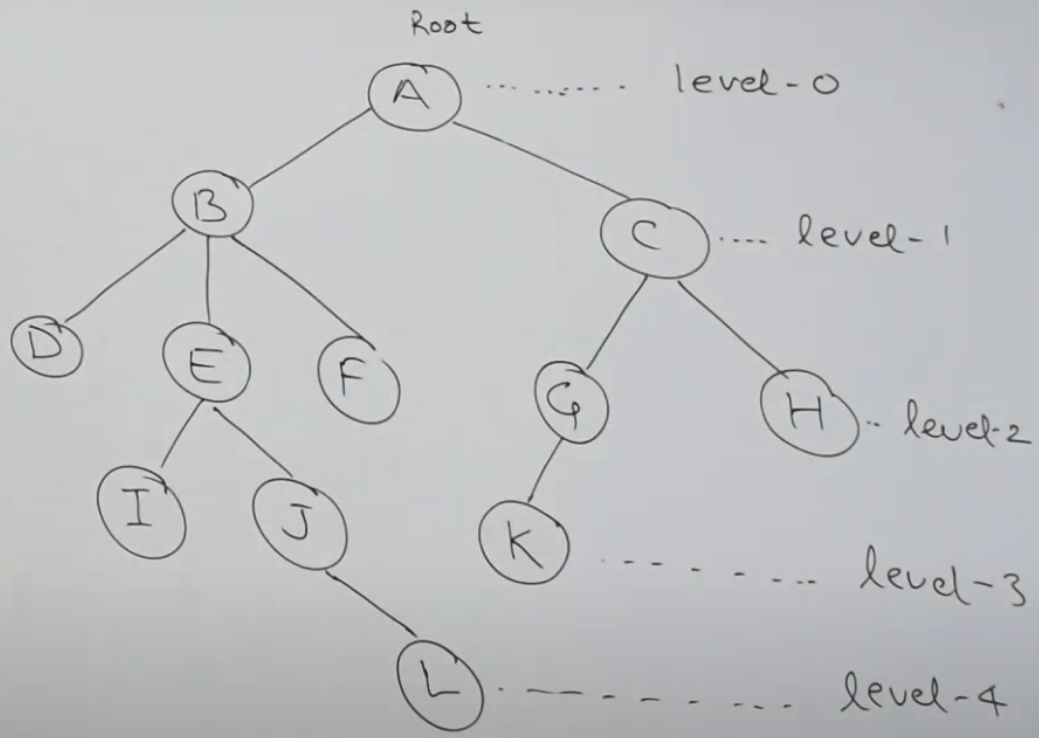
Trees

Tree is a non-linear hierarchical data structure.

Root
Parent
child
Siblings
Ancestor
Descendant
Degree
Height
Depth
Internal Node
Leaf Node
subtree
Forest



Binary Tree (Introduction) | Min and Max nodes at height h | Data Structure

Binary Tree (0, 1, 2) Atmost 2 child.

$n = 2^{h+1} - 1$

$n+1 = 2^{h+1}$

$h+1 = \log_2(n+1)$

$h = \log_2(n+1) - 1$

$h = 1$

$n = 2$

$h = n - 1$

$n = h + 1$

$\log(n+1) - 1 \leq h \leq n - 1$

$O(\log n) \leq O(n)$

$1 + 2 + 4 + 8 + \ldots$

$1 + 2 + 2^2 + 2^3 + \ldots + 2^h$

$h+1 \leq n \leq 2^{h+1} - 1$

$h = 1$

$h = 3$

$h = 2$

$n = 7$

$h = 3$

$h = 15$

$\frac{a(r^n - 1)}{r - 1}$ $r > 1$

$r = \frac{2}{1} = 2$

$a = 1$

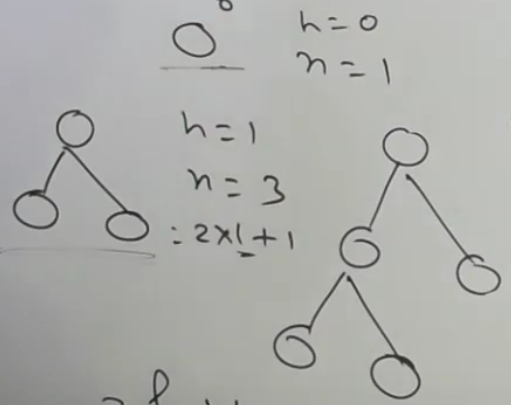
$\frac{1(2^{h+1} - 1)}{2 - 1}$

$n = 2^{h+1} - 1$

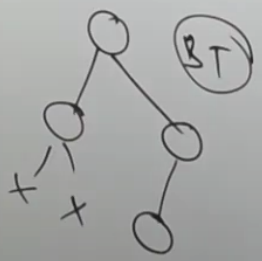
15:11 / 15:47

Types of Binary Tree | Full BT | Complete Binary Tree | Almost Complete Binary Tree | Data Structure

1. Full Binary Tree / strict B.T. (0, 2)
2. Complete Binary Tree
   * All leaf nodes are at same level.
   * Depth of all leaf nodes are same.



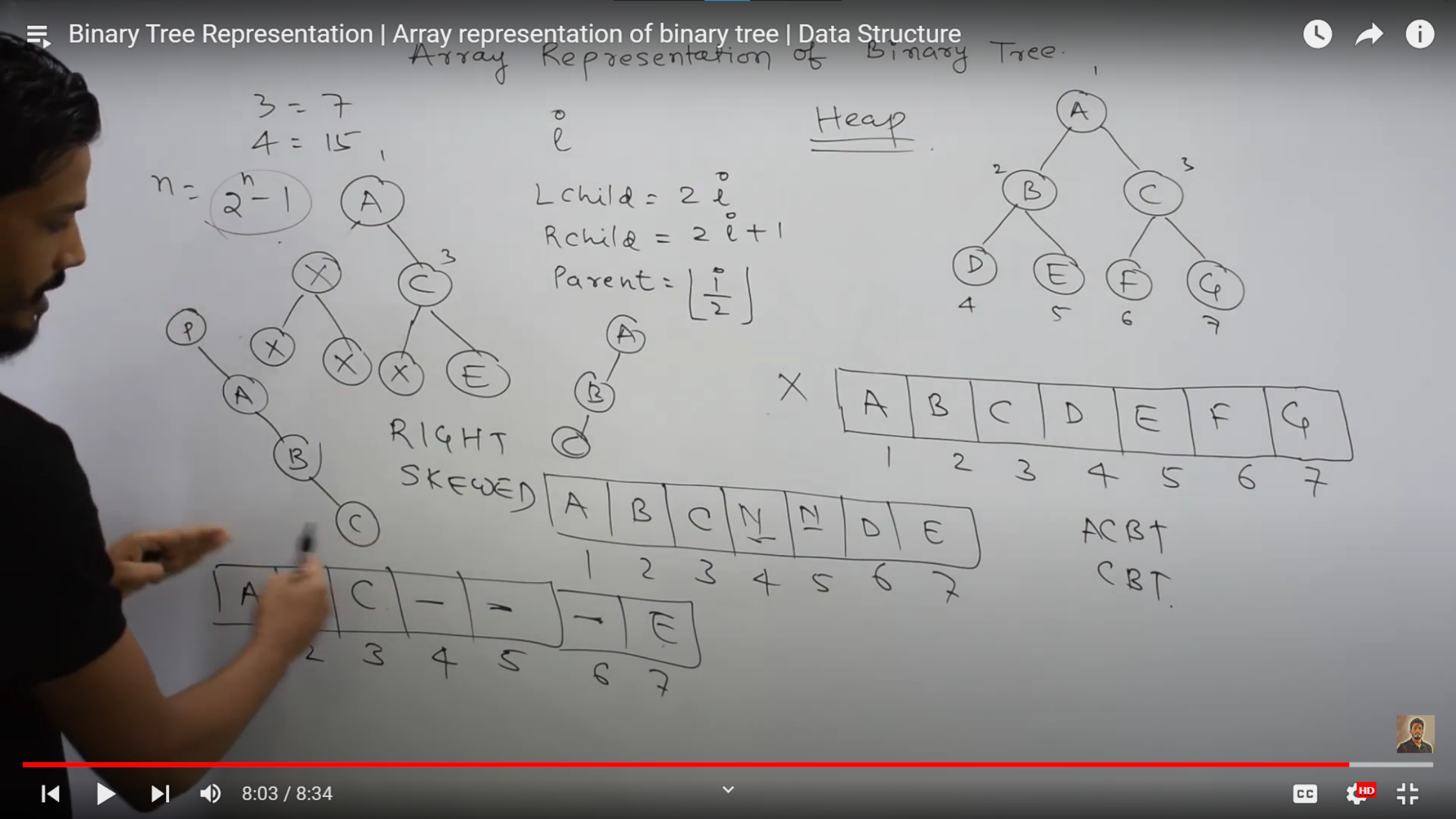
	B.T	FBT	CBT
min nodes	$n = h+1$	$n = 2h+1$	$n = 2^{h+1}-1$
max nodes	$n = 2^{h+1}-1$	$n = 2^{h+1}-1$	$n = 2^{h+1}-1$
min height	$h = \log(n+1)-1$	$h = \log(n+1)-1$	$h = \log(n+1)-1$
max height	$h = n-1$	$h = \frac{n-1}{2}$	$h = \log(n+1)-1$



