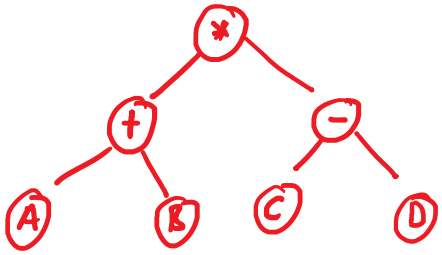
Lecture 9-10

Wednesday, October 20, 2021 8:00 AM

Expression Tree :-

A binary tree in which the internal nodes represents the operators and the external nodes represents the operands.

eg:- $(A+B)*(C-D)$



UNION type information

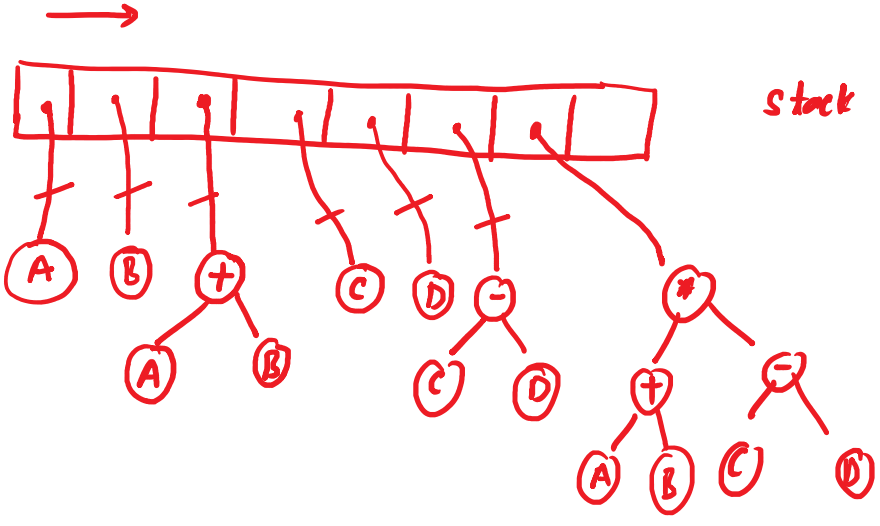
Infix : Inorder
Postfix : Postorder
Prefix : Preorder

Construction of an expression tree using postfix expression / postorder traversal

AB+ CD- *

1) If operand is encountered then push it into stack
2) If operator is encountered then pop 2 elements (a and b)
   and then push the tree with root as operator, its left child as 'a' and its right child as 'b' into the stack

pop 2 ↑ pop 1 ↑



stack is array of pointers

AB+CD-*

Push (A)
Push (B)
B → Pop()
A → Pop()

A → Pop()
Push (A+B)
Push (C)
Push (D)
C → Pop()
D → Pop()

Push (C-D)
C-D → Pop()
A+B → Pop()
Push ((A+B) * (C-D))

AVL Tree :- Adelson - Velskii Landis 1962
2 authors

Binary Search Tree — $\theta(\log_2 n)$ but $O(n)$
skew BST

Height of BST can be more

" AVL tree is a height balanced binary search tree "

Height balanced tree : An empty binary tree T is height balanced.
If T is a non-empty binary tree with T_L & T_R as its LST and RST respectively, then T is height balanced iff

1) T_L & T_R are height balanced
2) $|h_L - h_R| \leq 1$ where h_L & h_R are the heights of T_L & T_R respectively.

