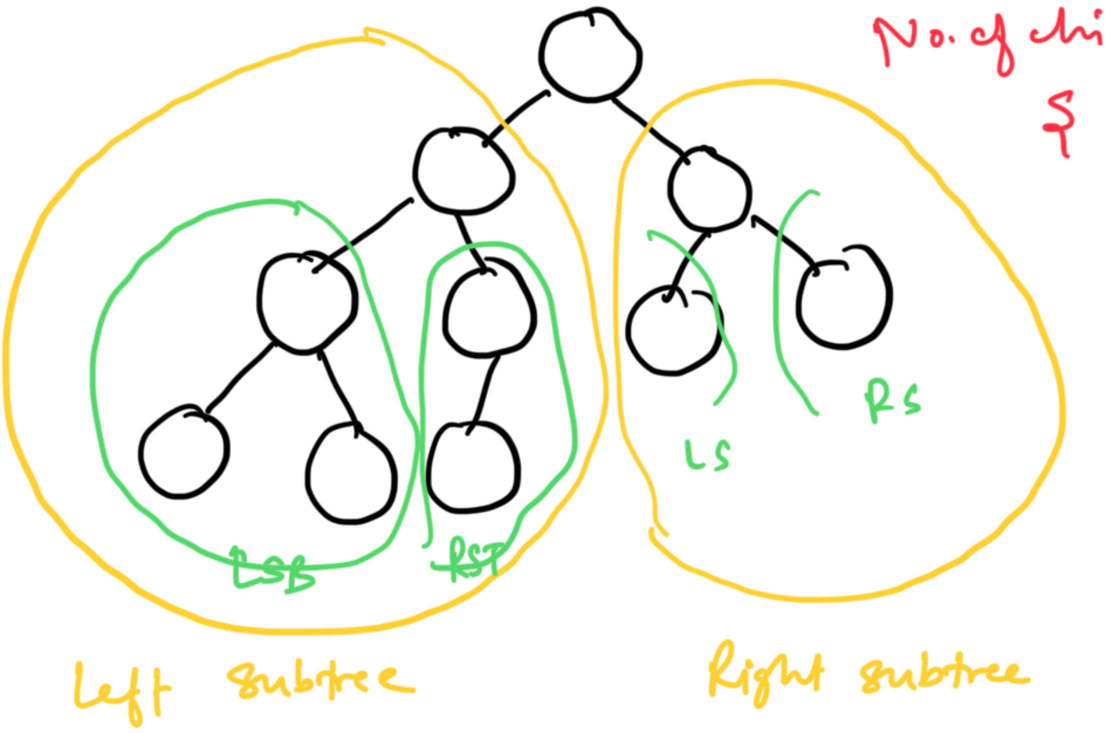
Binary Tree -

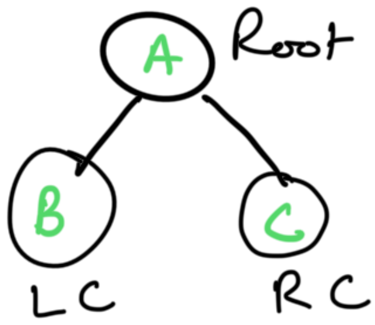
- A BT is a hierarchical structure in which each node has at most two children, refered as left child & right child



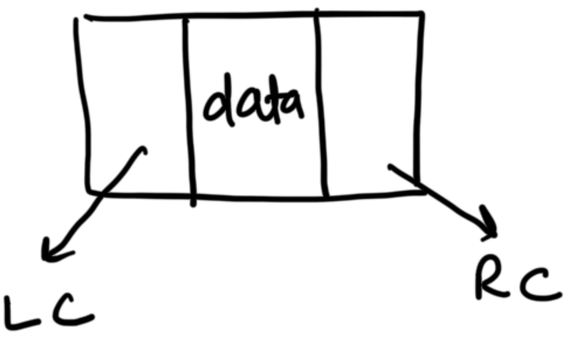
No. of childrens : max 2 {0, 1, 2}

Properties of BT :

1) Basic structure



- Data (value or key)
- LC (Reference pts to Lc)
- RC (Reference pts to RC)

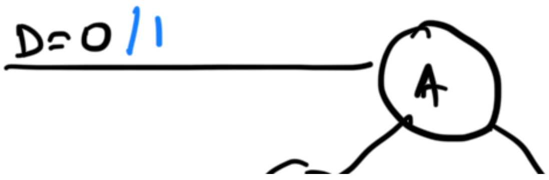


2) Maximum nodes -

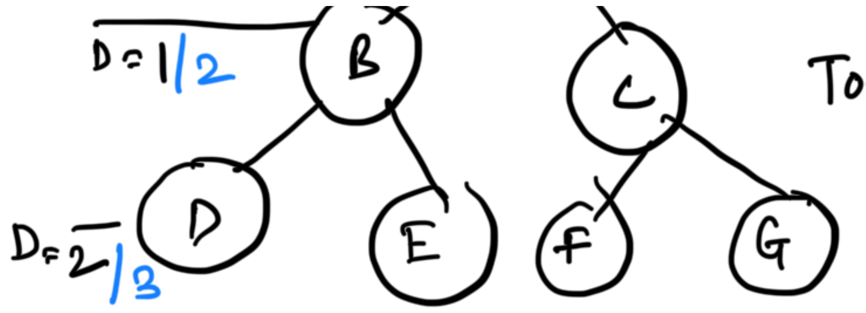
Imp! BT

- Max. no of nodes at level L ⇒ 2^L
- Max no. of nods in a BT of height h is $2^h - 1$