Leaf nodes = 2^h

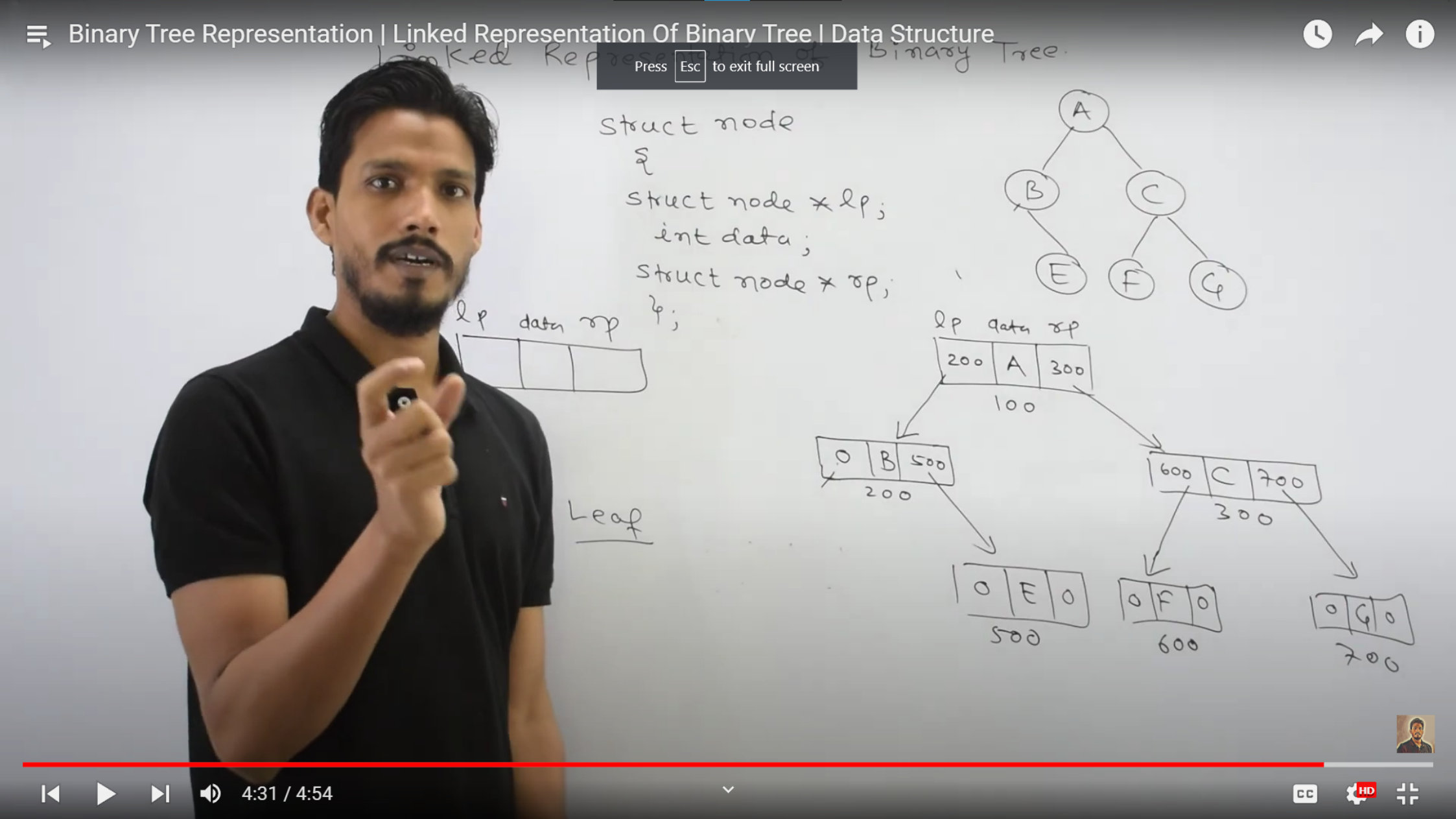
internal nodes = $2^h - 1$

Binary Tree Representation | Array representation of binary tree | Data Structure



Binary Tree Representation | Linked Representation Of Binary Tree | Data Structure

Press Esc to exit full screen



4:31 / 4:54

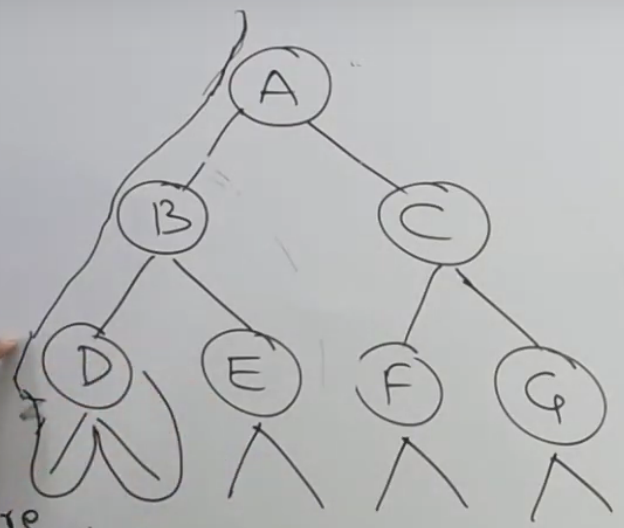
Binary Tree Traversals Example 1 [Shortcut] (Preorder, Inorder and Postorder) | Data structures

TRAVERSAL

Preorder : Root, Left, Right

Inorder : Left, Root, Right

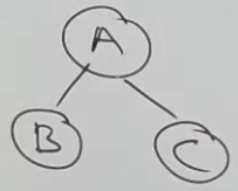
Postorder : Left, Right, Root



1) Pre : A B D

2) I : D

3) Pos



Pre : A B C

In : B A C

Post : B C A

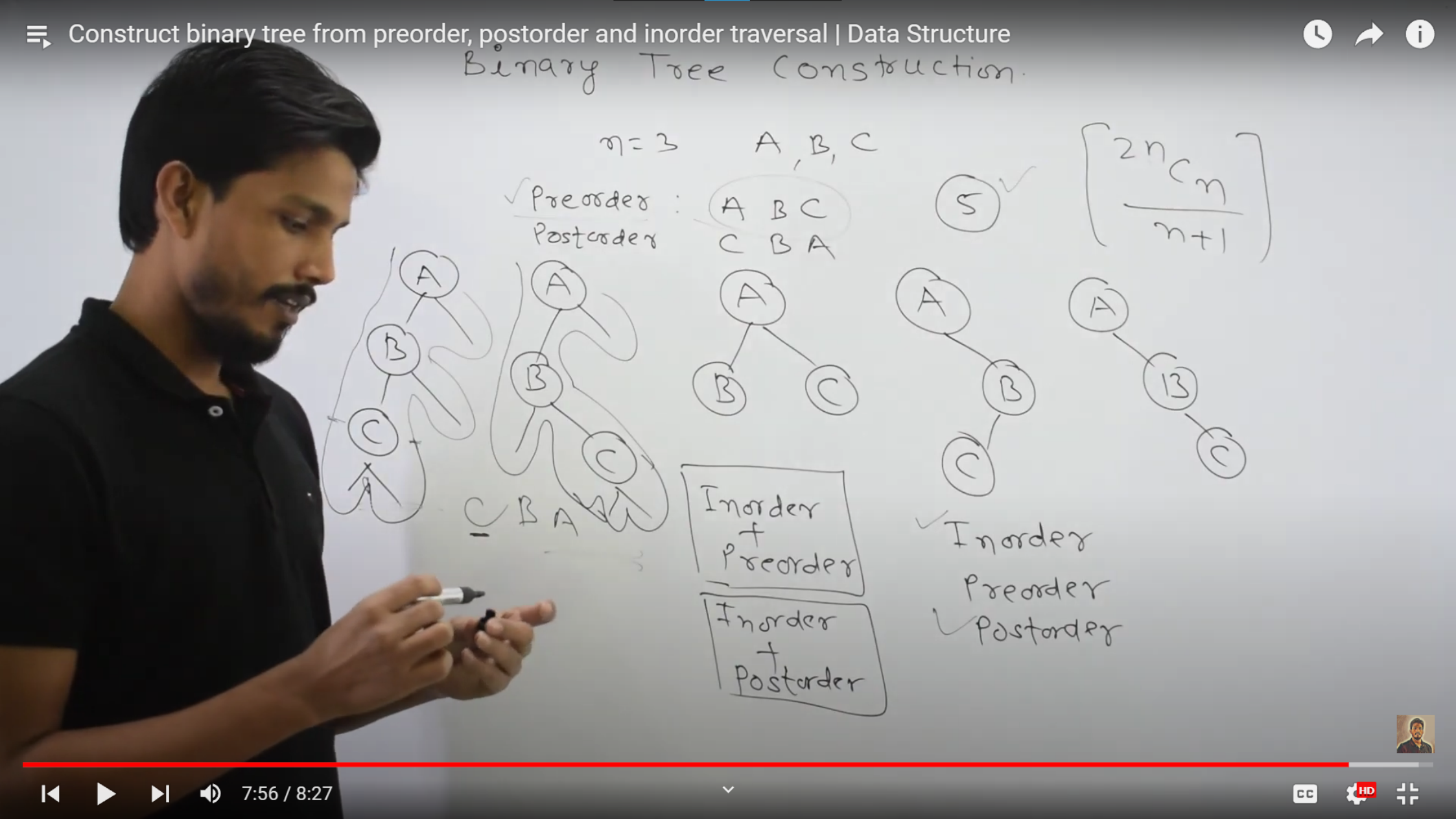
Pre : A (B D E) (C F G)

