FRANKLIN'S LECTURES

ALGORITHM ANALYSIS AND DESIGN

Module 2 | Part 1

CST306

SYLLABUS

- Advanced Data Structure and Graph Algorithm
- Self Balancing Tree
    - AVL Tree: Insertion and deletion opertions with all rotations indetail
- Disjoint Sets
    - Disjoint set operations
    - Union and find algorithms
- DFS and BFS traversals - Analysis
- Strongly Connected Components of a Directed graph
- Topological Sorting

PRACTICAL APPLICATIONS

1. Efficient Data Searching
2. Group Management
3. Path and Connectivity Analysis
4. Task and Dependency Management

AVL TREES

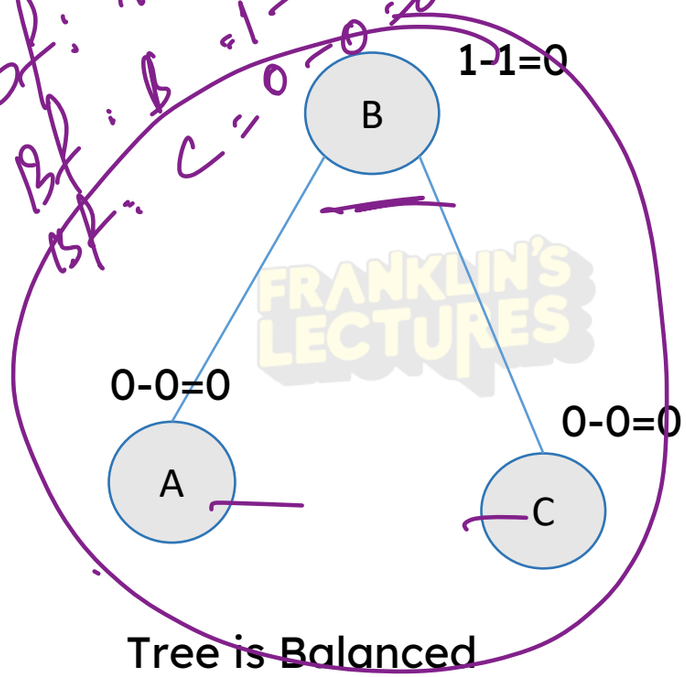
BST

- AVL Tree is invented by GM Adelson - Velsky and EM Landis in 1962. The tree is named AVL in honour of its inventors.
- AVL Tree can be defined as height balanced binary search tree in which each node is associated with a balance factor.

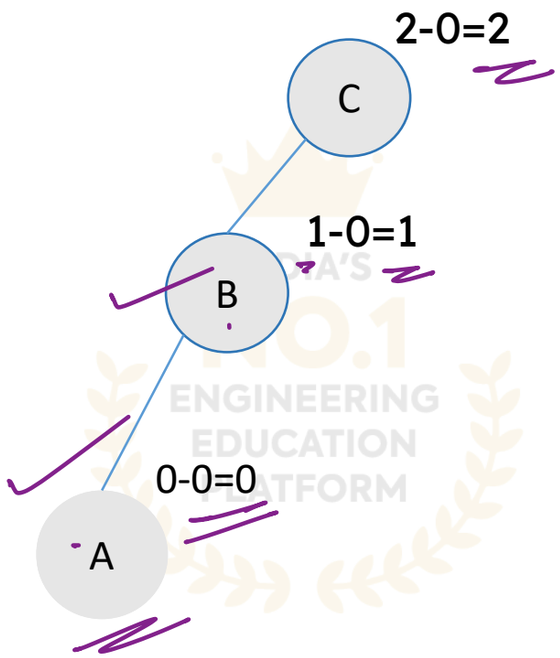
Balance Factor

- Balance Factor of a node = height of left subtree – height of right subtree
- In an AVL tree balance factor of every node is {-1, 0 or +1}
- Otherwise the tree will be unbalanced and need to be balanced.

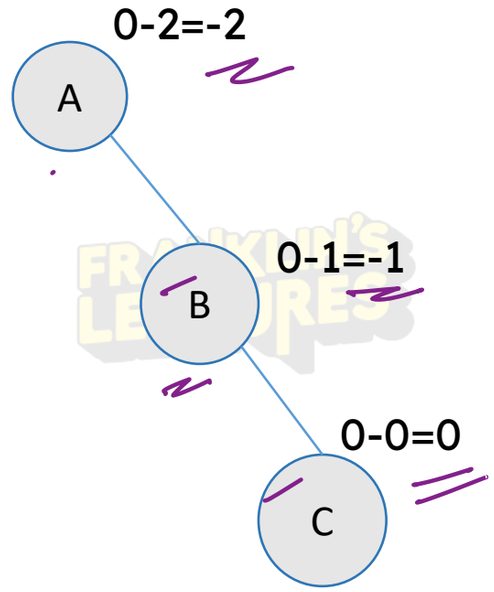
Balanced tree



Tree is Balanced



Tree is not Balanced.
Balance Factor of C is 2



Tree is not Balanced.
Balance factor of A is -2

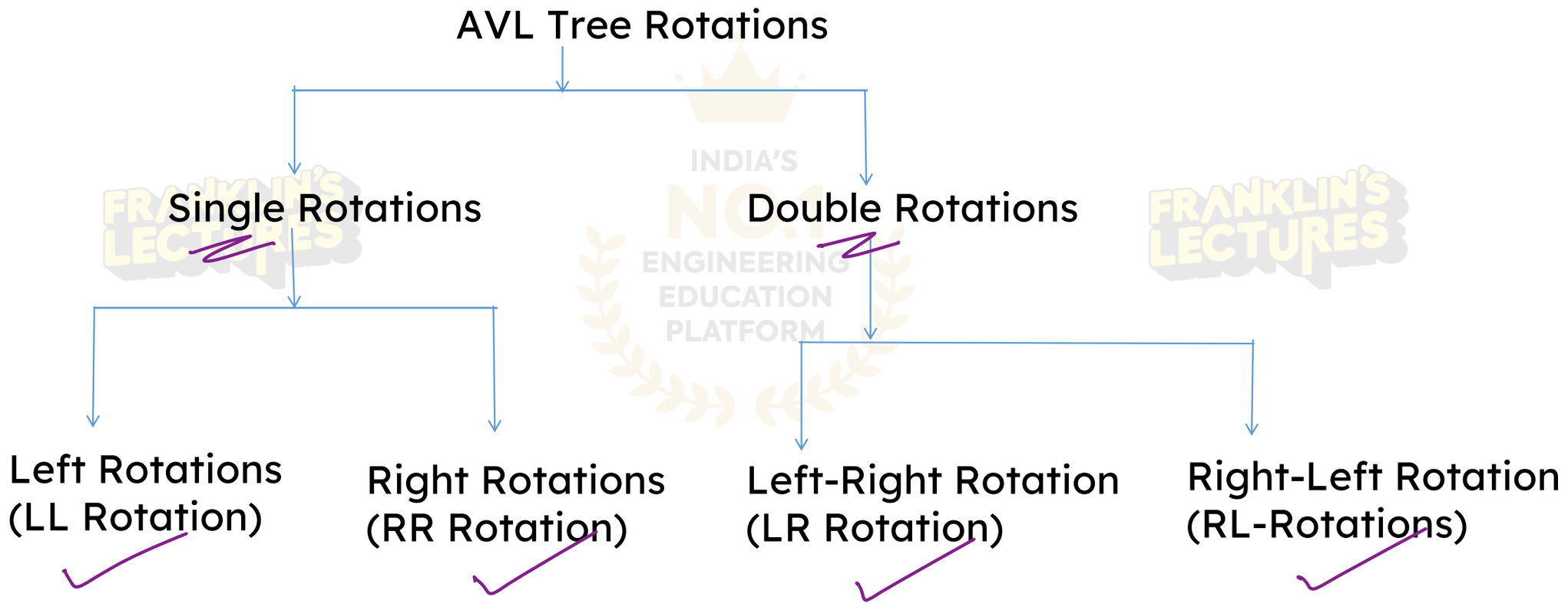
WHY AVL TREE?