Eg



Depth = 3



Total no. of nods

$2^3 - 1 = 7$

full BT

The max. no. of nodes of level i of BT is $2^i, i>0$

$$\boxed{Max = 2^i , i \geq 0}$$

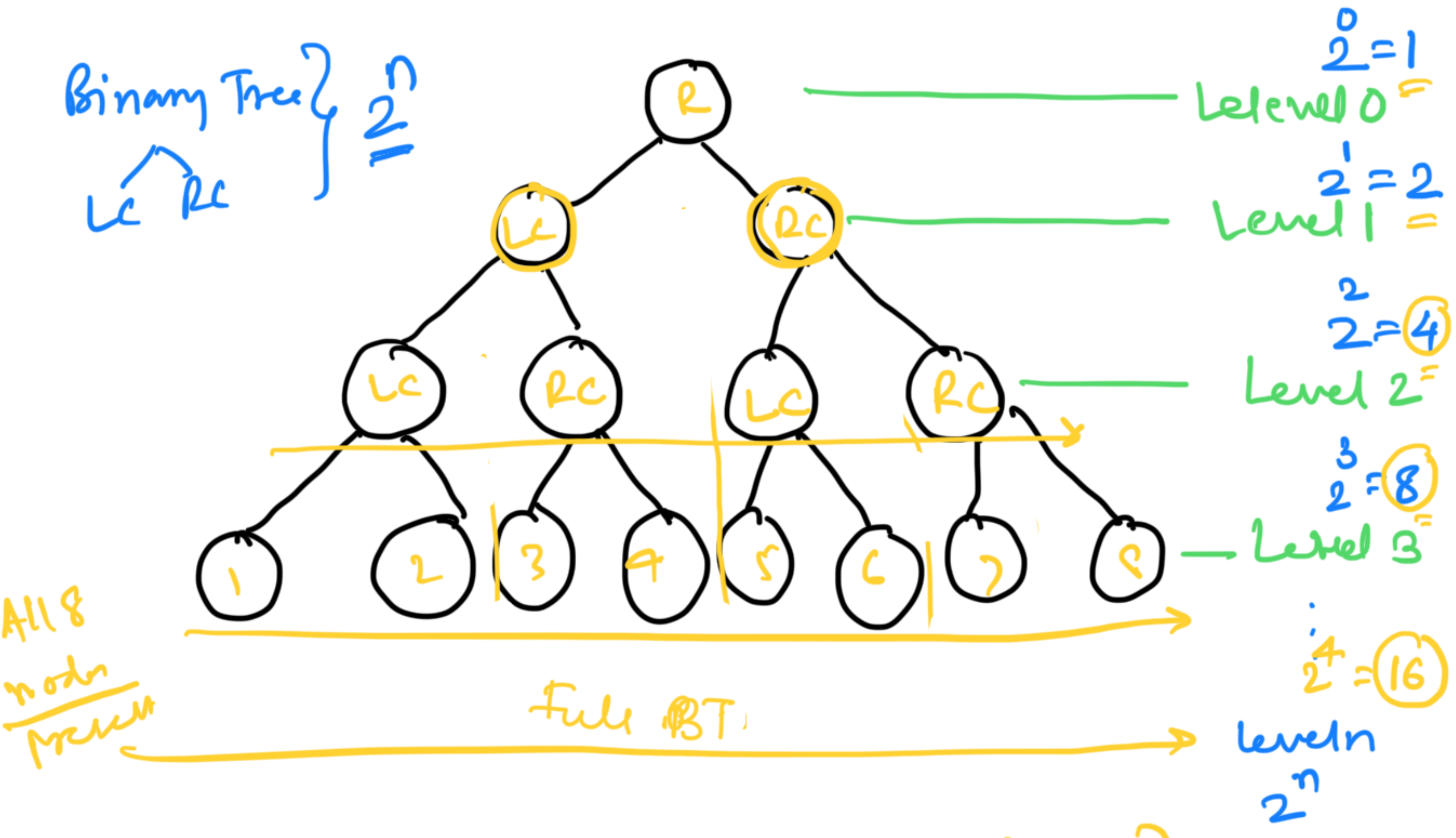
D=0

The max no. of nods of level i of BT is $2^{i-1}, i \geq 1$

full BT

$$\boxed{Max = 2^{i-1} , i \geq 1}$$

D=1

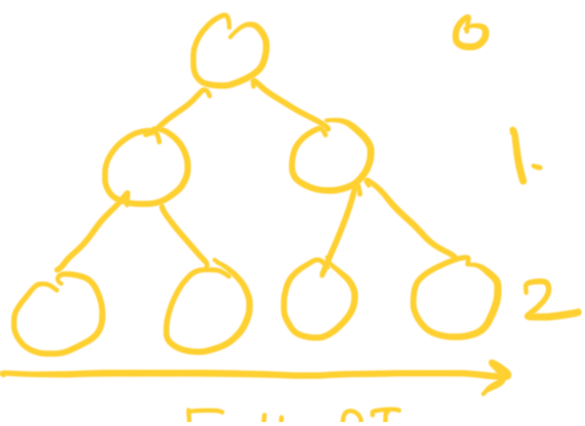


BT = O(log n)

$\underbrace{BT = Level = 2^n}_{\downarrow}$ nodes → present

Full BT

leve=2 → $2^2 = 4$



Full BT

③ Height of Tree →

The height of tree is the number of edges in the longest path from the root to leaf node.