Inorder : (D B E) A (F C G)

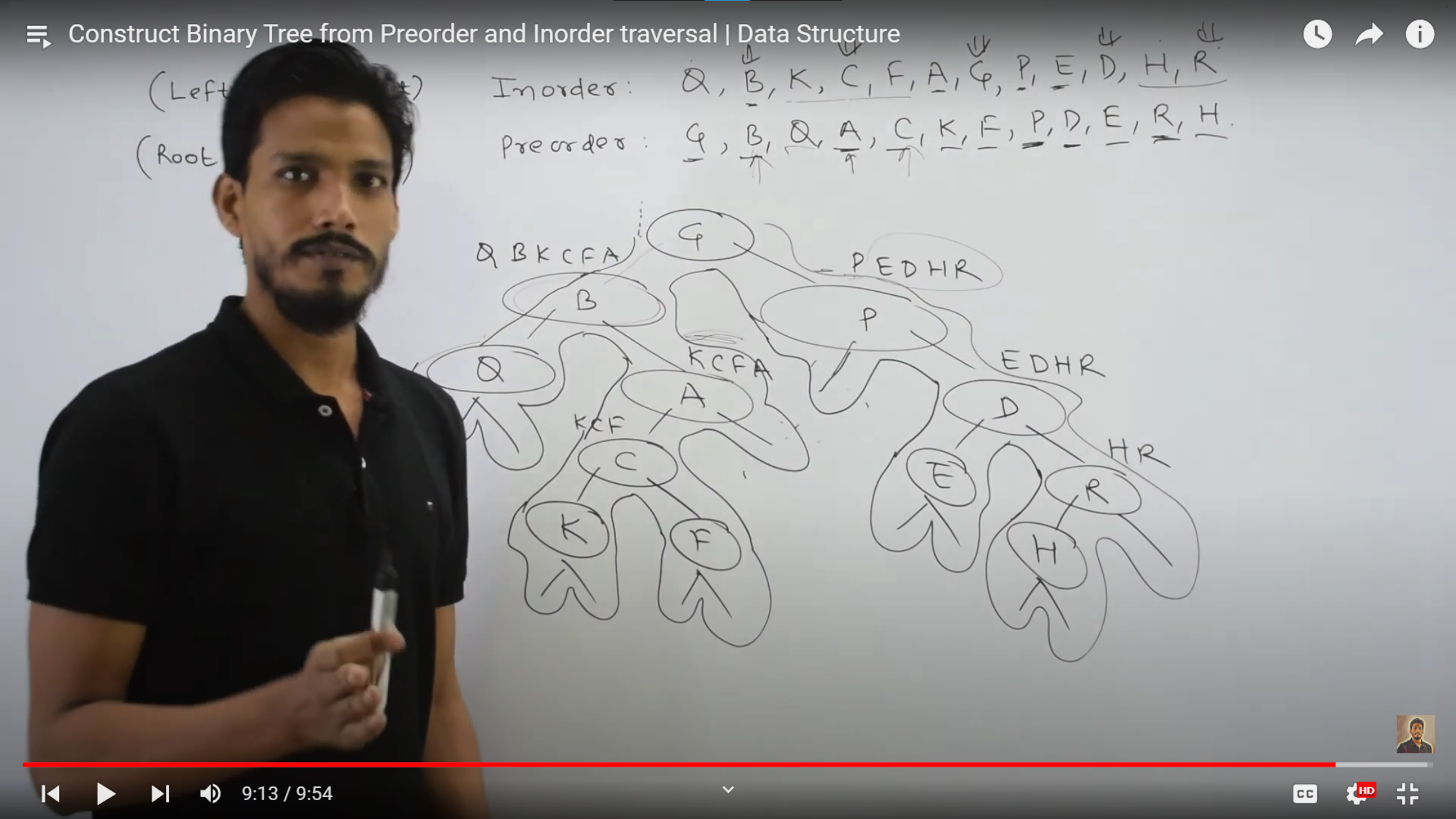
Post : (D E B) (F G C) A

8:36 / 11:52

Construct binary tree from preorder, postorder and inorder traversal | Data Structure



Construct Binary Tree from Preorder and Inorder traversal | Data Structure



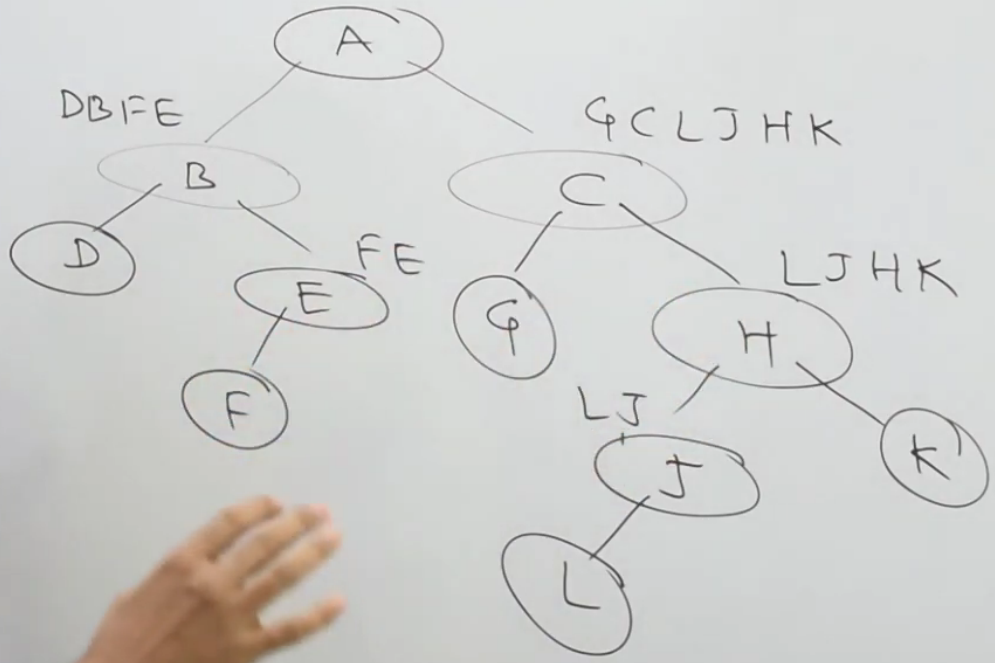
9:13 / 9:54

Root, Right)