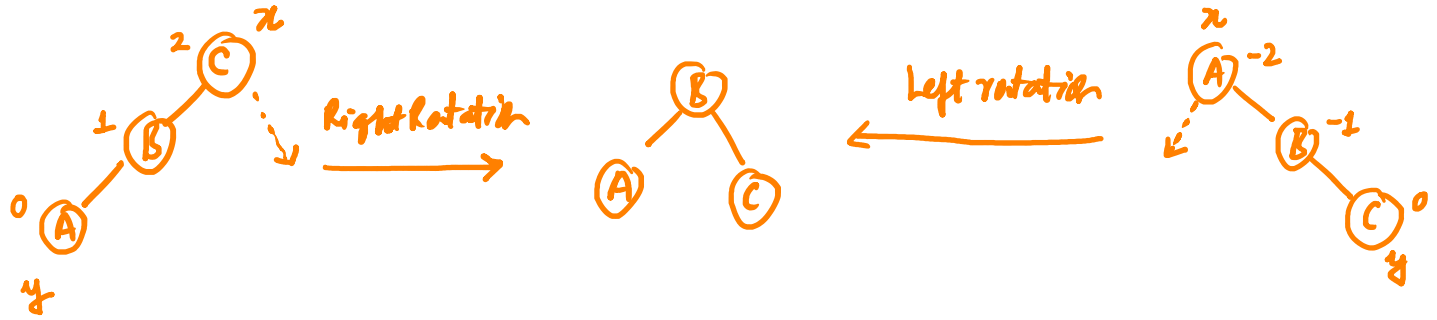
Balance Factor of a node = $h_L - h_R \in \{-1, 0, 1\}$

Insertion in a AVL tree :- New node 'ij' is inserted as per the logic of BST.

AVL Tree Node

Lets assume that there is an imbalance in the AVL tree on inserting 'ij'
Balance Factor of atleast one node along the path from 'ij' to root is either +2 or -2

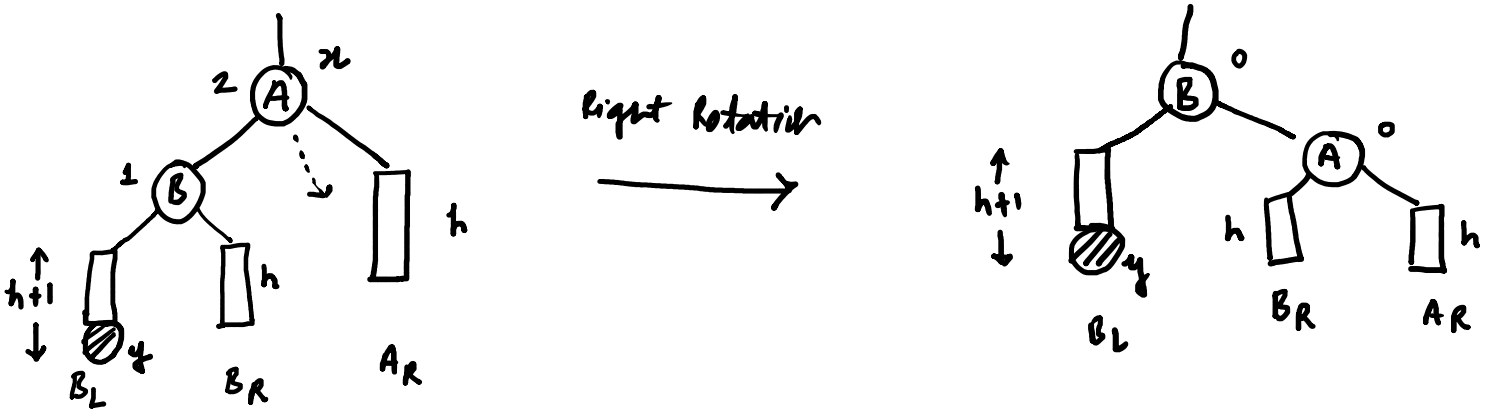
Rotation → Left rotation
→ Right rotation



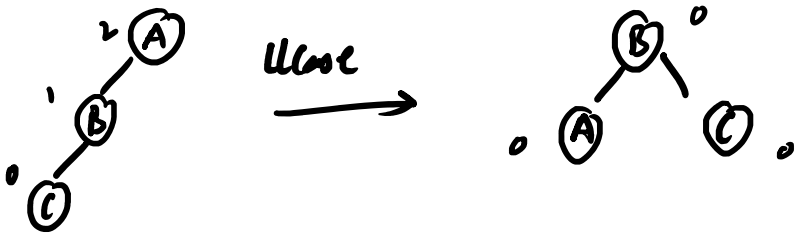
A new node 'y' got inserted and let 'x' be the nearest ancestor of 'y' that creates imbalance. With respect to 'x' look for the location of 'y'. There are 4 possibilities :-