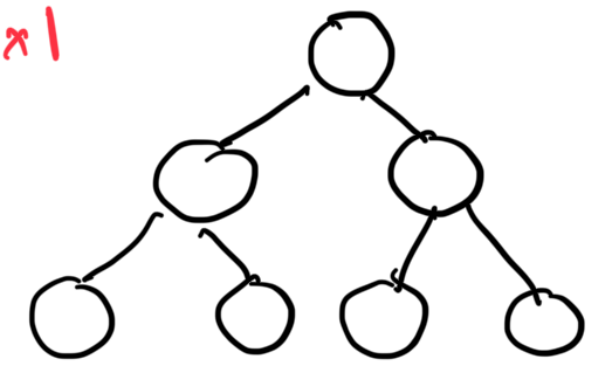
The height of tree with n nodes is at most n-1 → O(log n) Time Complexity

Type of BT:

1) Full BT → BT with complete levels

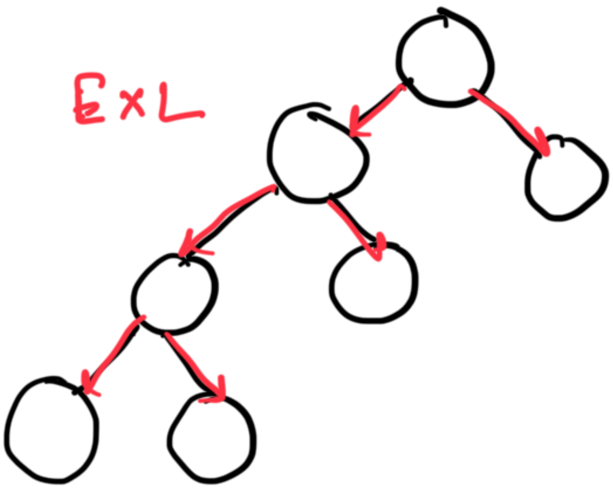
- Every node has $\boxed{0 \text{ or } 2}$ children

Ex1



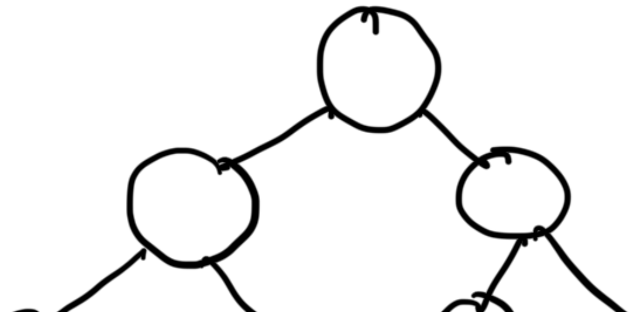
Full BT

Ex2



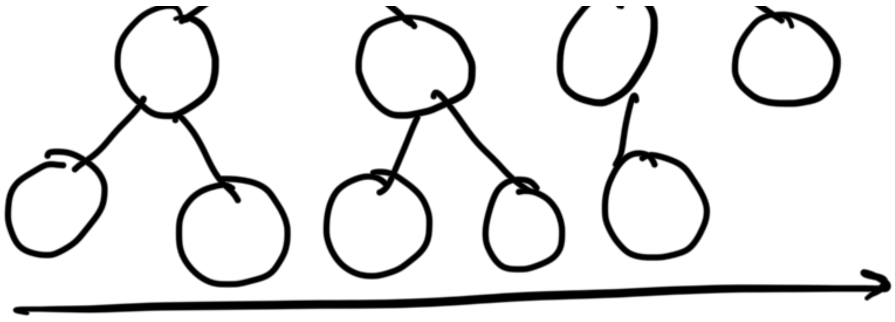
Full BT

2) Complete BT :- BT in which at every level, except possibly the last, has to be filled and all nodes are as far as left as possible