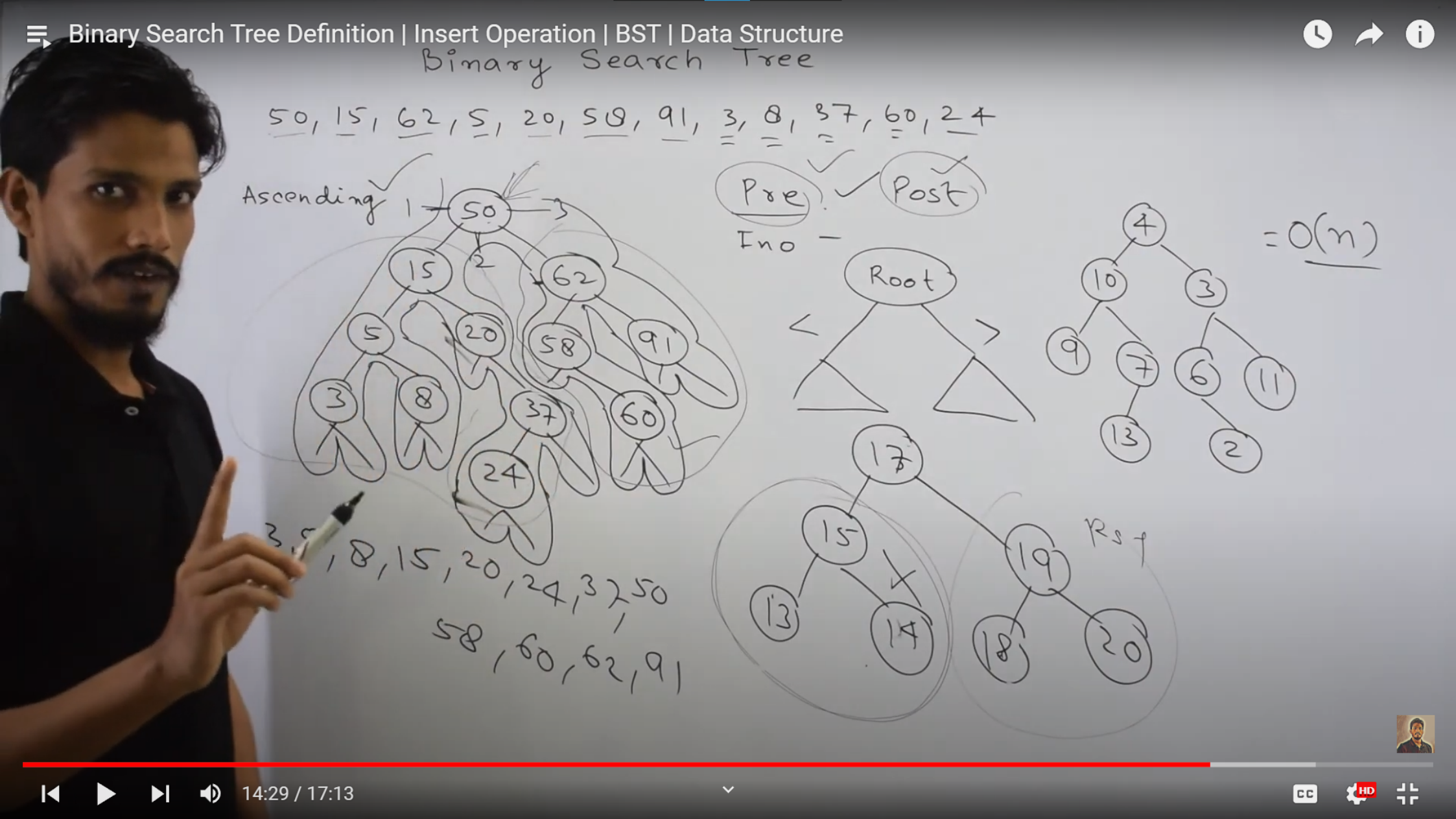
ght, Root)