1) LL Case – y is present in the LST of LST of x
2) LR Case – y is present in the RST of LST of x
3) RL Case – y is present in the LST of RST of x
4) RR Case – y is present in the RST of RST of x

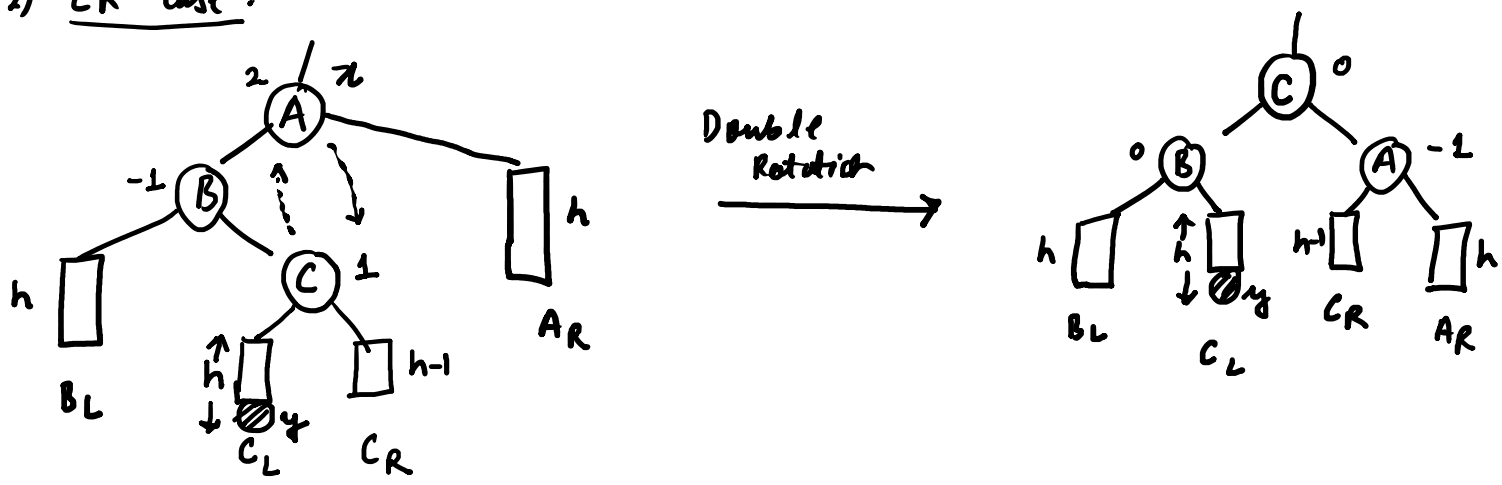
1) LL case :-

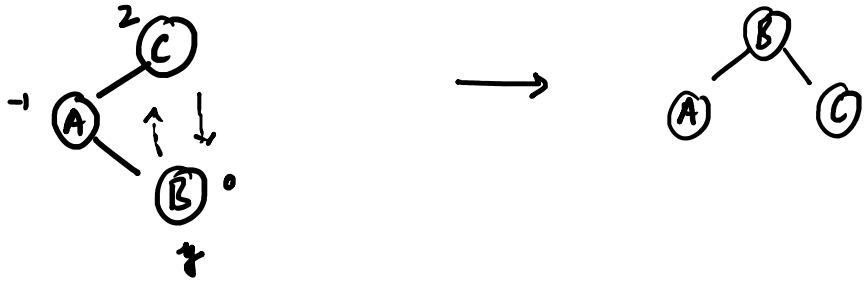


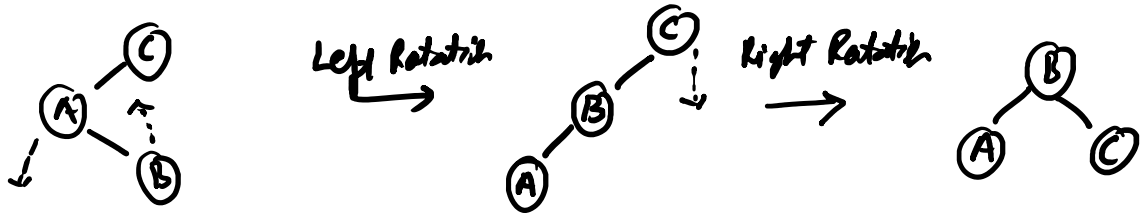
Parent of B, Parent of A, B_R Parent, B's right child, A's left child