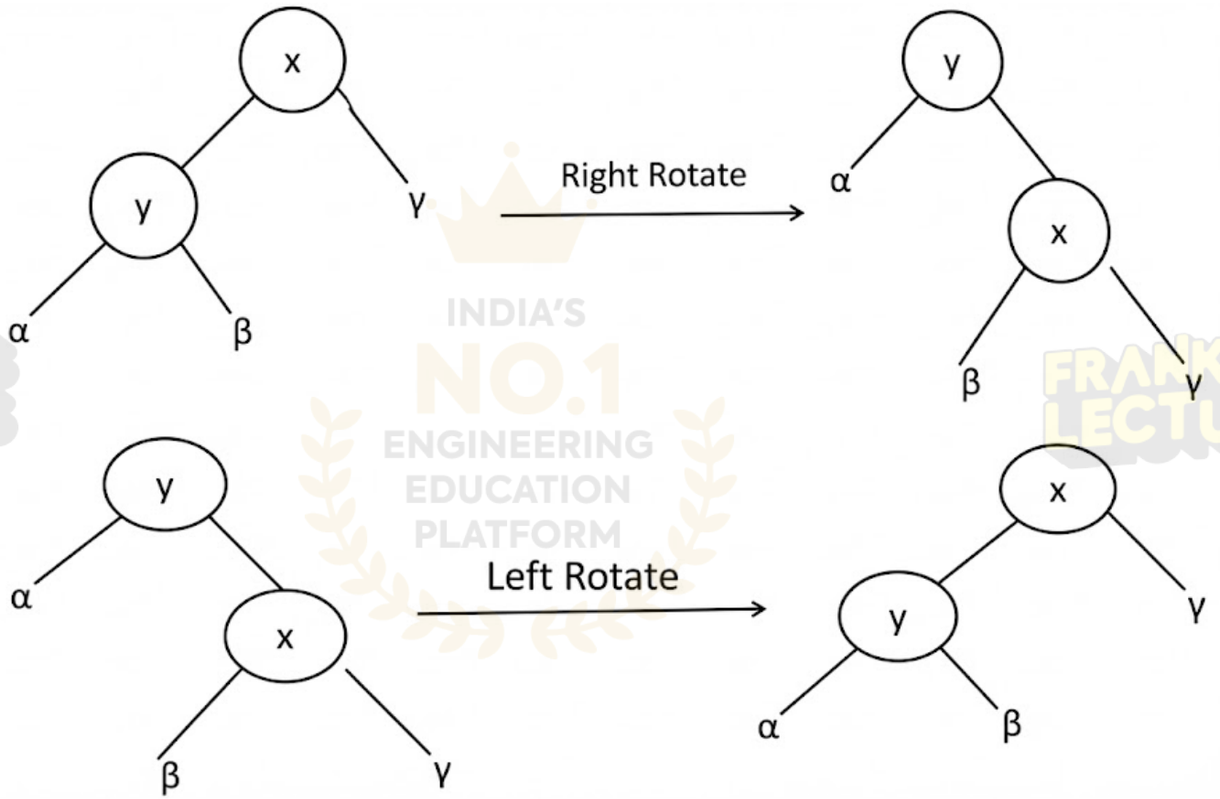
O(h) ⇒ O(n). bad

FRANKLIN'S LECTURES

- Most of the Binary Search Tree(BST) operations (eg: search, insertion, deletion etc) take O(h) time where h is the height of the BST.
- The minimum height of the BST is log n
- The height of an AVL tree is always O(log n) where n is the number of nodes in the tree.
- So the time complexity of all AVL tree operations are O(log n)
- An AVL tree becomes imbalanced due to some insertion or deletion operations
- We use rotation operation to make the tree balanced.

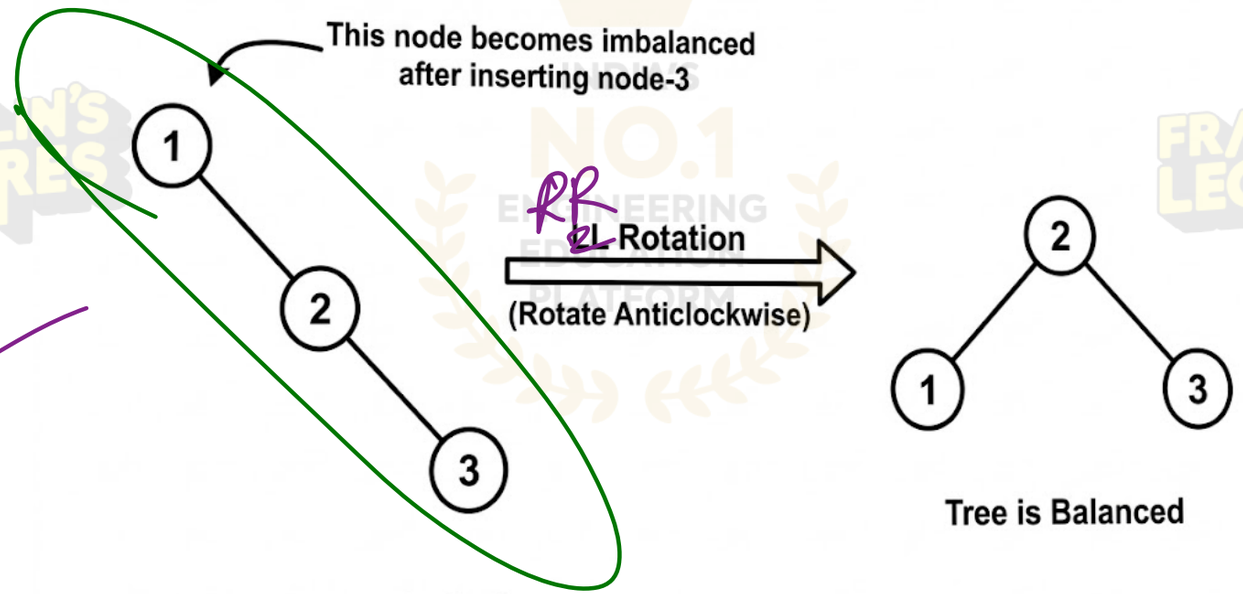
- There are 4 types of rotations.