Inorder : (D B F E) A (G C) L J H K

Postorder : D F E B G L J K H C A

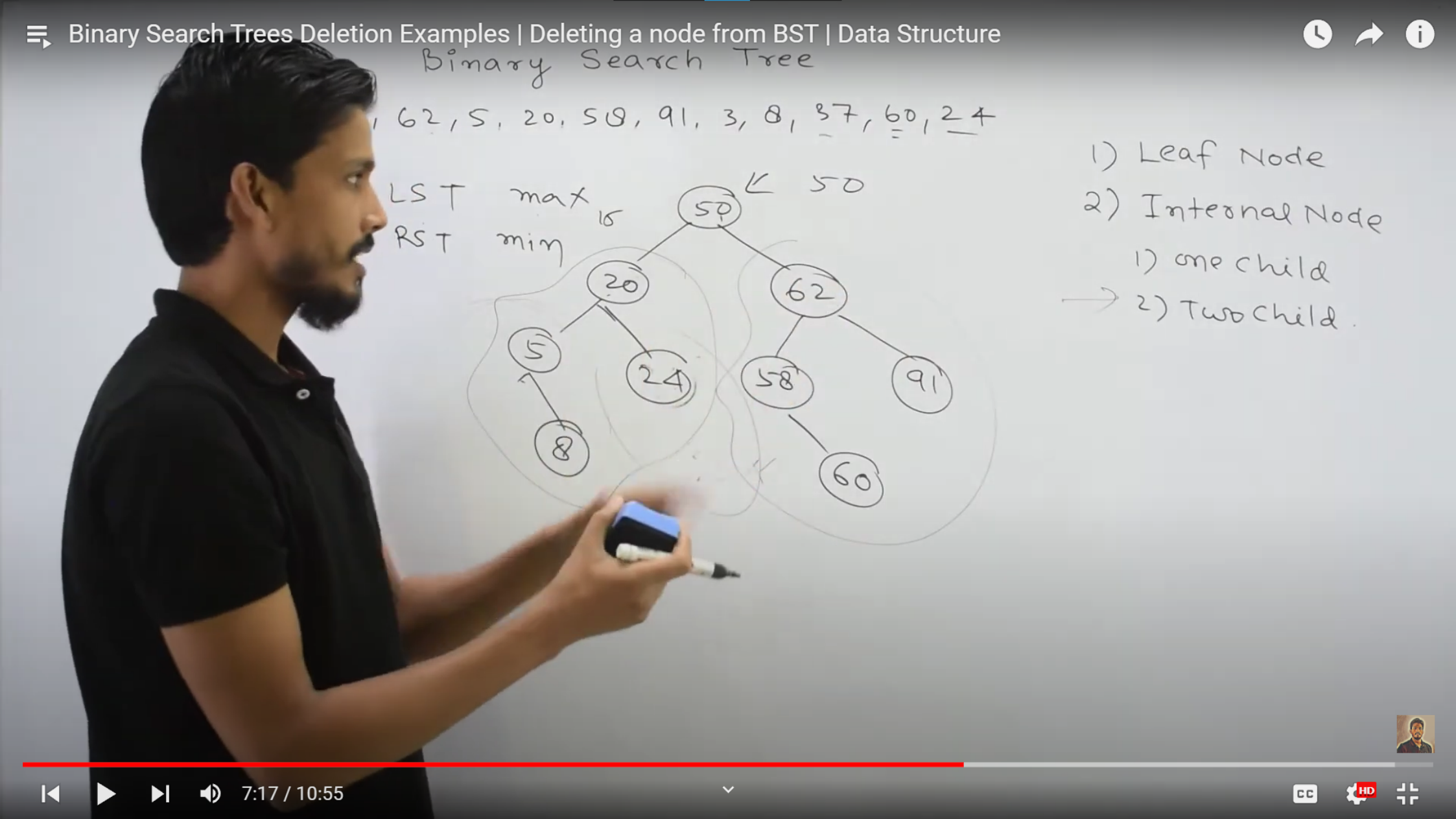


Binary Search Tree Definition | Insert Operation | BST | Data Structure



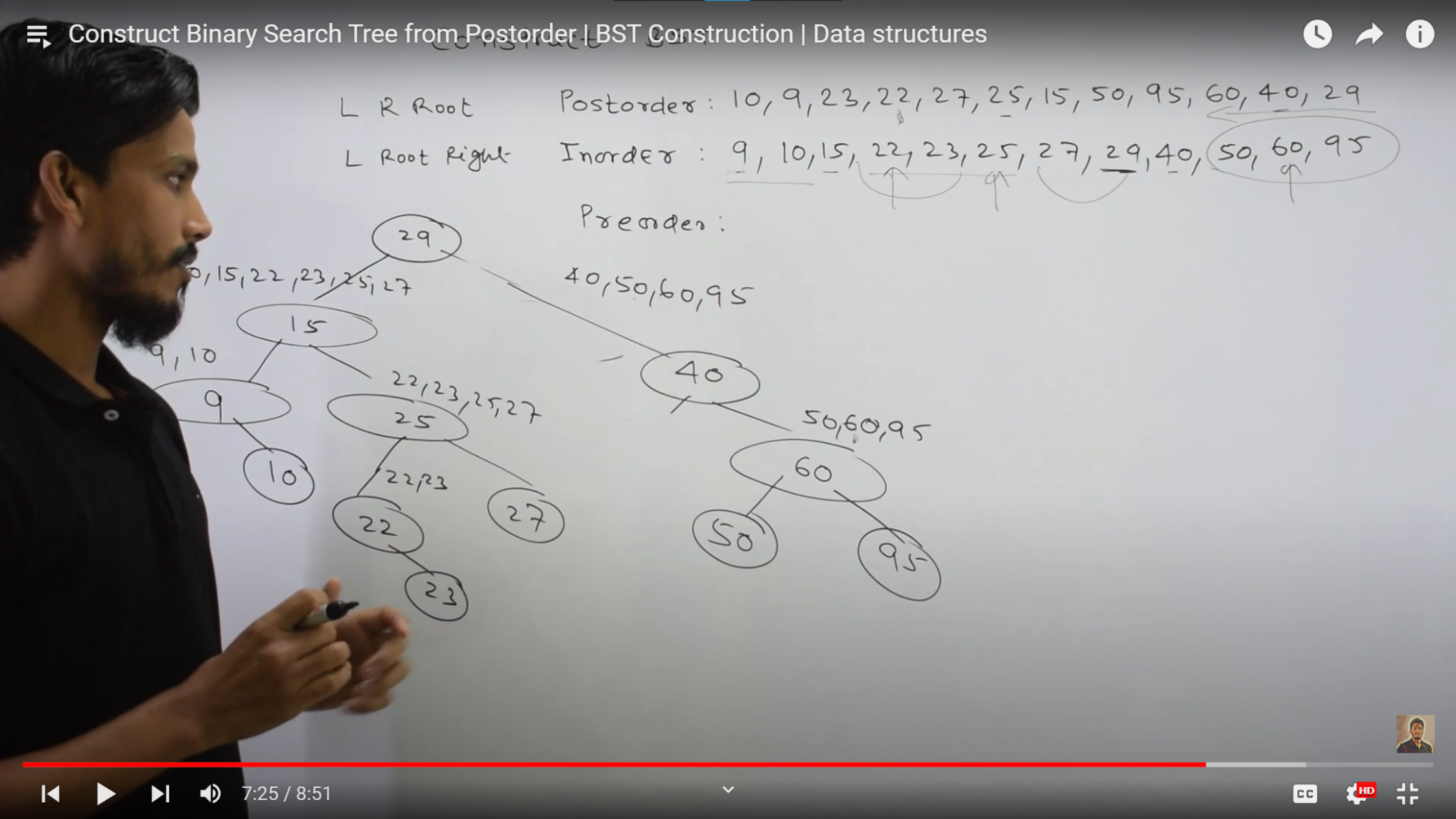
14:29 / 17:13

Binary Search Trees Deletion Examples | Deleting a node from BST | Data Structure



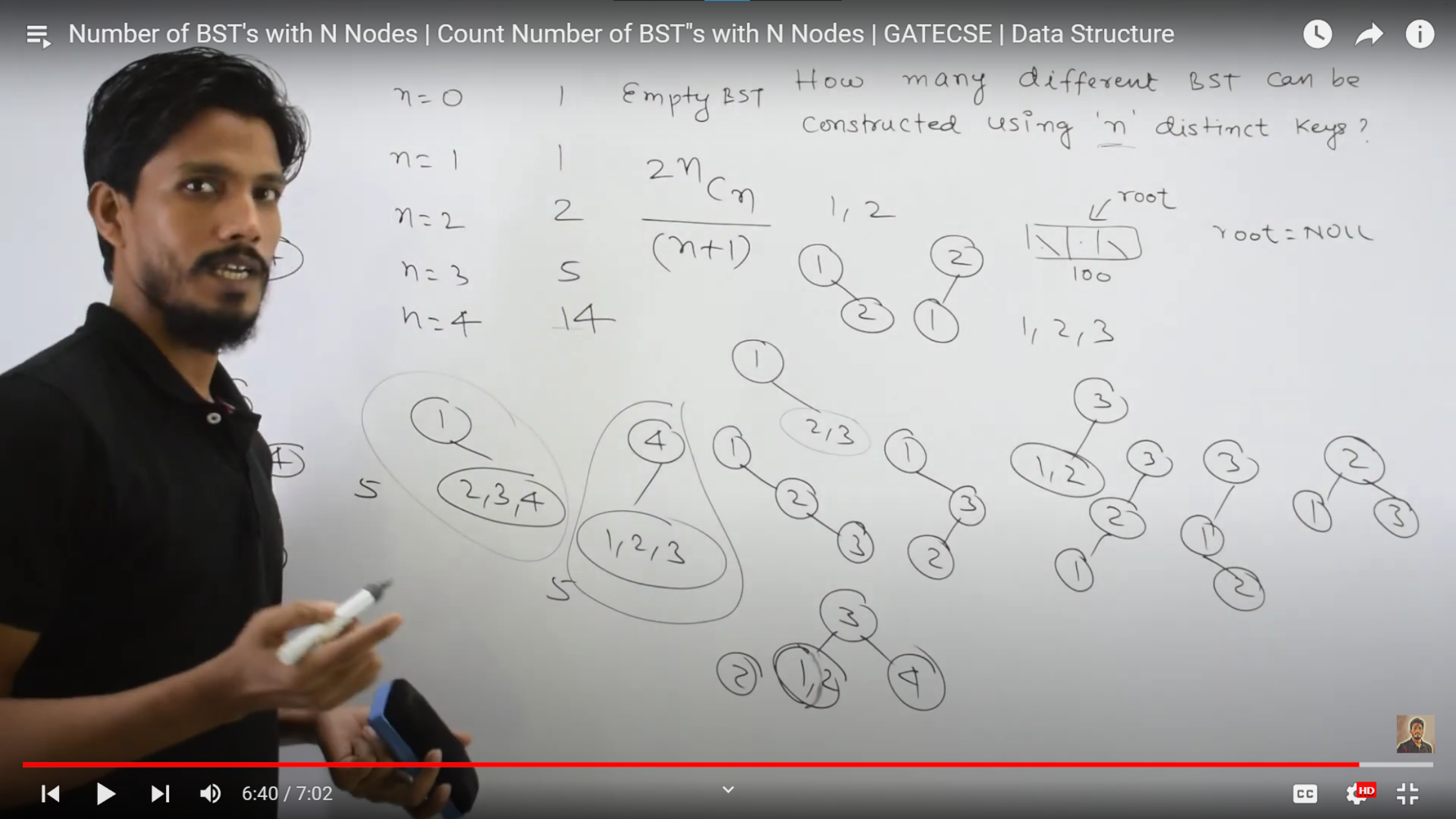
7:17 / 10:55

Construct Binary Search Tree from Postorder | BST Construction | Data structures



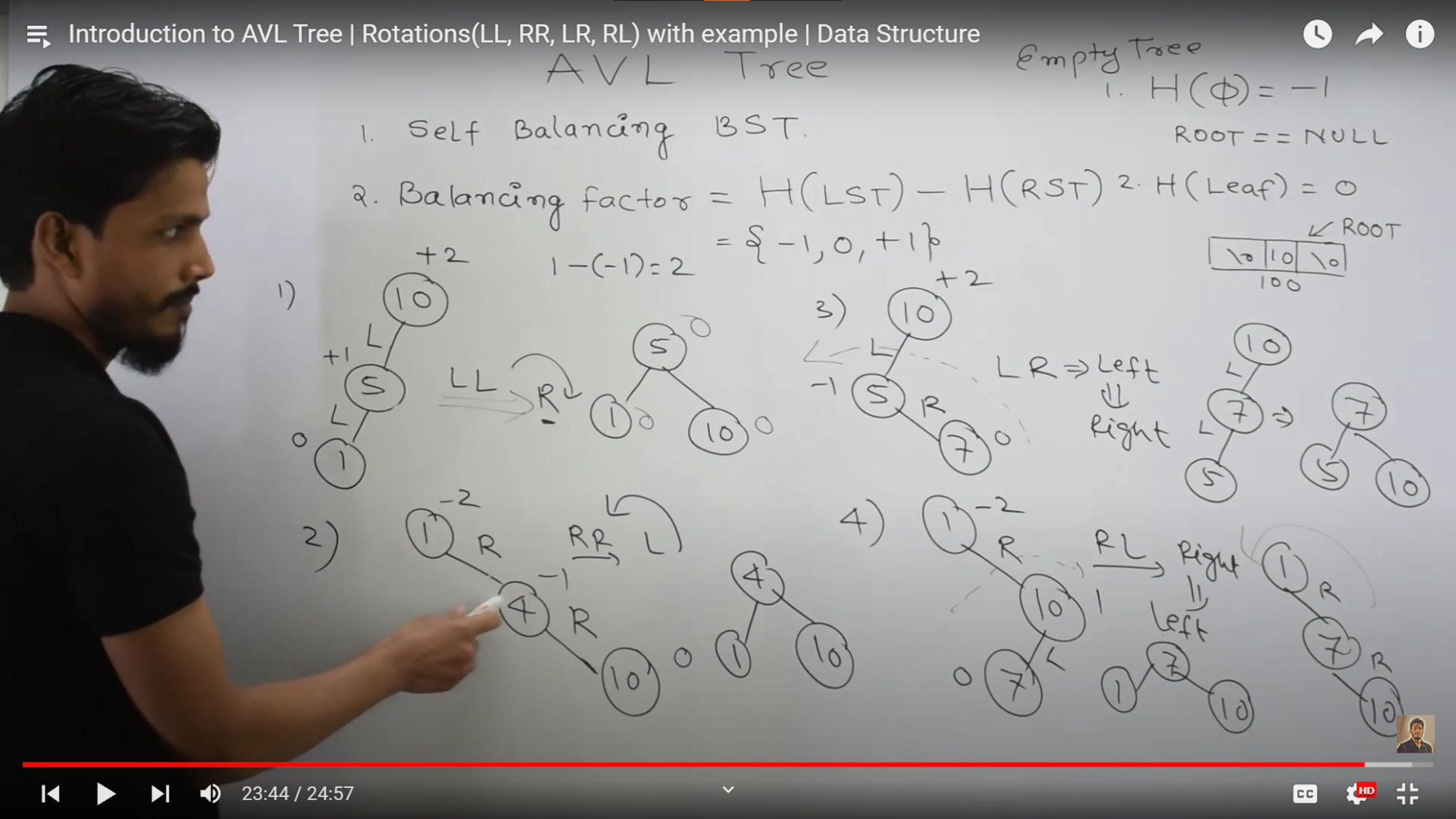
7:25 / 8:51

Number of BST's with N Nodes | Count Number of BST"s with N Nodes | GATECSE | Data Structure