Root
LC
RC

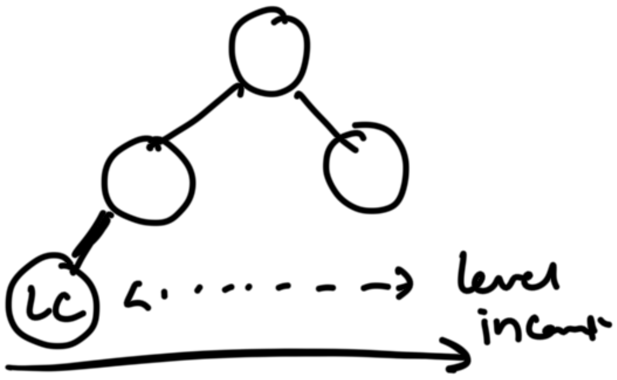
{0, 1, 2}



↓
LC

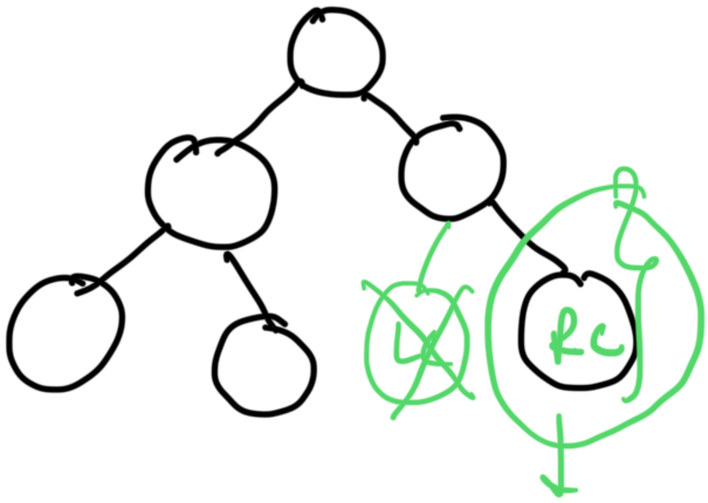
Complete BT

Ex 2



Complete BT

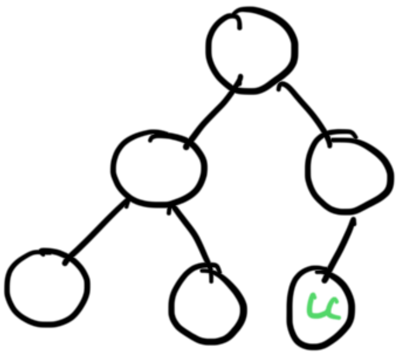
Ex 3



× Complete BT

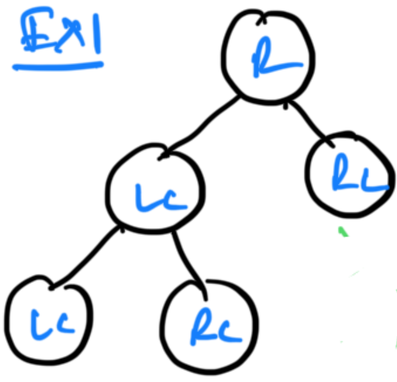
Incomplete BT

Ex 4

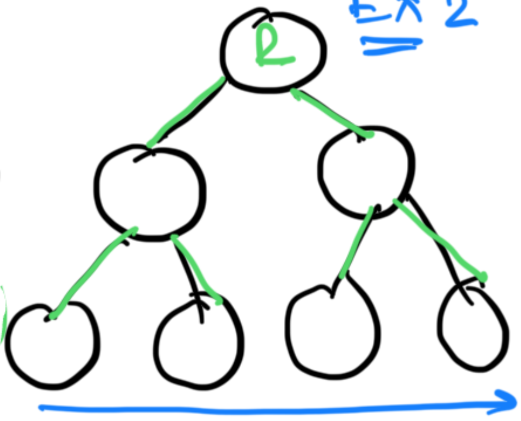


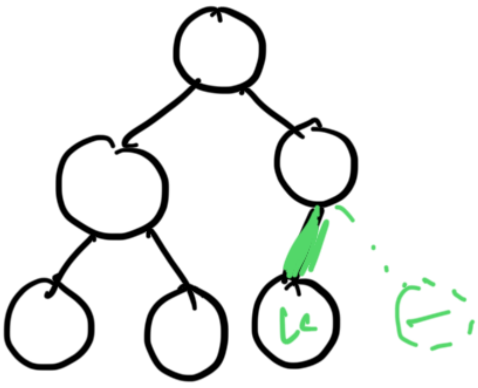
Complete BT

Ex 1



Ex 2

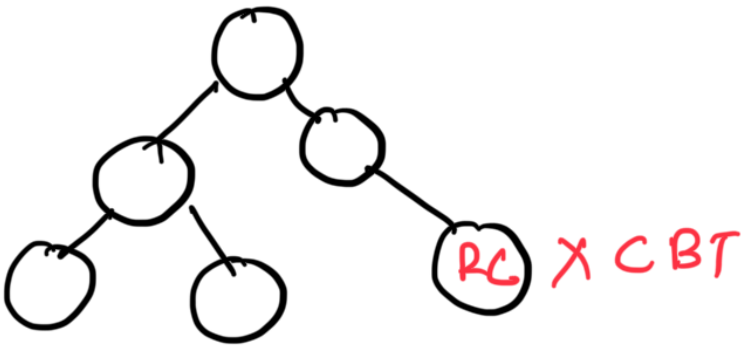


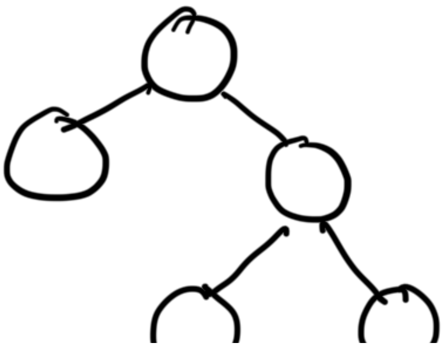