x
y
γ
α
β
Right Rotate
y
α
x
β
γ
y
α
x
β
γ
Left Rotate
x
y
γ
α
β

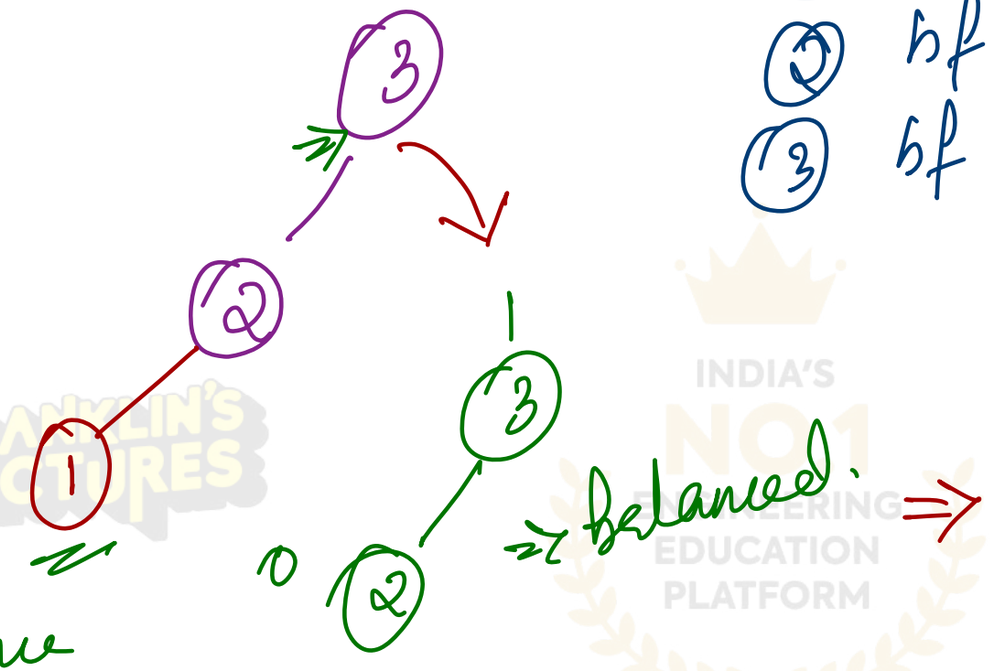
SINGLE LEFT ROTATION(LL ROTATION)

- In LL rotation every node moves one position to left from the current position



LL Rotation

insert 1



LL imbalance

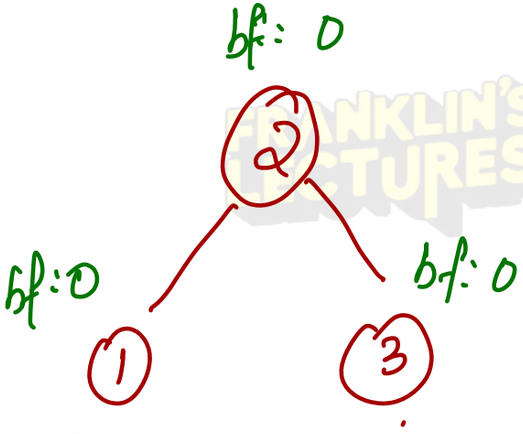
:- LL rotation (right turn)

① bf: 0-0=0

② bf: 1-0=1

③ bf: 2-0=2

⇒ balanced.

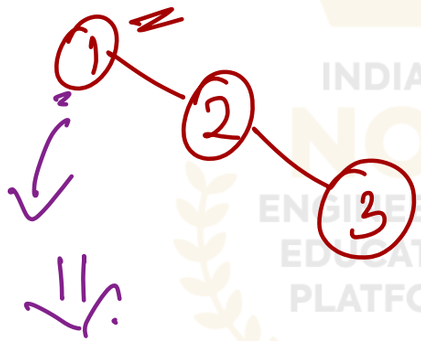


Balanced tree.

SINGLE RIGHT ROTATION

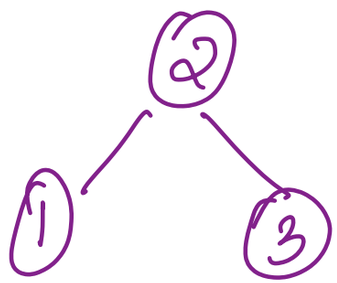
- In RR rotation every node moves one position to right from the current position

insert 3



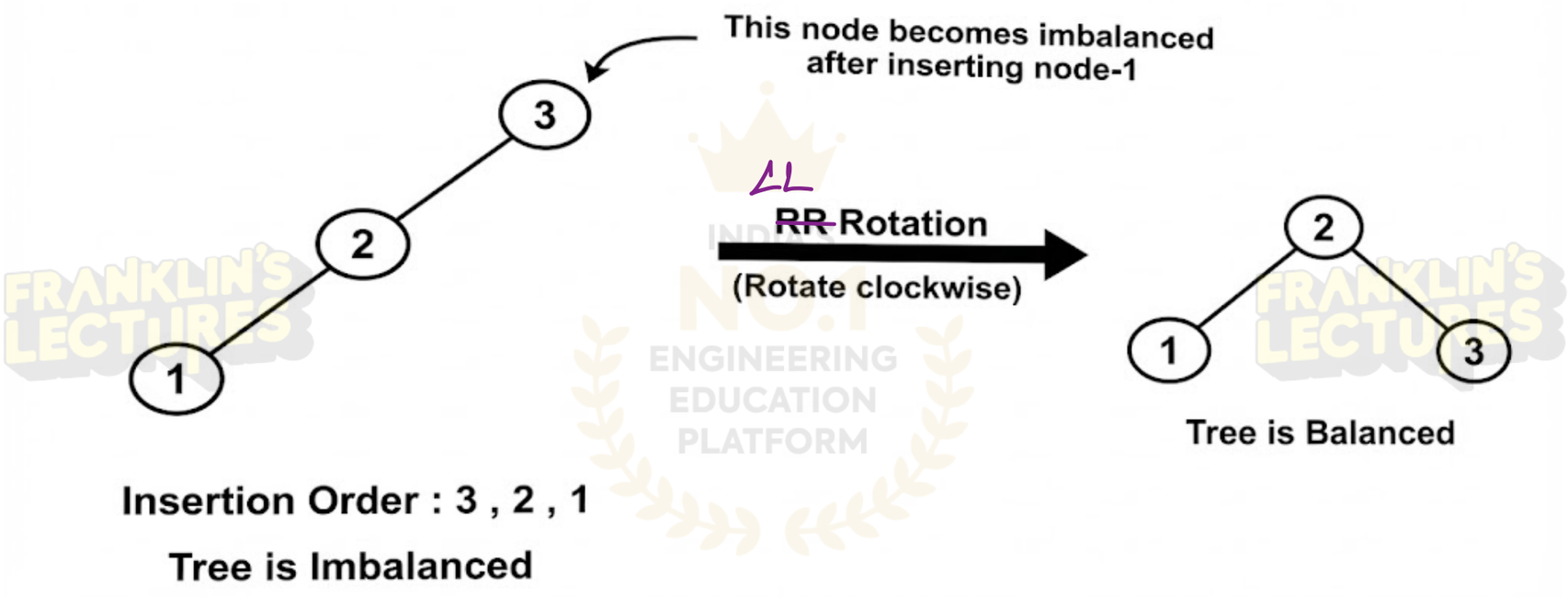
③ bf = 0
② bf : 0 − 1 : −1
① bf : 0 − 2 : −2