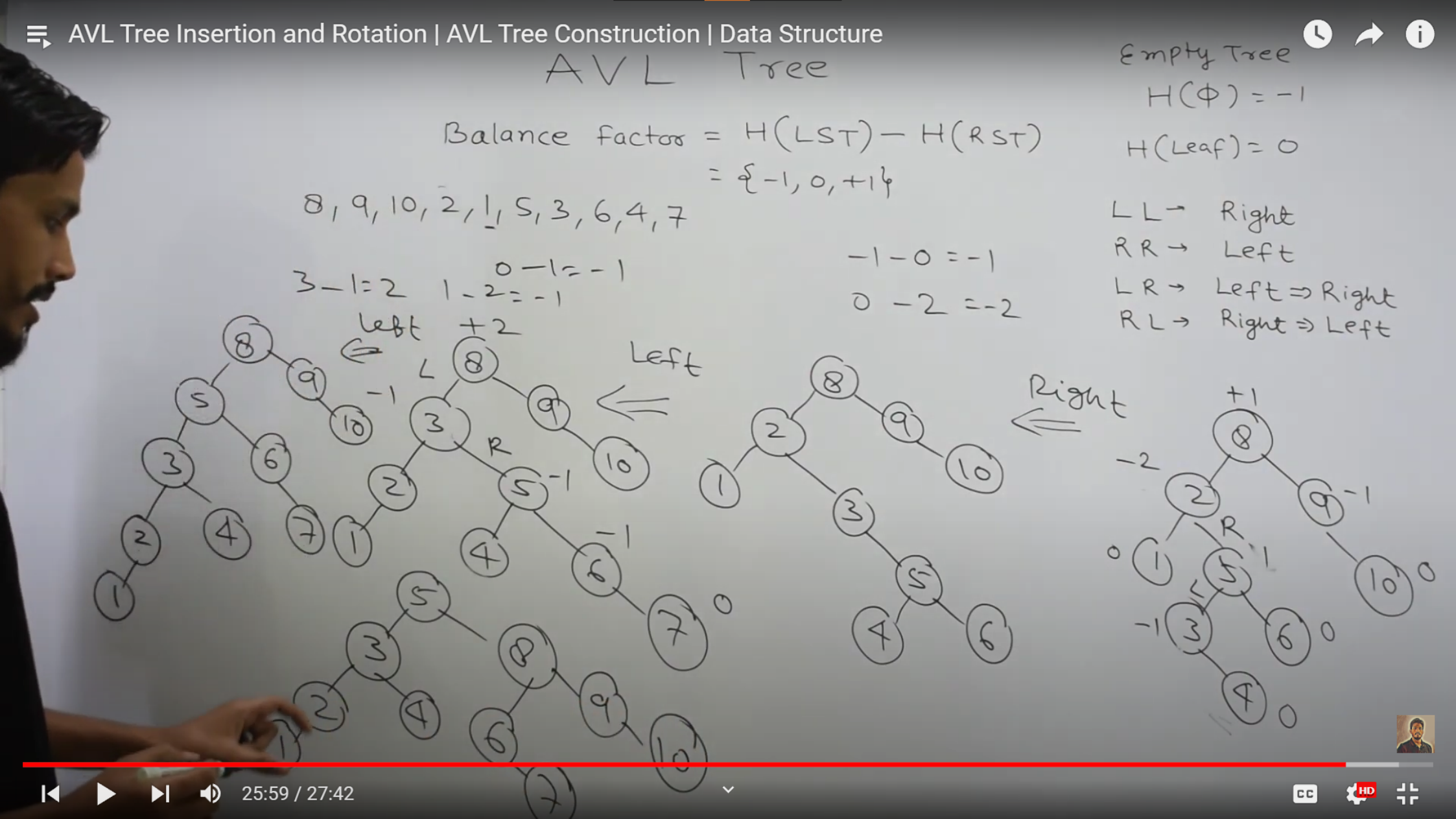
6:40 / 7:02

Introduction to AVL Tree | Rotations(LL, RR, LR, RL) with example | Data Structure



23:44 / 24:57

AVL Tree Insertion and Rotation | AVL Tree Construction | Data Structure



25:59 / 27:42

Introduction to Red Black Tree | Properties of Red Black trees | RB Tree | Data structure

Red Black Tree

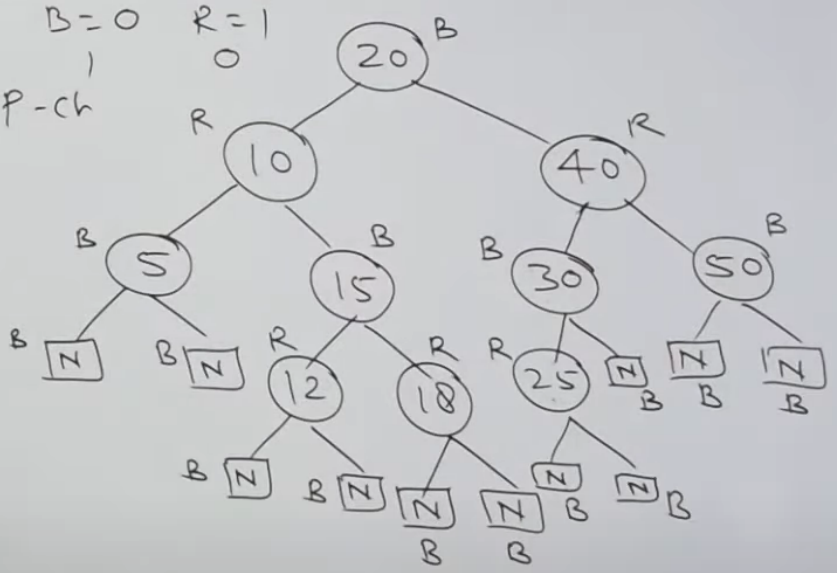
RB Tree is a Binary Search Tree with one extra bit of storage per node.

lp	Color	Key	rp

1. Every node is either Red or Black.
2. The root is black.
3. Every Leaf (NIL) is black.
4. If a node is Red, then both its children are black.
5. For each node, all paths from node to descendant leaves contain the same no. of black nodes.