BT ✓
Full BT ✓
CBT ✓

BT
FBT
CBT
Perfect BT

BT
FBT × {0, 2}
CBT {0, 1, 2}
↓
LC





BT

FBT ✗ {0,2}

CBT ✗ {0,1,2}

↓

RC

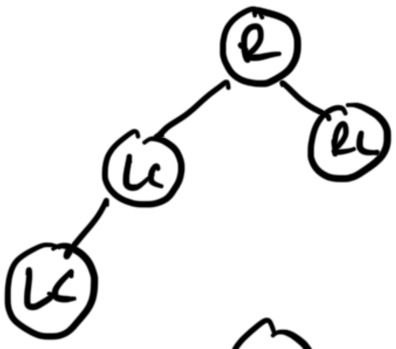
ICBT

BT

FBT {0,2}

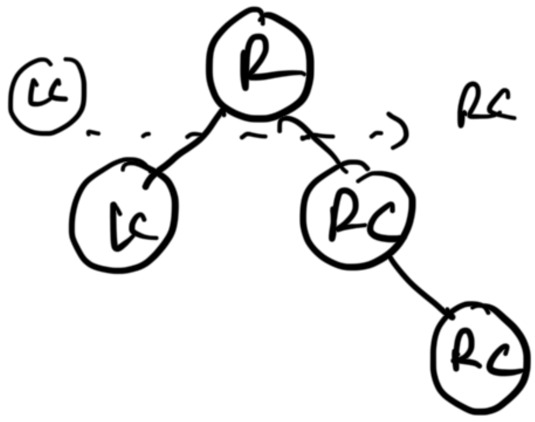
CBT ✗ {0,1,2}

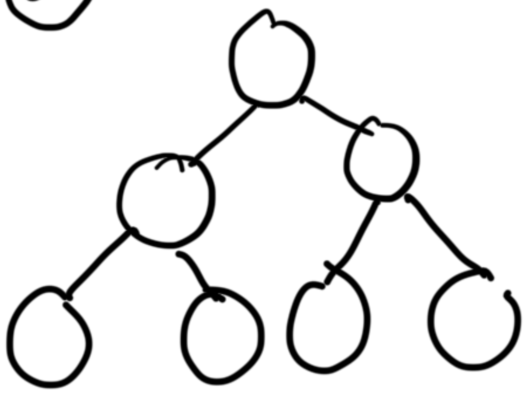
(Left c first preference)



BT

CBT





BT

CBT

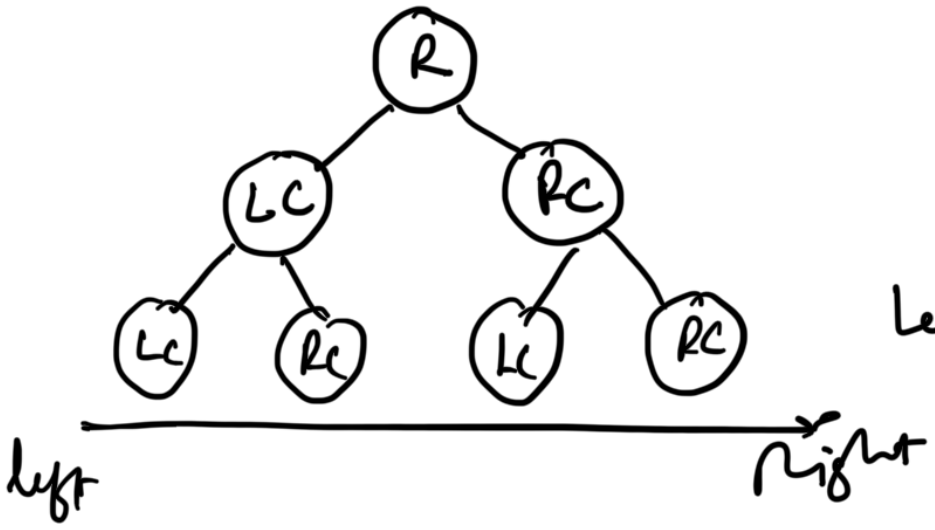
Full BT

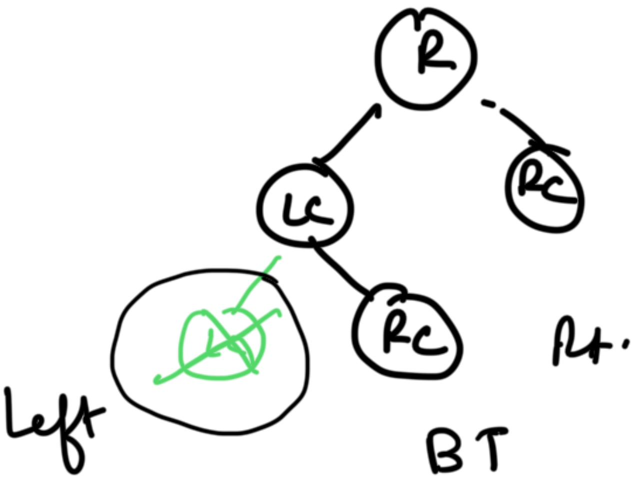
Perfect BT

BT

InCBT

(LC is missing)