RR imbalance :- RR Rotation.
(left turn)



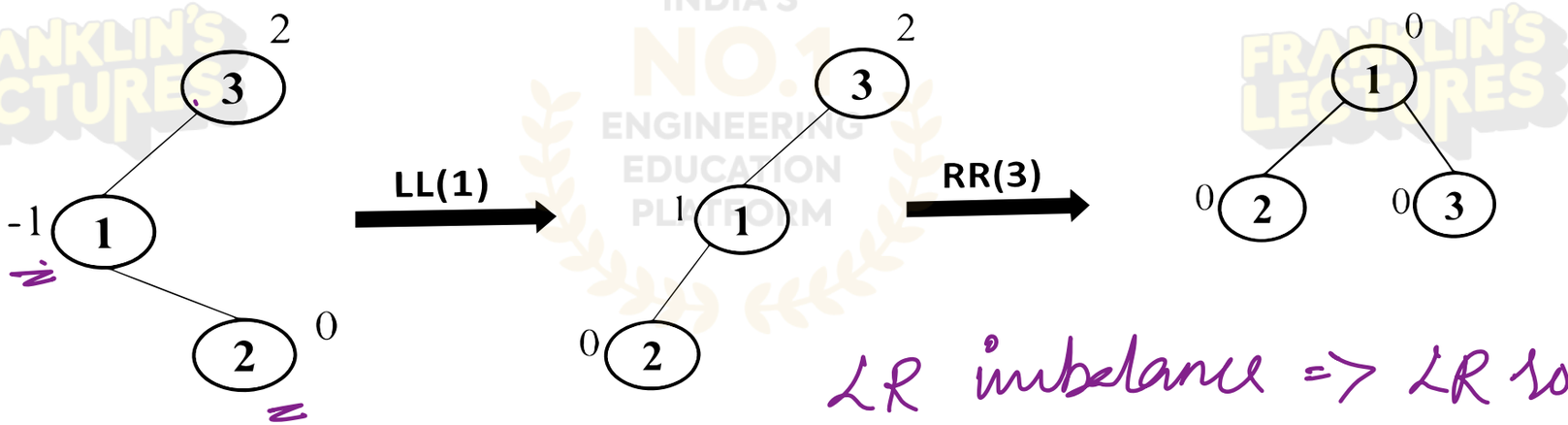
① bf : 0
② bf : 0
③ bf : 0

⇒ balanced


This node becomes imbalanced
after inserting node-1
3
2
1
LL
RR Rotation
(Rotate clockwise)
2
1
3
Tree is Balanced
Insertion Order : 3 , 2 , 1
Tree is Imbalanced

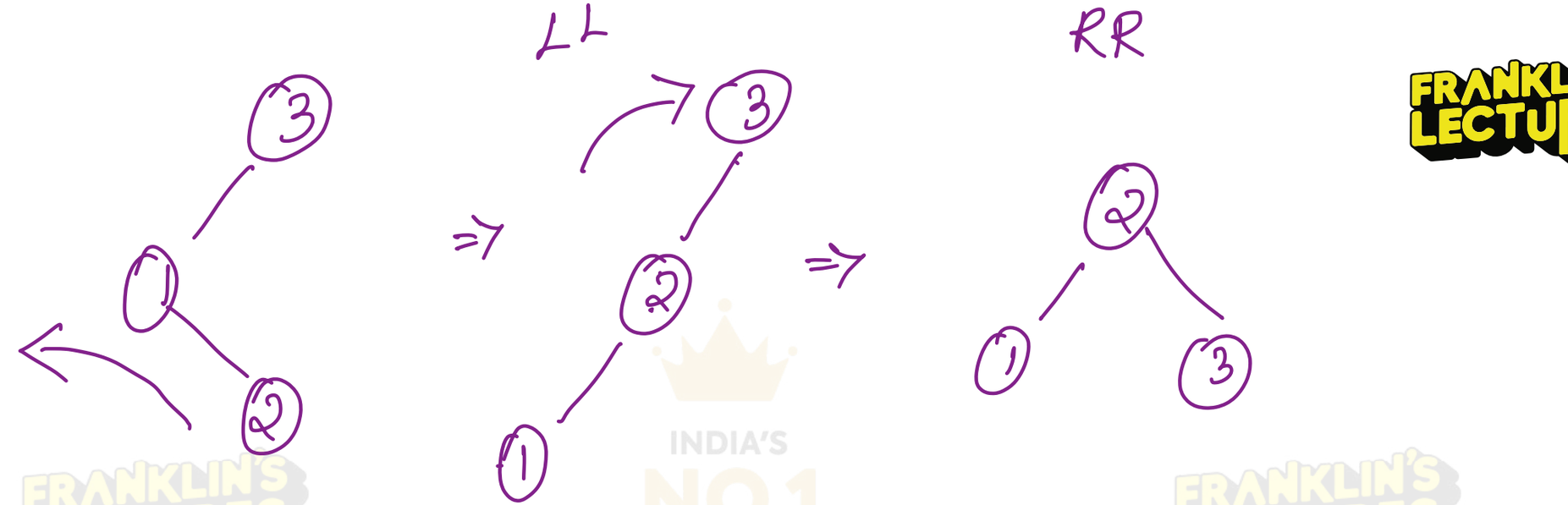
LEFT -RIGHT ROTATION(LR ROTATION)

- The LR rotation is the combination of single left rotation followed by single right rotation.