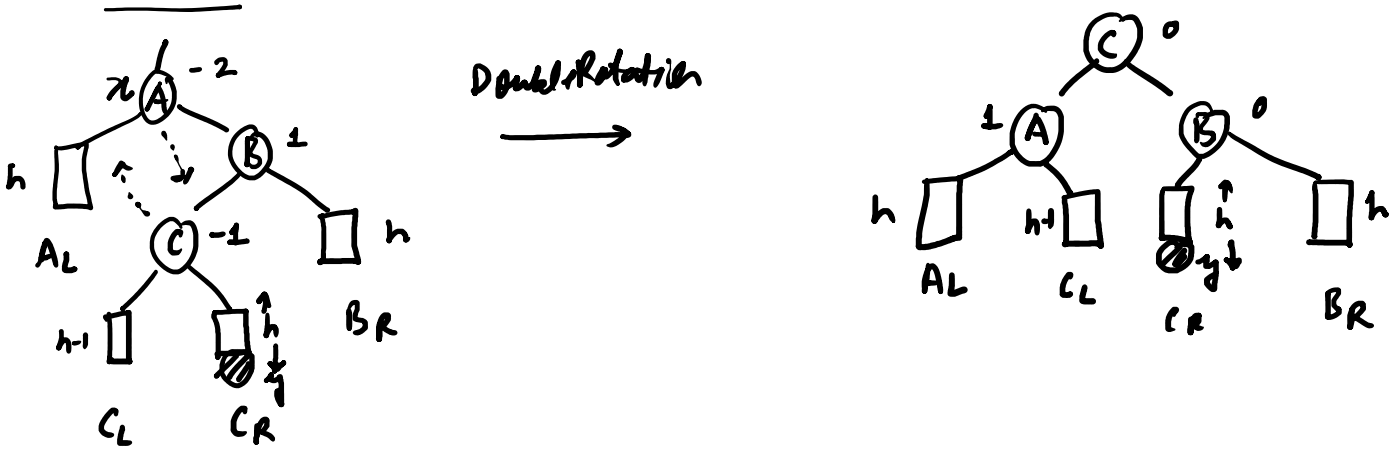
2) LR Case :-

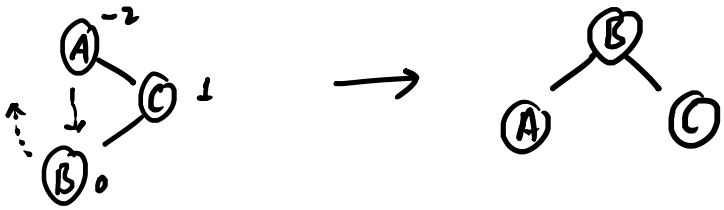


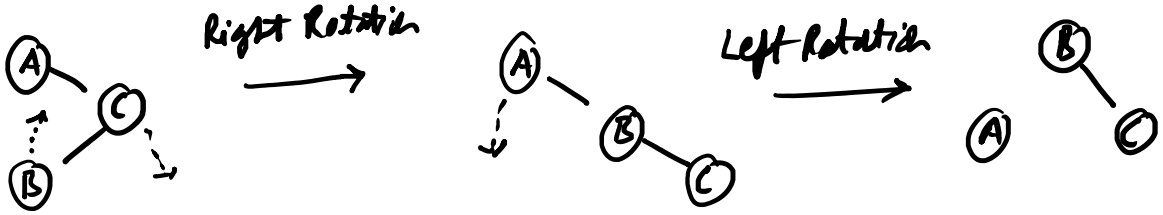




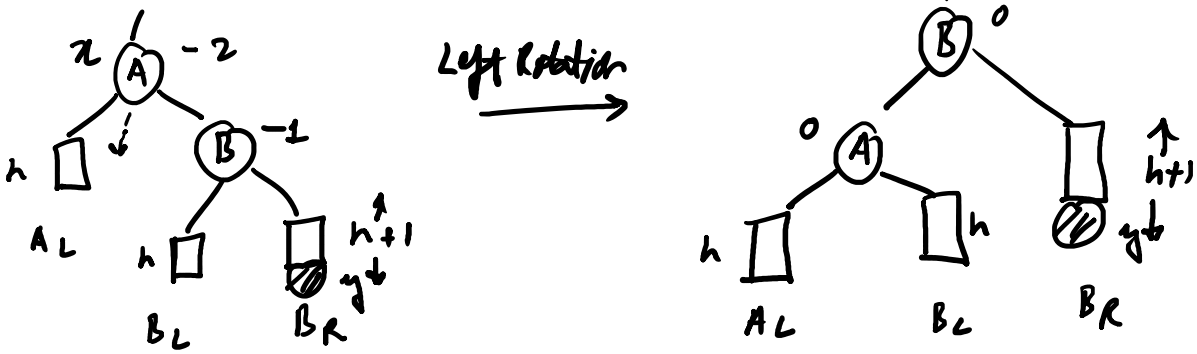
3) RL Case :-

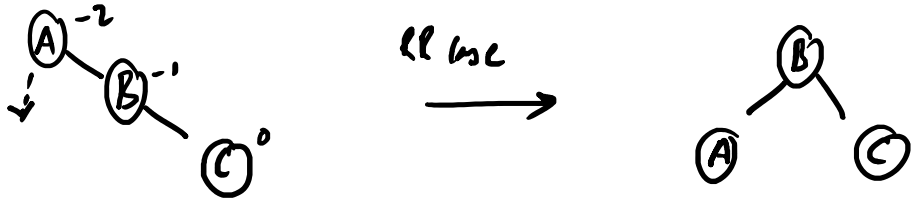






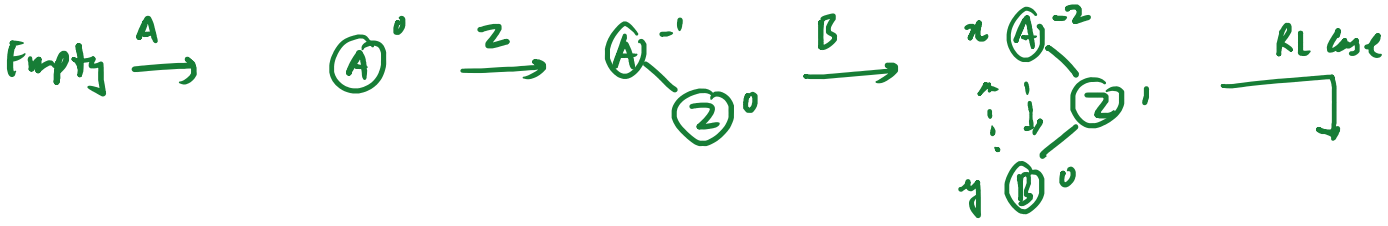
4) RR Case :-