BT

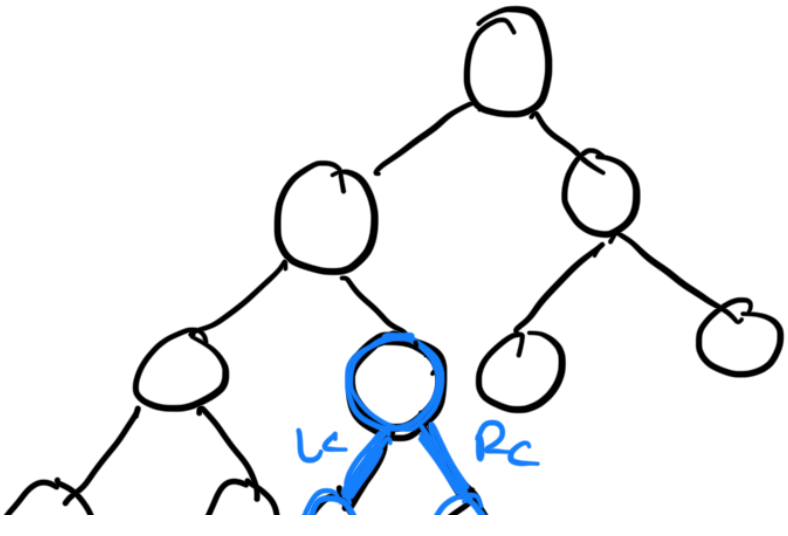
InCBT

preference to LC → CBT ✓

RC →

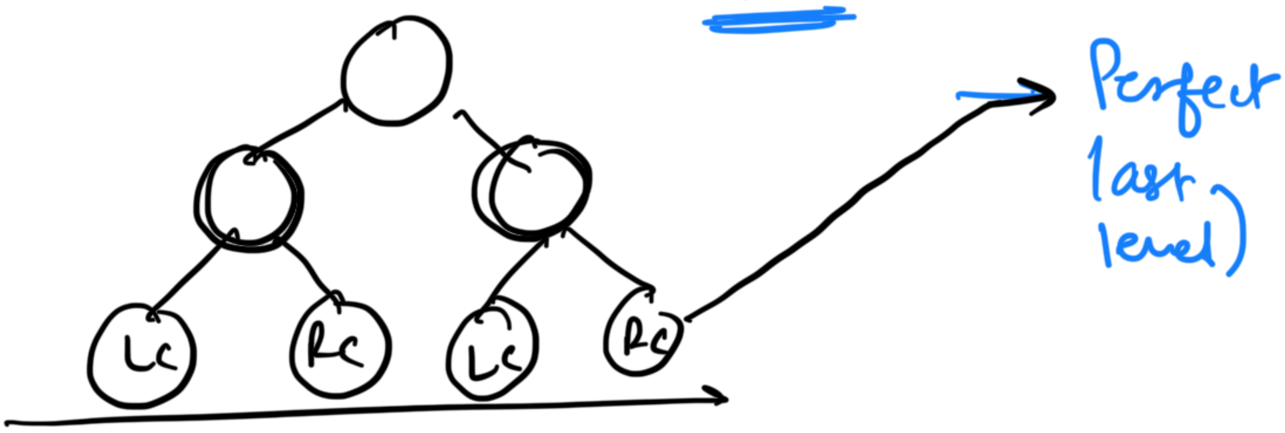
LC Missing } In CBT

Ex



{0,2} → Full BT

{0,1,2} → CBT



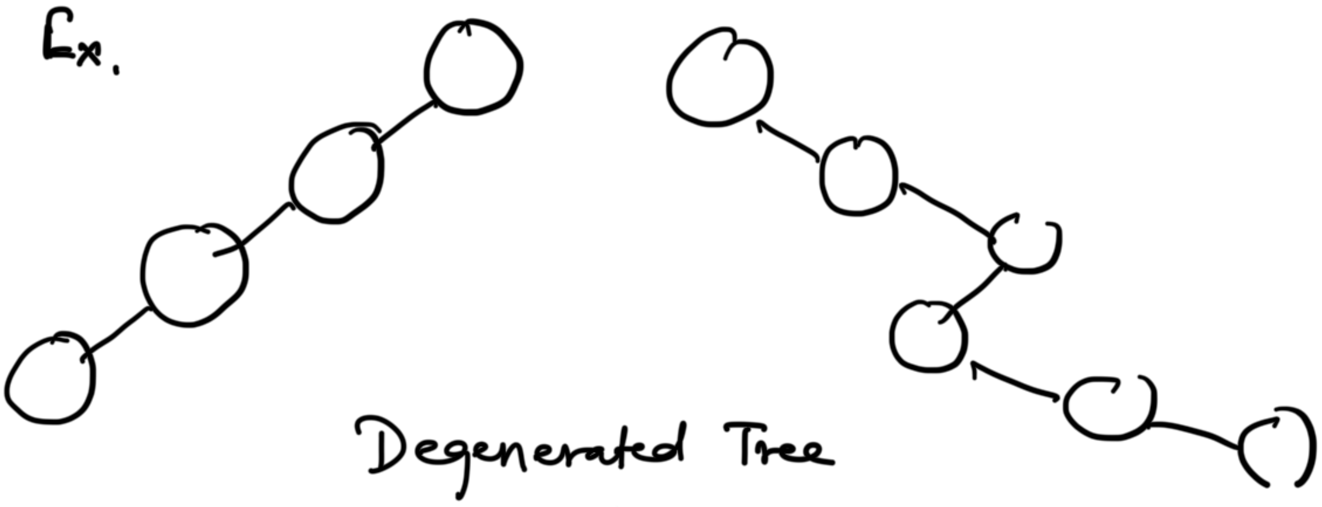
1. Full BT → Every node has 0 or 2 children

2. Perfect BT → All internal modes have 2 children. and all leaf noder are at same level

3. Complete BT – All levels are completely filled except possibly the last, which is filled from Left to right (LC → priority)

4. Degenerated Tree (Skewed tree) – Every parent node has only one child (left or riget) making the tree behaving lik a linked list.

Ex.