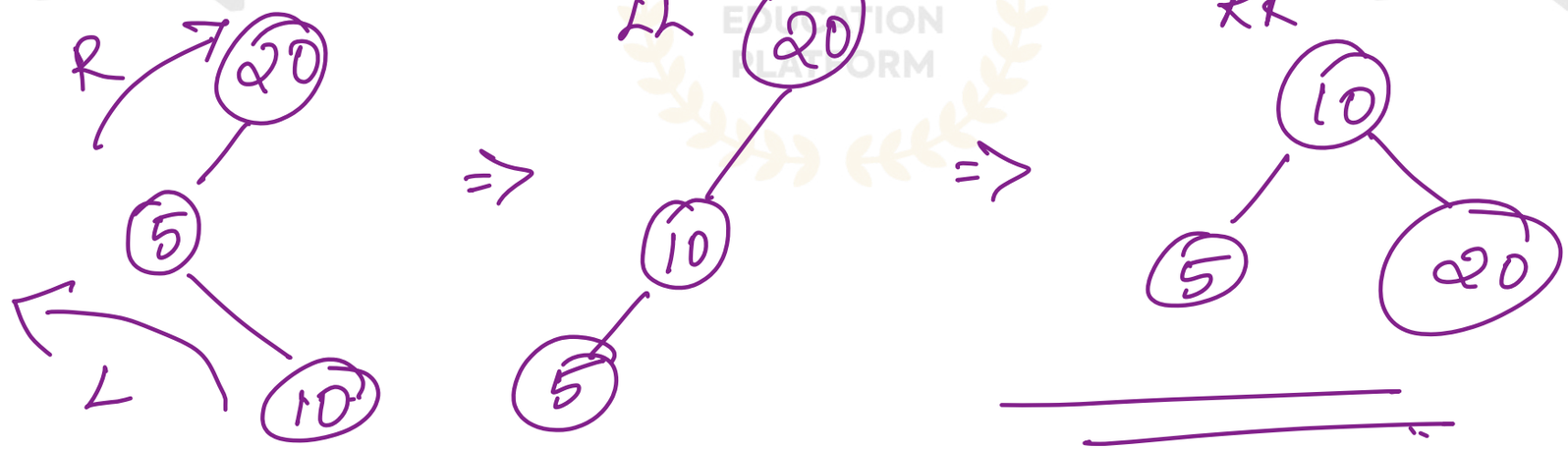
LR imbalance => LR rotation

Insertion Order : 3, 1, 2



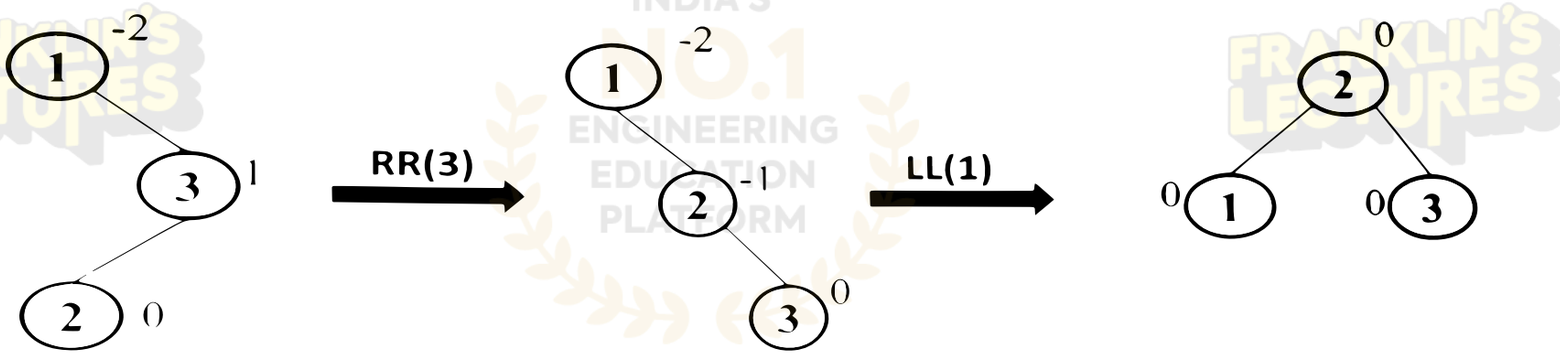
Ex:

Insert 10



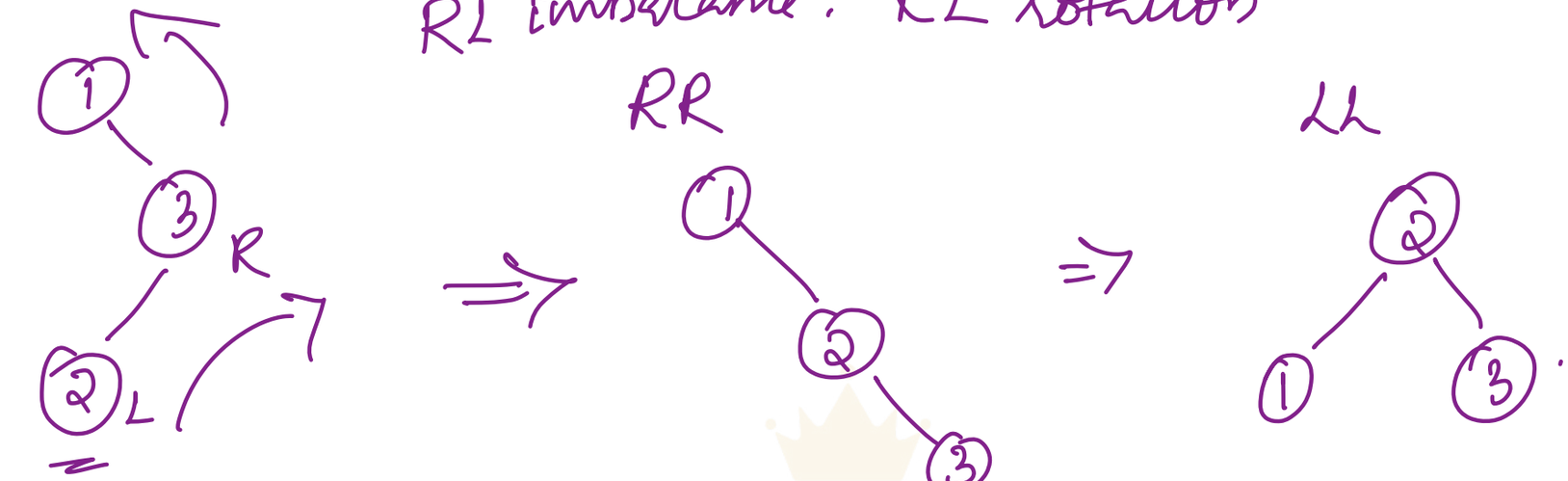
RIGHT -LEFT ROTATION(RL ROTATION)

- The RL rotation is the combination of single right rotation followed by single left rotation.



Insertion Order : 1,3,2

RL Imbalance : RL rotation



AVL TREE INSERTION ALGORITHM

bf: hlf − hrf

FRANKLIN'S LECTURES

1. Insert the node as the leaf node. Use BST insertion procedure
2. After insertion check the balance factor of every node
3. If the tree is imbalanced, perform the suitable rotation. Let z be the newly inserted node. x be the first unbalanced node on the path from z to root. y be the child of x on the path from z to root
   - If y is the left child of x and z is in the left subtree of y, then perform ~~RR~~ LL Rotation with respect to x.
   - If y is the right child of x and z is in the right subtree of y, then perform ~~LL~~ RR Rotation with respect to x.

- If y is the left child of x and z is in the right subtree of y, then perform LR Rotation.
- If y is the right child of x and z is in the left subtree of y, then perform RL Rotation.