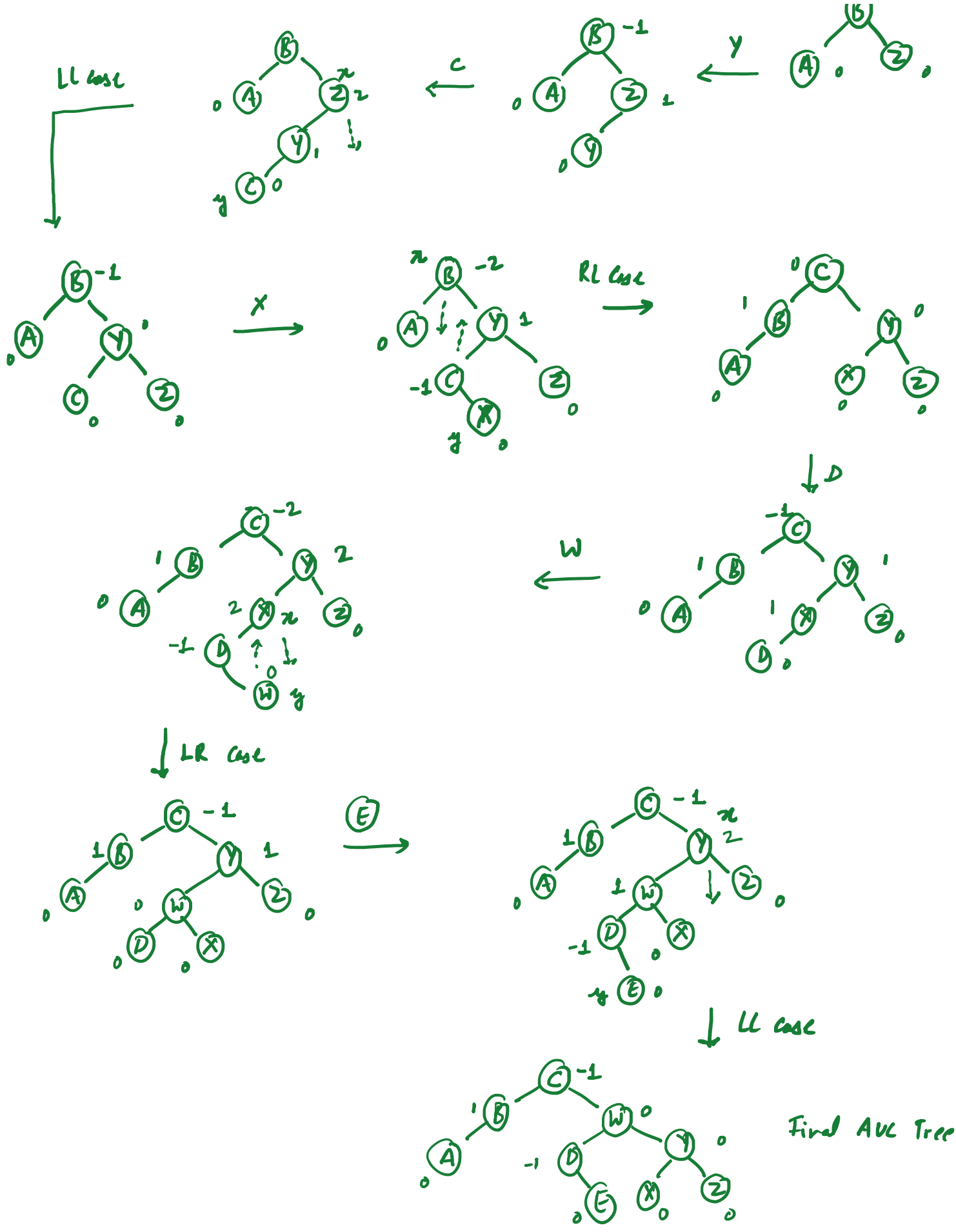




Insert the following data one at a time in an empty AVL Tree

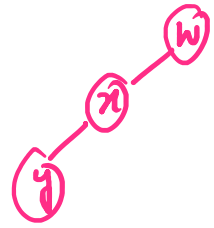
A, Z, B, Y, C, X, D, W, E



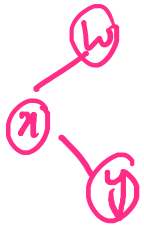

LL case
C
y
X
RL case
D
W
LR case
E
LL case
Final AVL Tree

Deletion in a AVL Tree :-

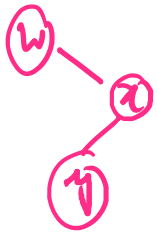
1) Delete as per the logic of BST Deletion.
2) Lets say 'w' be the node whose balance factor is either +2 or -2 & is the nearest ancestor to the deleted node.
3) Lets say 'x' be the child of 'w' having higher height.
4) Lets say 'y' be the child of 'x' having higher height.