B=0 R=1

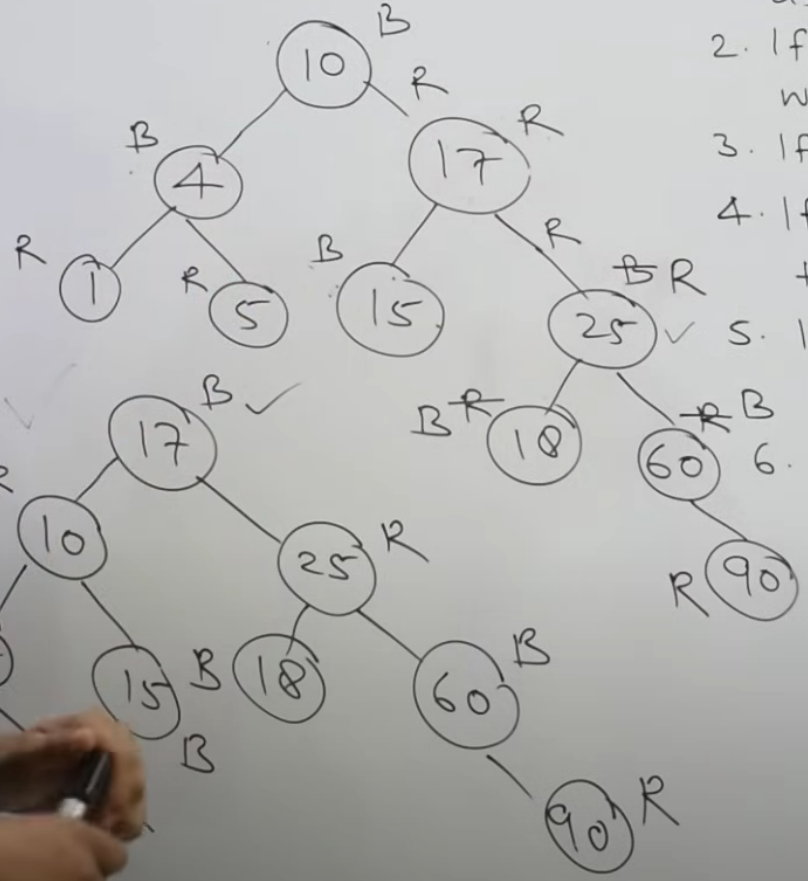
No R-R P-ch



8:09 / 15:05

RB Tree Insertion

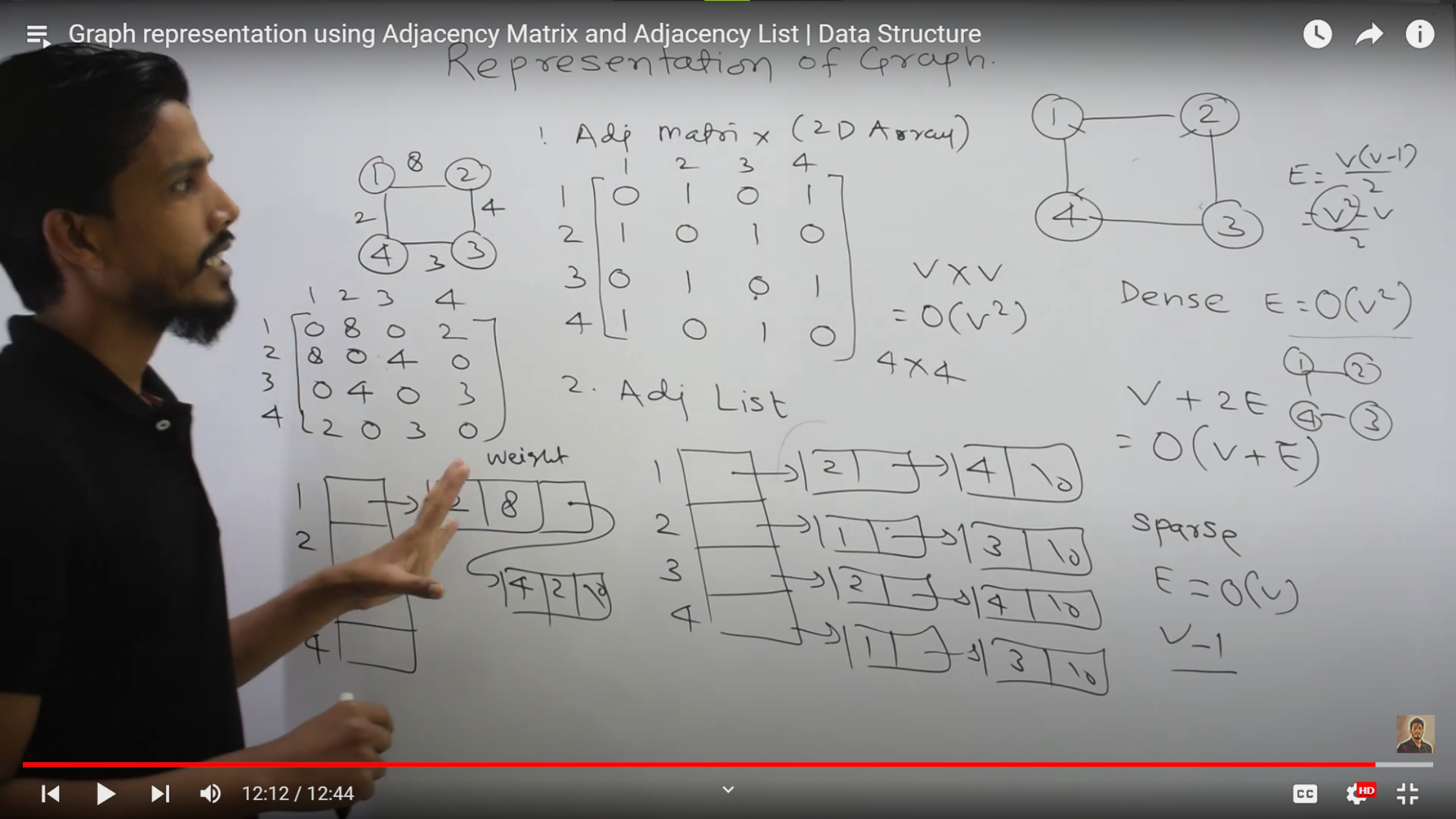
10, 18, 5, 4, 15, 17, 25, 60, 1, 90



1. If tree is empty then create newnode as root node with color Black.
2. If tree is not empty then insert newnode with color Red.
3. If the parent of newnode is Black then exit.
4. If the parent of newnode is Red then check the color of parent's sibling of newnode.
5. If it's color is Black or NULL then make suitable rotation & Recolor it.
6. If it's color is Red then Recolor.

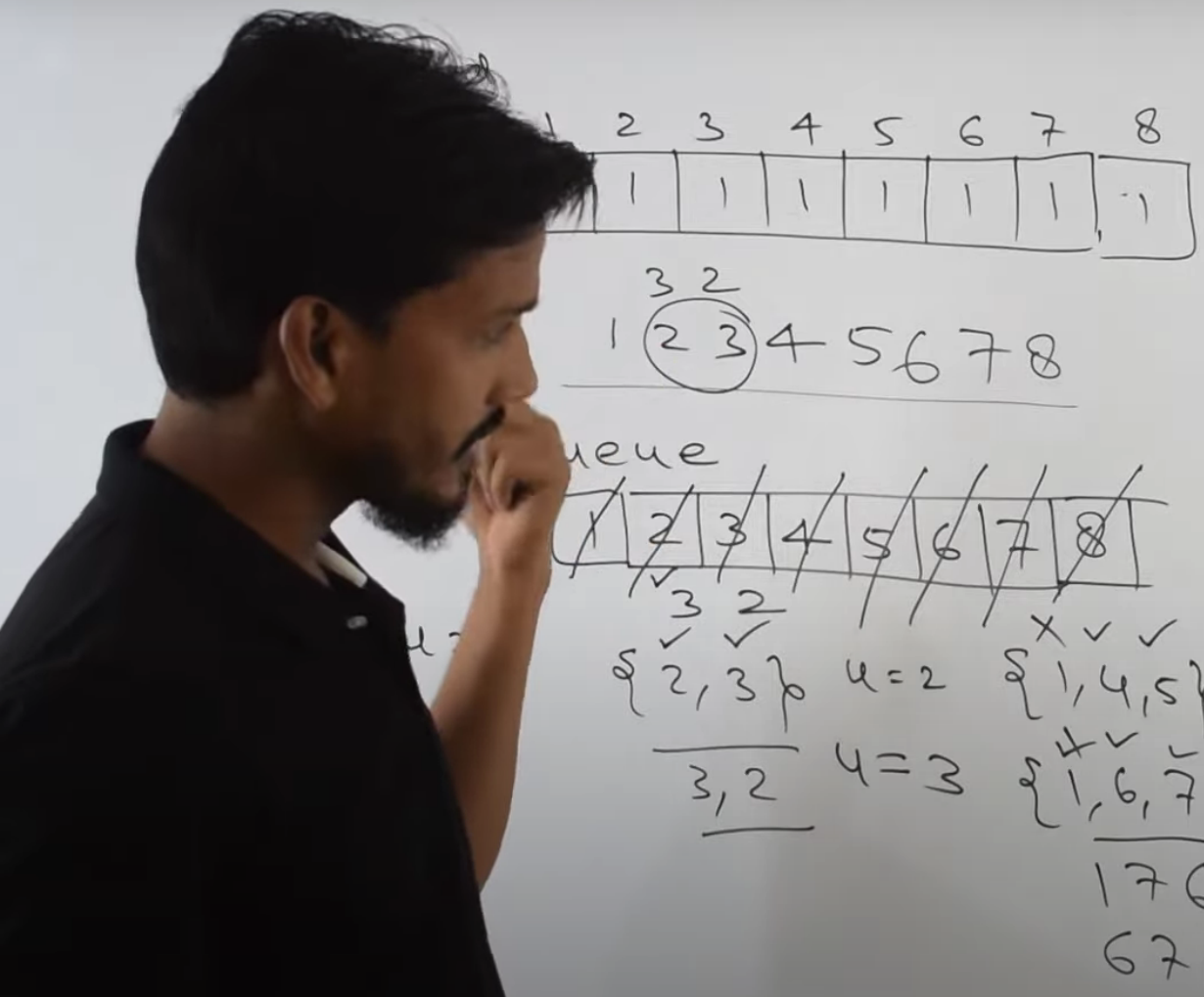
1. Root is always Black.
2. No two adjacent Red nodes
3. No. of Black nodes in every path are same.

