth=k, 2^k-1, k≥0

depth=k, $(2^{k+1}-1)$ k≥0

Ex.5

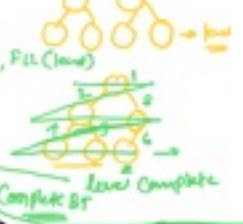
— BT — (0,1,2)

— Complete BT → BT, SBT, FullC (level)

— Full BT (level)

— strict BT (0 & 2)

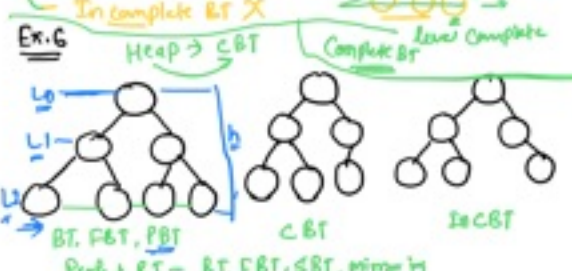
— Incomplete BT ✗



Ex.6

Heap → CBT

Complete BT



BT, FBT, PBT — CBT — SCBT

Perfect BT — BT, FBT, SBT, mirror

→ Total no. of nodes in perfect BT $= 2^{h+1}-1$, h≥0

nodes — (n)

height — (h)