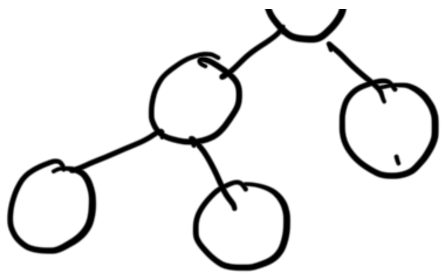


Degenerated Tree
or
skewed Tree

strictBT – Strictly BT {0,2} | one single Child is

Ex

not allowed



BT

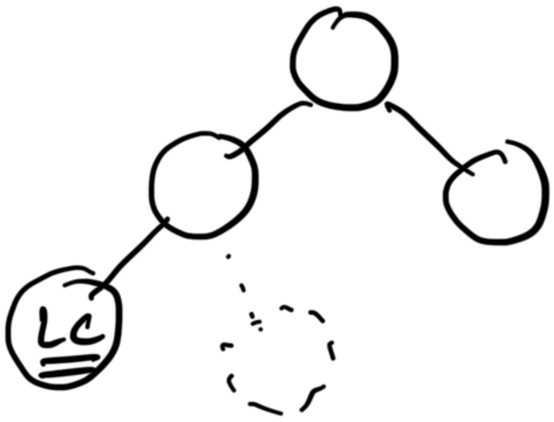
Full BT

CBT → Left

PBT X

ICBT X

ACBT → All most Complete BT



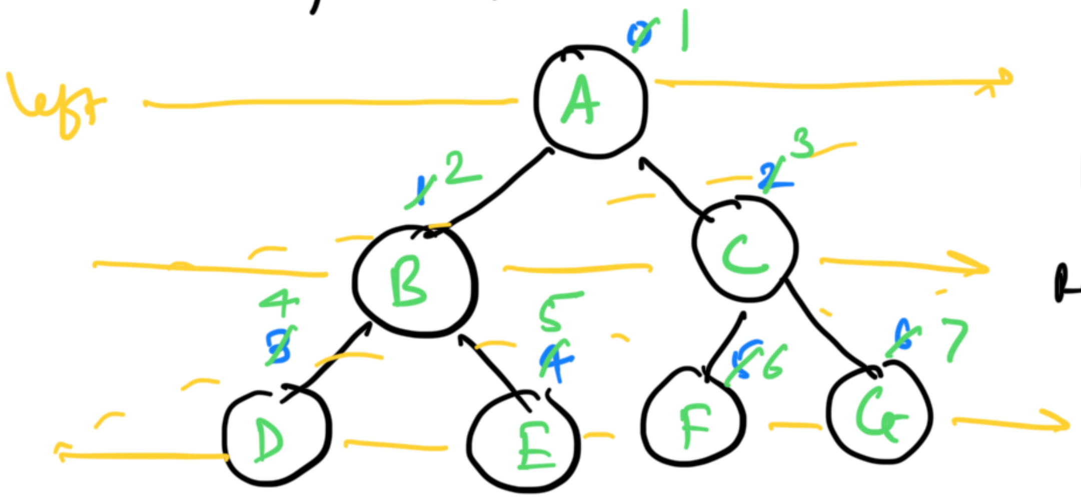
BT

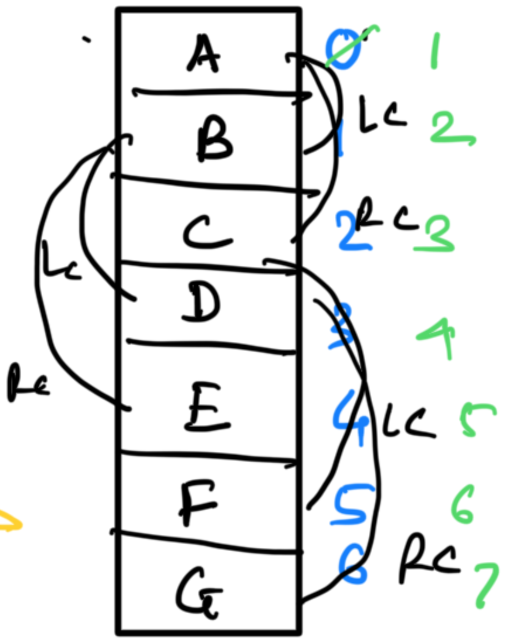
CBT

ACBT

BT Representation in 2 ways –

1. Array Implementation