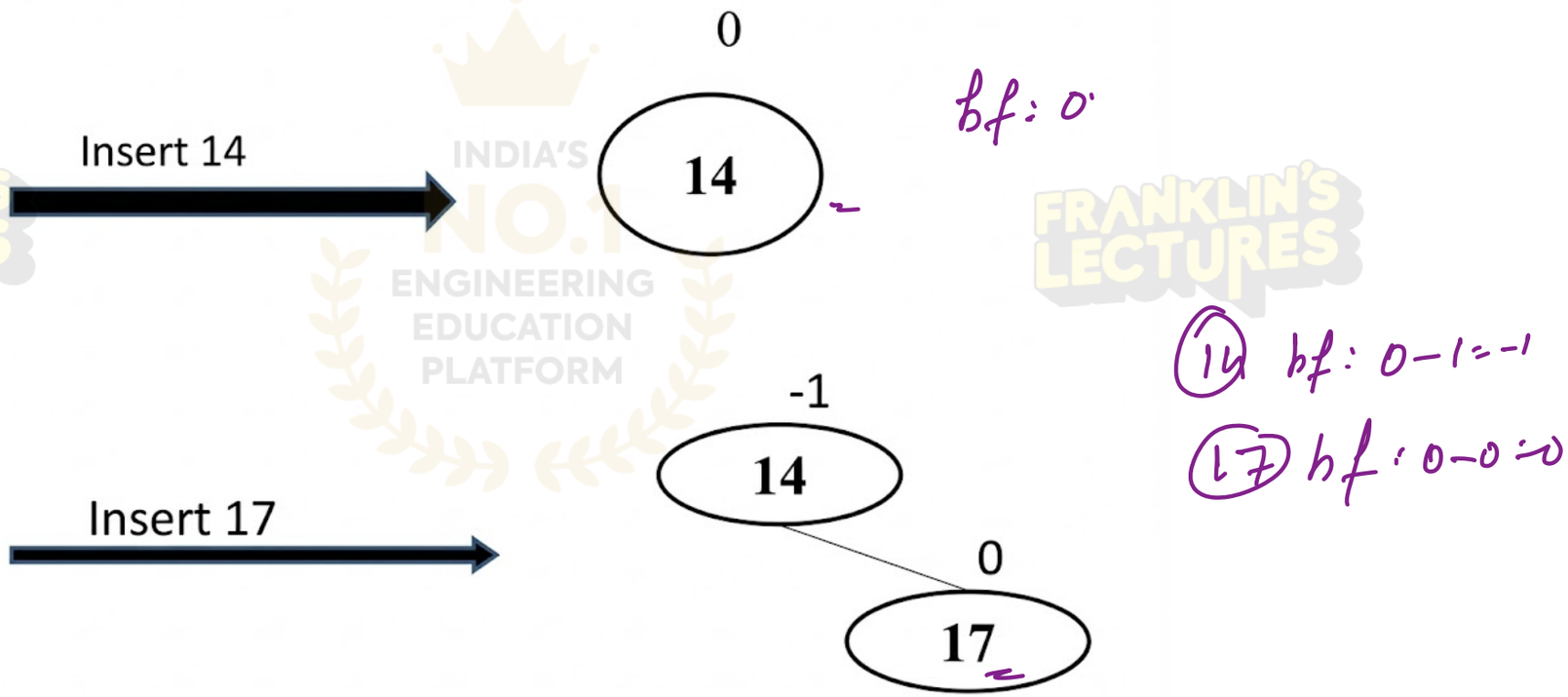
- Complexity of AVL tree insertion = O(log n)

  Where log n is the height of the tree.

- Examples:

1. Insert 14,17,11,7,53,4 and 13 in to an empty AVL Tree.

Answer:

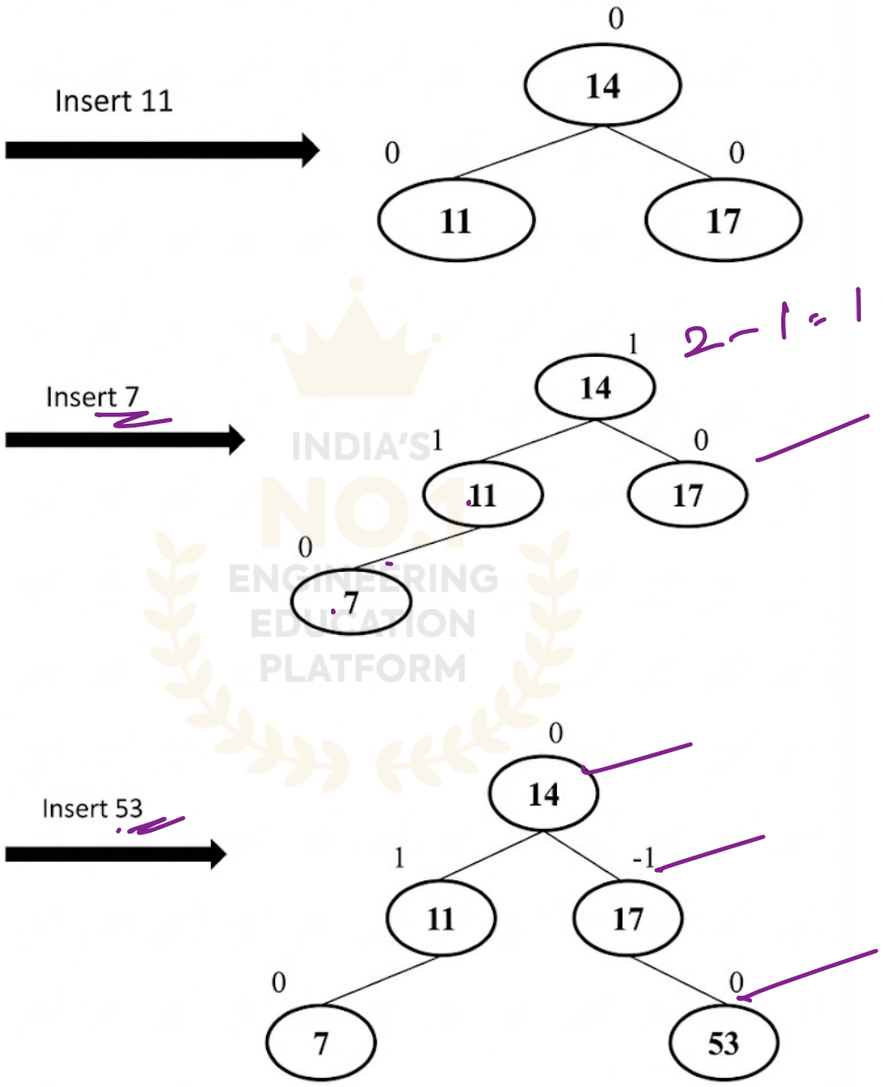


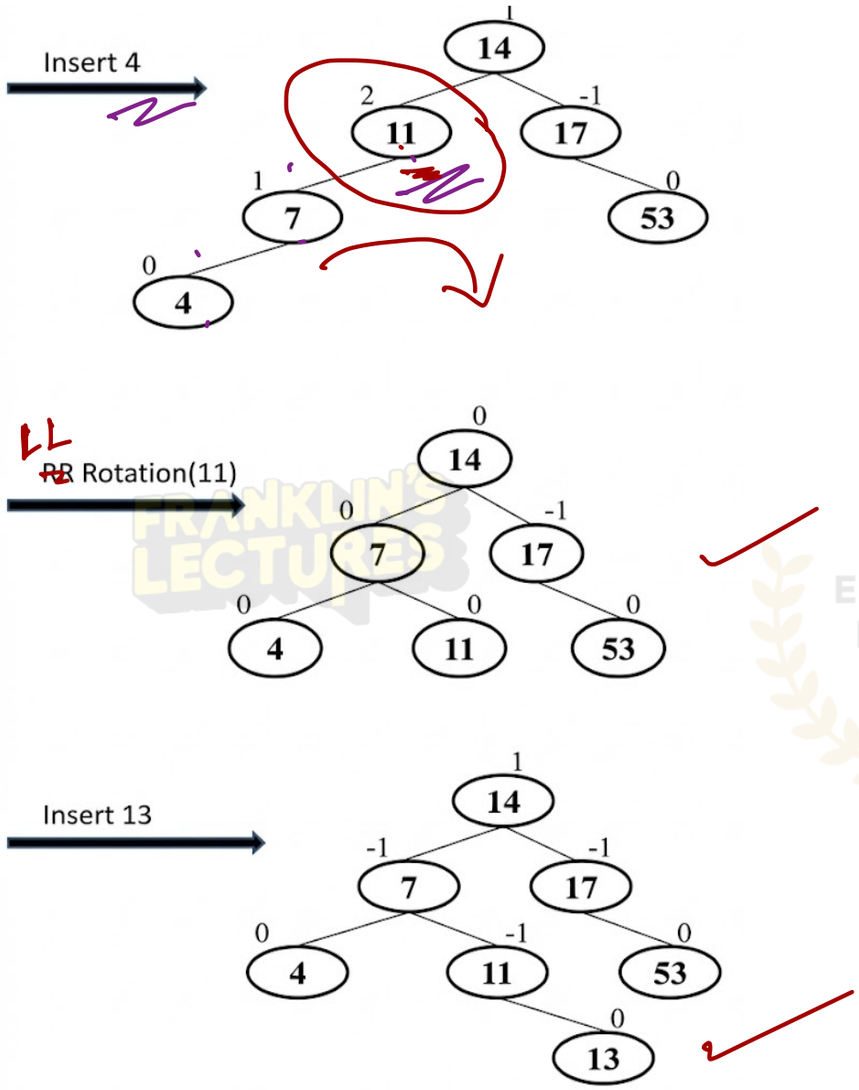
bf: 0

(14) bf: 0-1=-1

(17) bf: 0-0=0

14 bf : 1-1 = 0
17 bf : 0-0 : 0
11 bf : 0-0 = 0





The tree is now imbalanced because the balance factor of node 11 is 2.
Perform RR Rotation with respect 11

2. Insert numbers from 1 to 8 into an empty AVL tree

3. Insert 10,20,15,25,30,16,18 and 19 in to an empty AVL Tree.

AVL TREE DELETION ALGORITHM

Let w be the node to be deleted.

1. Delete w using BST deletion procedure
2. Starting from w, travel up and find the first unbalanced node.
   Let x be the first unbalanced node.
3. If balance factor(x)>1 then y=leftchild(x)
4. Else y=rightchild(x)
5. If y is the leftchild of x
   1. If balance factor(y)≥ 0 then z=leftchild(y)
   2. Else z=rightchild(y)

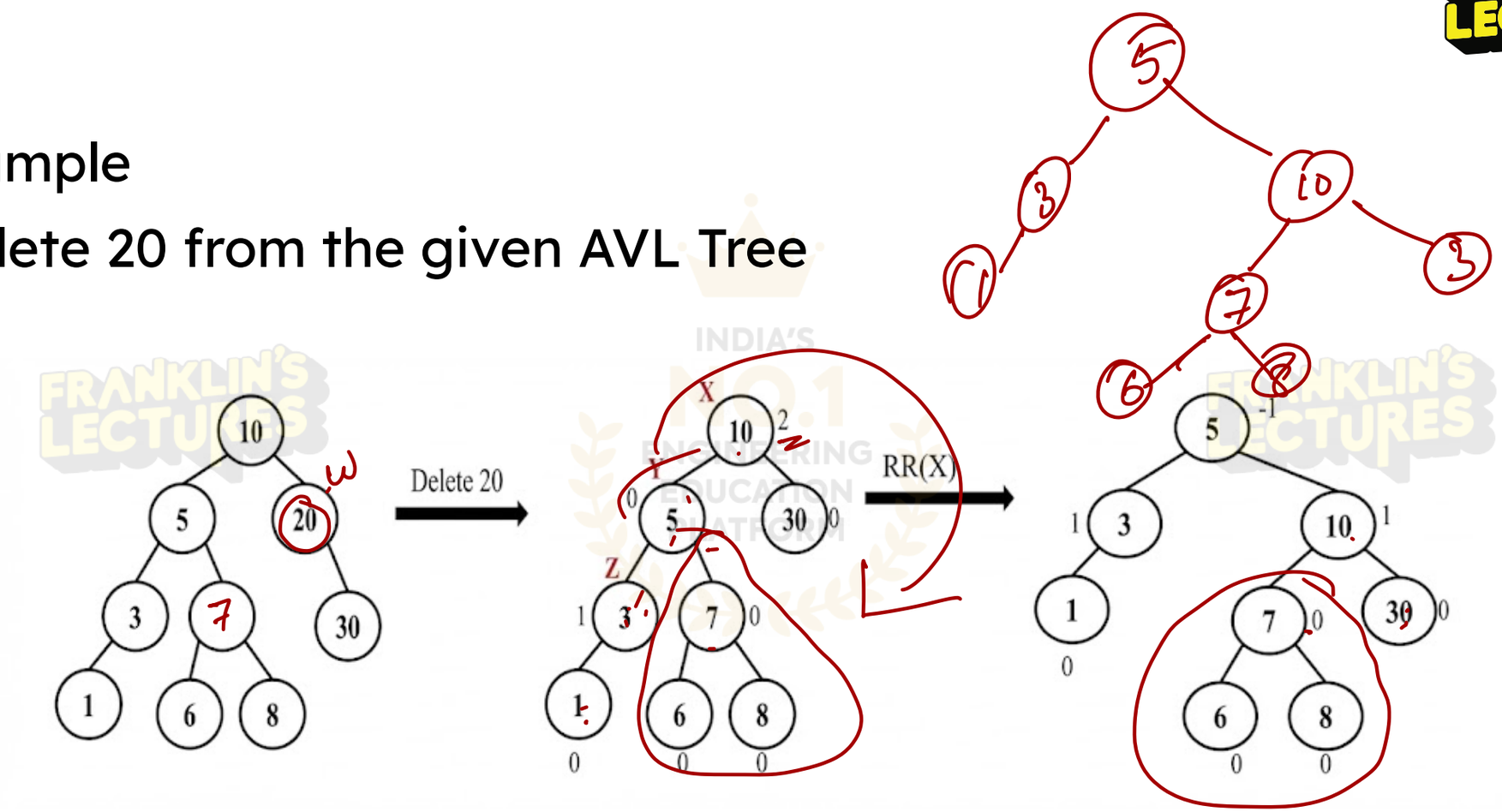
6. Else
    1. If balance factor(y) ≤ 0 then z=rightchild(y)
    2.Else z=leftchild(y)
7. If y is the left child of x and z is the left child of y, then perform RR Rotation with respect to x.
8. If y is the right child of x and z is the right child of y, then perform LL Rotation with respect to x.
9. If y is the left child of x and z is the right child of y, then perform LR Rotation with respect to x.
10. If y is the right child of x and z is the left child of y, then perform RL Rotation with respect to x.