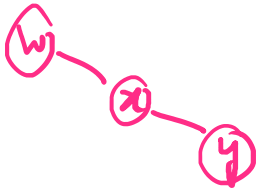
LL Case



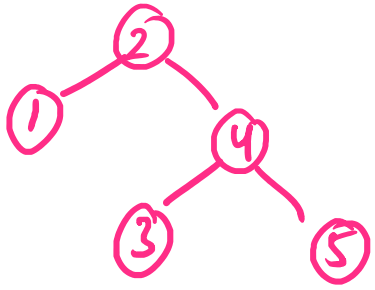
LR case



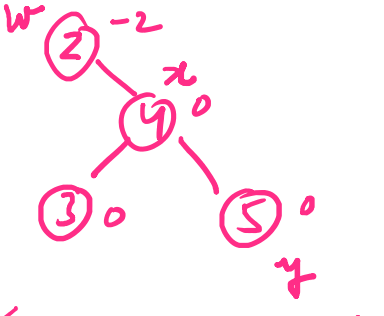
RL Case



RR case

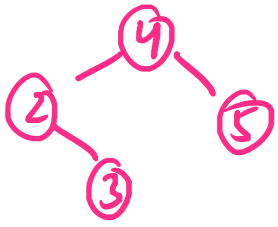


Delete 1



RL or RR

LL case



Priority Queue :-

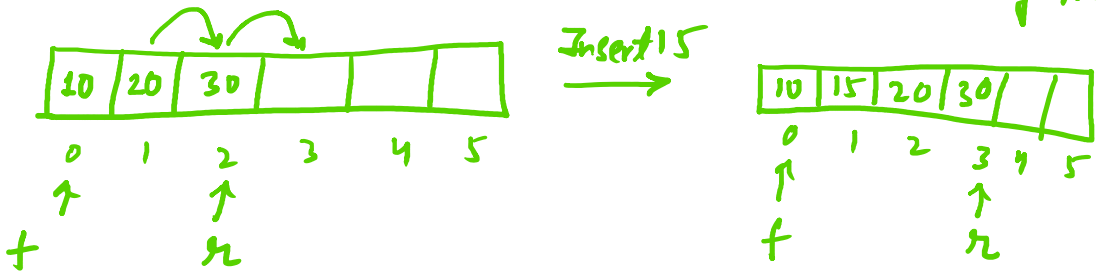
Higher Priority Data must be accessed first

Max Priority Queue → Higher data has higher priority

Min Priority Queue → Lower data has higher priority

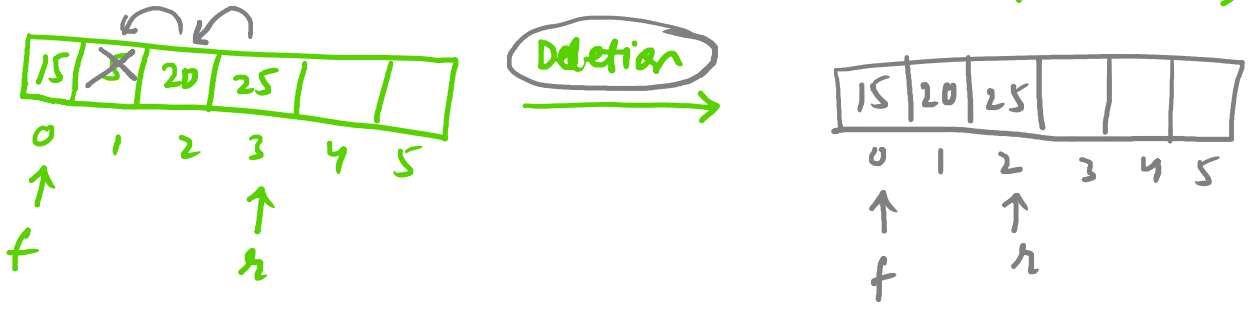
Min Priority Queue :-

Case 1: Deletion is a frequent operation (More of deletion & less of insertion)



Deletion is O(1) & Insertion is O(n)

Case 2: Insertion is a frequent operation (More of insertion & less of deletion)



Deletion is O(n) & Insertion O(1)