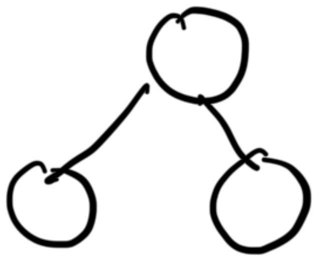


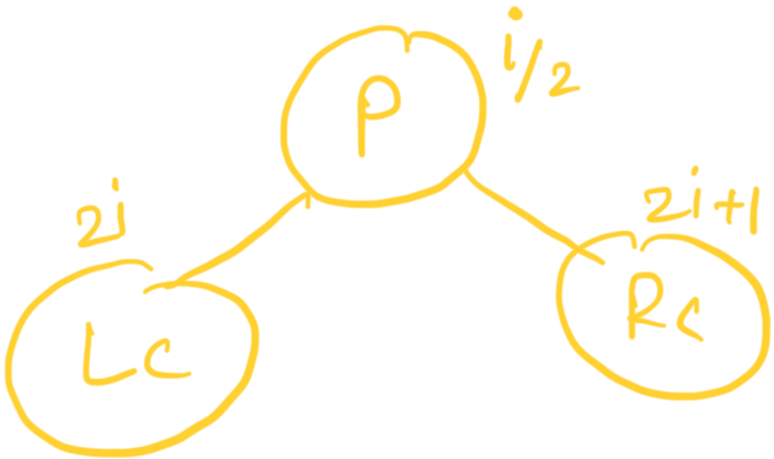
B → 2i

→ 2(1) = 2

Root LC RC

Array Implementatio



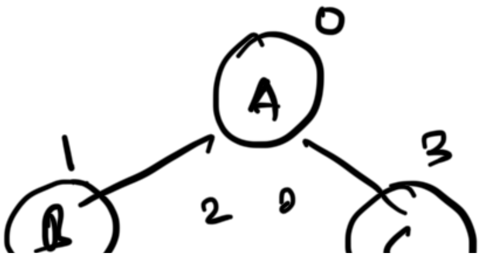


Parent = $\frac{i}{2}$

LC = 2i

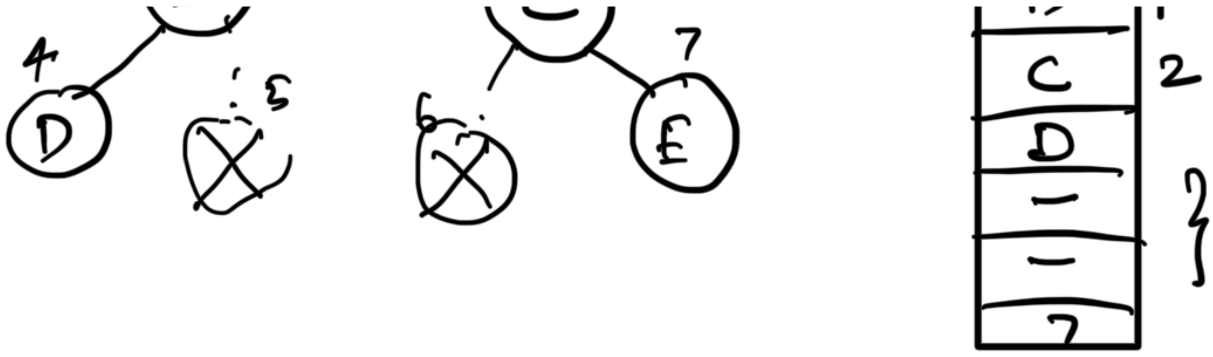
RC = 2i+1

Ex



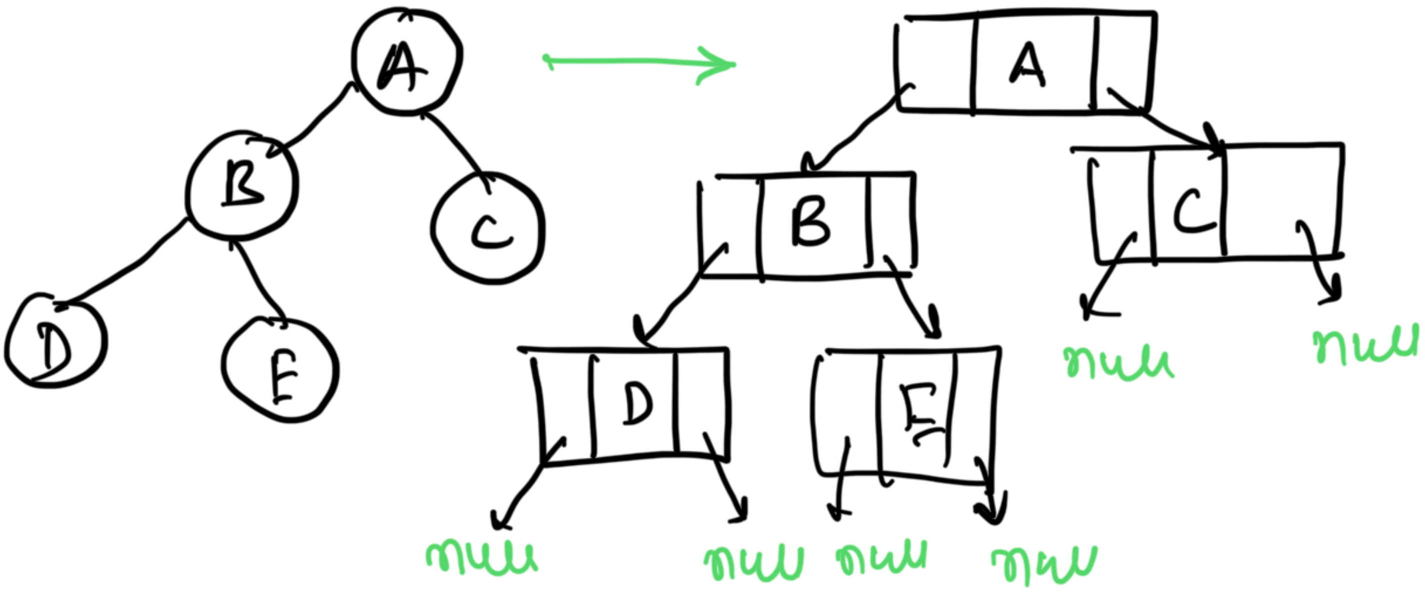
Index calculate

A	0
B	1



Disadvantage of Array Implementation { Memory wastage

2. Dynamic Implementation using DLL



?