- Complexity of AVL tree deletion = O(log n)

  Where log n is the height of the tree.

- Example

1. Delete 20 from the given AVL Tree



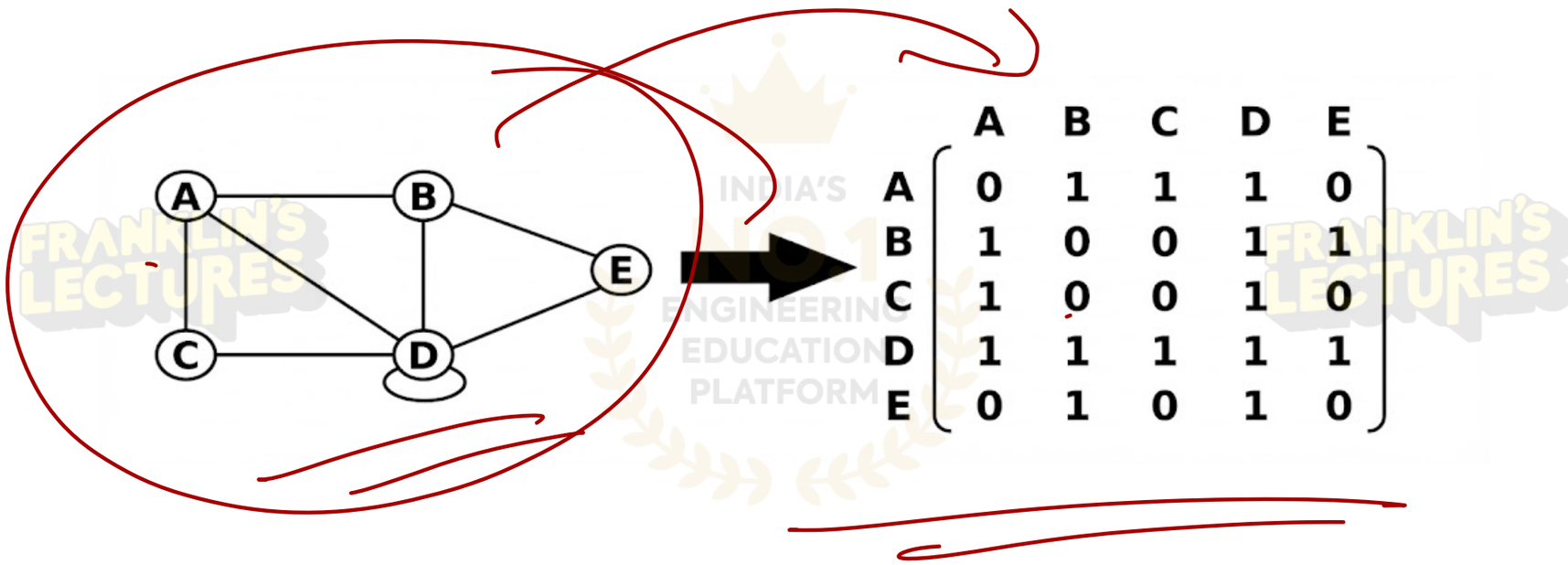
GRAPHS

- Graph is a data structure that consists of following two components:
  - A finite set of vertices(nodes).
  - A finite set of edge (ordered pair of the form (u,v)
- A graph G = (V , E), where V is a set of vertices and E is a set of edges.
- Representations of graph.
  - Adjacency Matrix
  - Adjacency List

ADJACENCY MATRIX

FRANKLIN'S LECTURES

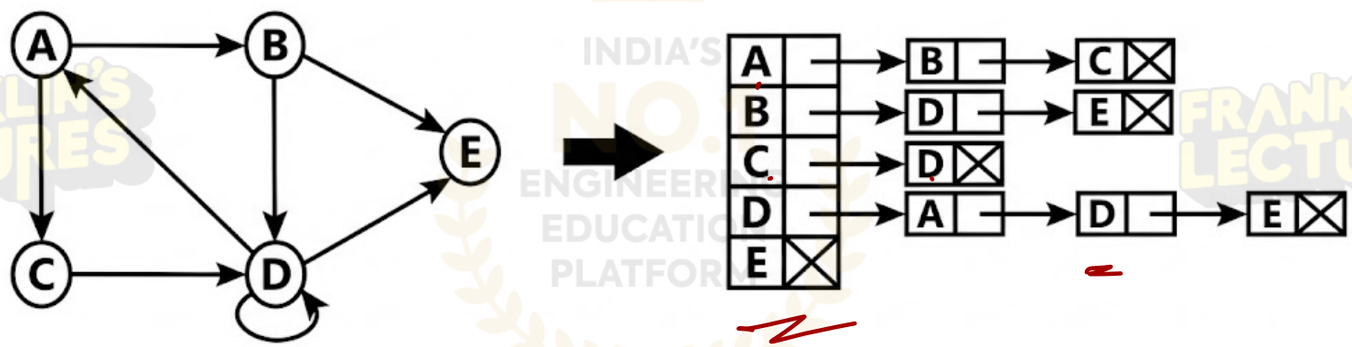
- Adjacency Matrix is a 2D array(say adj[][]) of size |V| x |V| where |V| is the number of vertices in a graph.
- If adj[i][j] = 1, then there is an edge from vertex i to vertex j.
- Adjacency matrix for undirected graph is always symmetric.
- Adjacency Matrix is also used to represent weighted graphs.
  - If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w.


A
B
C
D
E
A B C D E
A 0 1 1 1 0
B 1 0 0 1 1
C 1 0 0 1 0
D 1 1 1 1 1
E 0 1 0 1 0

ADJACENCY LIST:

- An array of linked lists is used.
- Size of the array is equal to number of vertices.
- An entry array[i] represents the linked list of vertices adjacent to the ith vertex.
- This representation can also be used to represent a weighted graph. The weights of edges can be stored in nodes of linked lists.

A → B A → C	B → D B → E	C → D	D → A D → D D → E	E → X



TYPES OF GRAPH:

- Undirected Graph: A graph with only undirected edges.
- Directed Graph: A graph with only directed edges.
- Directed Acyclic Graphs (DAG): A directed graph with no cycles.
- Cyclic Graph: A directed graph with at least one cycle.
- Weighted Graph: It is a graph in which each edge is given a numerical weight.
- Disconnected Graphs: An undirected graph that is not connected.

FRANKLIN'S LECTURES

THANK YOU