→ Number of leaves (height) = 2^h

→ Number of non-leaf nodes = 2^{h+1}

Array Representation

Relation

Parent = i

Left child = 2i

Right child = 2i+1

index = 1: 2i+1, 2i+2

index = 0: (i−1)/2 → check

Traversal

1. Inorder
2. Preorder
3. Postorder

Expression Tree

Equation — a+b*c+d*e → precedence → stack

Rules →

Node → Operator

Leaf node → Variables / Operand

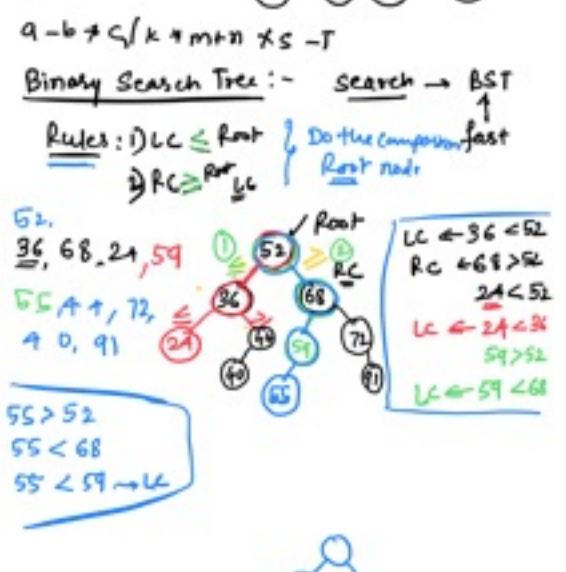
a − b * c / k * m + n * s − T

Binary Search Tree :- search → BST → fast

Rules : 1) LC ≤ Root

2) RC ≥ Root, LC

Do the comparison Root node



52, 36, 68, 24, 59

55 ← 4, 72, 40, 91

LC ← 36 < 52

RC ← 68 > 52

24 < 52

LC ← 24 < 36

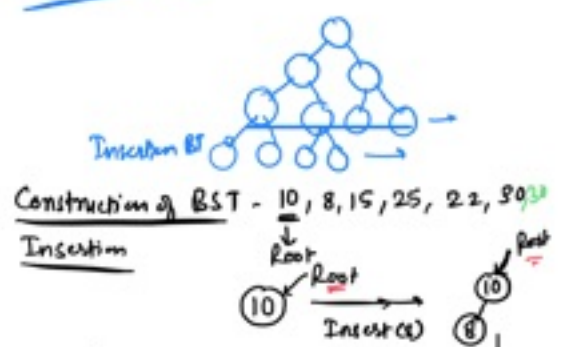
59 > 52

LC ← 59 < 68

55 > 52

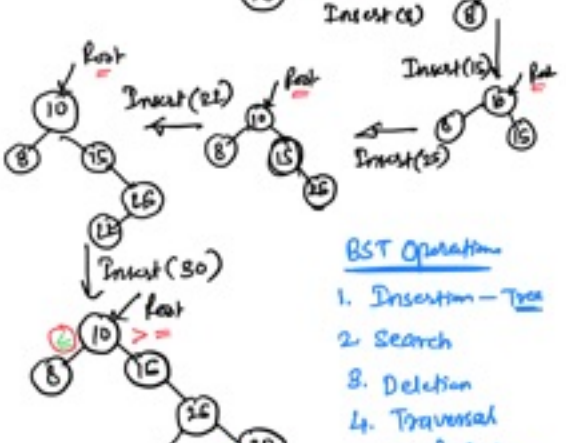
55 < 68

55 < 59 → LC



Construction of BST - 10, 8, 15, 25, 22, 30

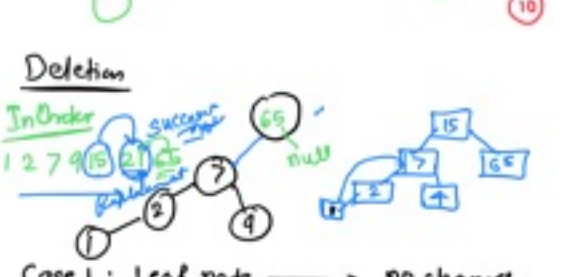
Insertion



BST Operations

1. Insertion — Tree
2. Search
3. Deletion
4. Traversal
   - Preorder
   - Postorder
   - Inorder

Deletion

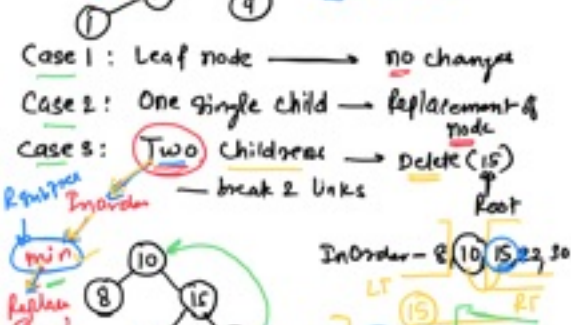


Case 1 : Leaf node → no change

Case 2 : One single child → Replacement of node

Case 3 : Two Children → Delete (15) — break 2 links

Root

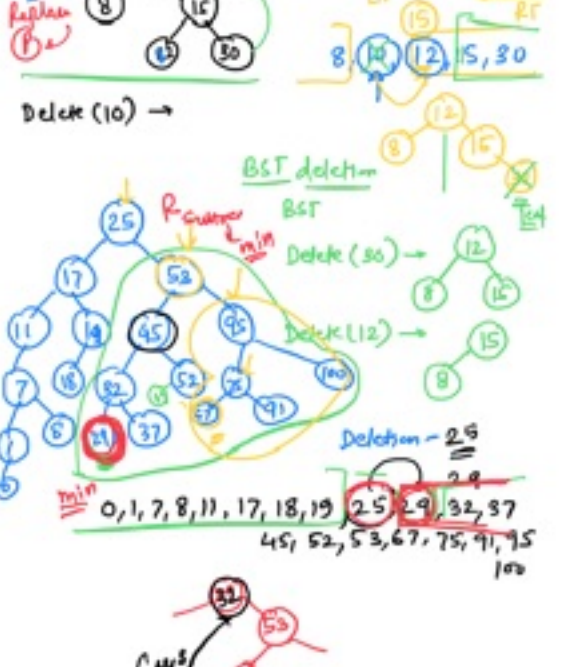


Inorder - 8, 10, 15, 22, 30

Delete (10) →

BST deletion

BST



0, 1, 7, 8, 11, 17, 18, 19, 25, 29, 32, 37, 45, 52, 53, 67, 75, 91, 95, 100

Case 1

Case 2

Expression Conversion -

a + b

Operands Operator Operands

Infix : a + b

prefix : + a b

postfix : a b +

Follow the precedence

A + B * C / D − E ^ F * G

Precedence

Unary operator : −, +, NOT

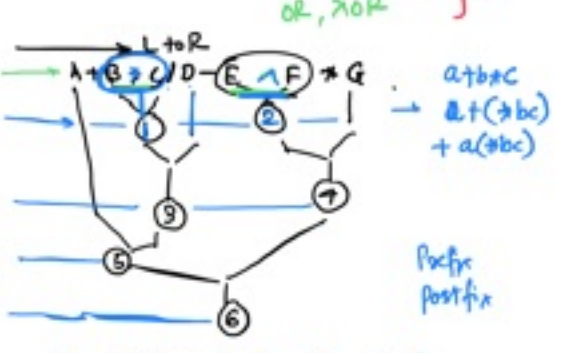
Exponential : ^ → Right to Left

Associativity

* /

+ −

<, <=, >, >=, AND, OR, XOR → Left to Right



a+b*c

a+(*bc)

+a(*bc)

Prefix

Postfix

((A+B)*((C/D)−(E^(F*G))))

Prefix —

A+B*C

= A + (*BC)

= +A(*BC)

= +A*BC

A*B+C

= (*AB) + C

= +(*AB)C

= +*ABC

Postfix —

A+B*C

= A + (BC*)

= A(BC*)+

= ABC*+

A*B+C

(AB*) + C

AB*C+

① (A+((B^C)−D))*(E−(A/C))

ABC^D−+EAC/−*

② [((A+B)*((C/D)] − (E^(F*G))))

AB+ CD/* E (FG*)^−