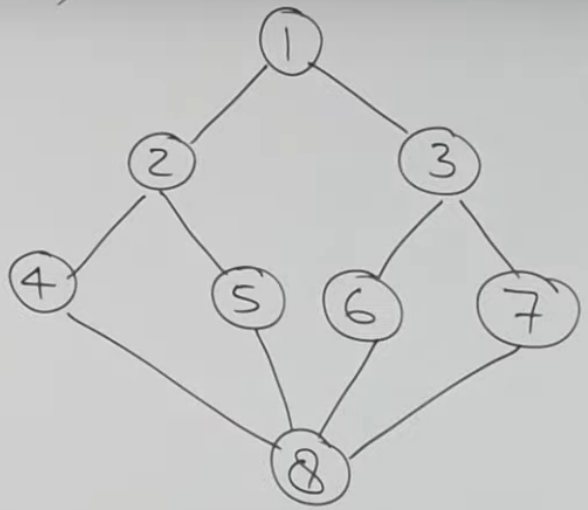
Graph representation using Adjacency Matrix and Adjacency List | Data Structure



Breadth First Search (BFS) With Time Complexity| Graph Traversal | GATECSE | Data Structure

Breadth First Search (BFS)

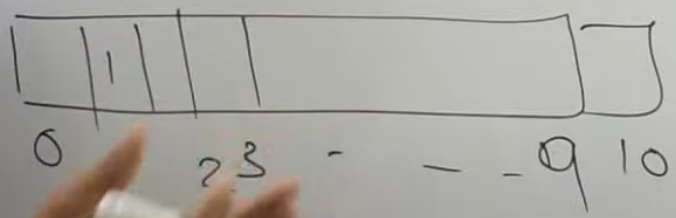




DAT

Record

(1 ... 10)



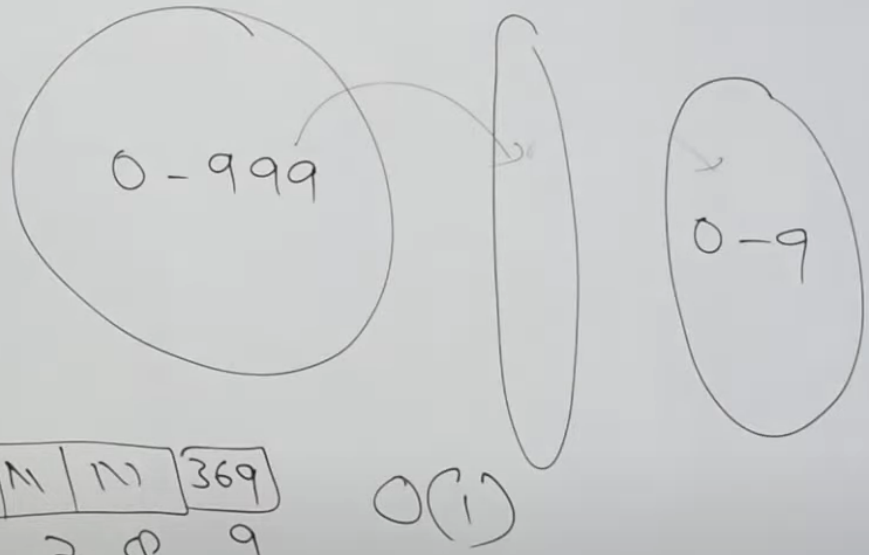
USA ⇒ O(n)
SA ⇒ O(log n)
LL ⇒ O(n)
B.T ⇒ O(n)
BST ⇒ O(n)
BBST ⇒ O(log n)

O(1)

Hashing

Taking a set of very large no. and then mapping into a set of small no

HASH Fn



CRT

1) chaining
2) open Add
   1) Linear
   2) Quadratic
   3) Double

146 % 10 = 6
133 % 10 = 3
990 % 10 = 0
369 % 10 = 9
233 % 10 = 3

0	1	2	3	4	5	6	7	8	9
990	N	N	133	N	N	146	N	N	369

Collision

O(1)

Types of Hash Function

Division Modulo

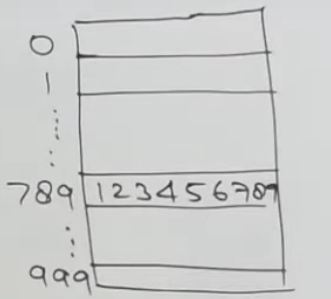
M = 1000

Key = 123456789

HF(key) = key mod M

= 123456789 mod 1000

= 789



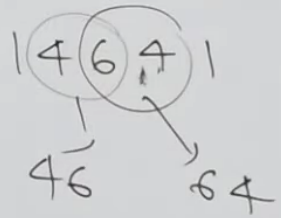
Hash Table

2) Mid Square

M = 100

Key = 121

$(121)^2 = 14641$



3) Digit Extraction

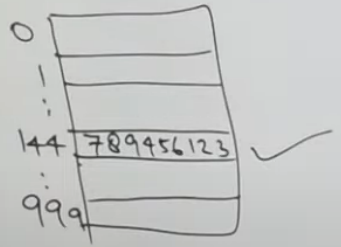
M = 1000

Key = 789 456 123

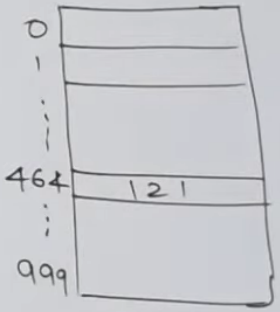
= 789 + 456 + 123 = 1368

= 1368

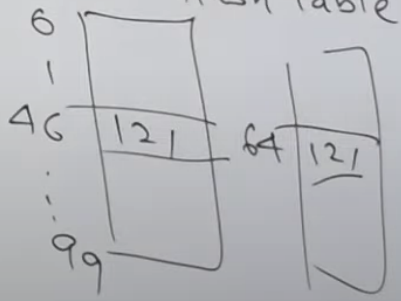
= 144



1) Square
2) Mid



Hash Table



Chaining in hashing | open addressing | collision resolution techniques | Data Structure

CHAINING (OPEN HASHING)

Keys: 25, 97, 80, 65, 38, 40, 59, 75, 17, 99, 35, 23

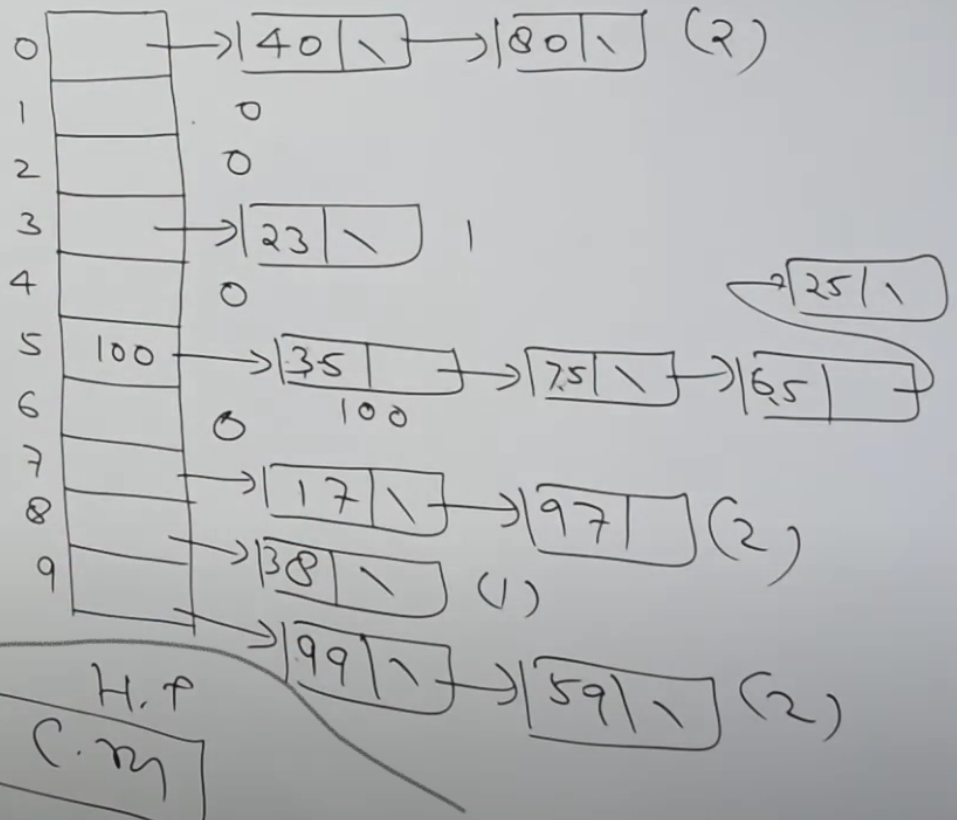
M = 10.

h(key) = key mod M.

10, 20, 30, 40, 50, 60

25 mod 10 = 5

65 mod 10 = 5



Avg UNIFORM In

Load factor $\alpha = \frac{n}{m}$

worst

Search => O(n)

Del => O(n)

$\frac{100}{10}$, 10

= 10

O(1) (4)

Search

$n = O(m)$

$= C + \frac{n}{m}$

$= C + C \cdot \frac{n}{n}$

$= O(1)$

H.f

$m = C \cdot n$

13:32 / 15:58

Linear Probing (Open Addressing)

Keys: 56, 79, 41, 44, 21, 43, 11, 29

✓ $H'(k) = k \bmod M$, $M = 10$

$21 \bmod 10 = 1$

$56 \bmod 10 = 6$

$79 \bmod 10 = 9$

$H(k,i) = (H'(k) + i) \bmod M$

for $i = 0, 1, 2 \ldots\ldots M-1$

$H(56,0) = (6+0) \bmod 10 = 6$

[$H(21,0) = (1+0) \bmod 10 = 1$

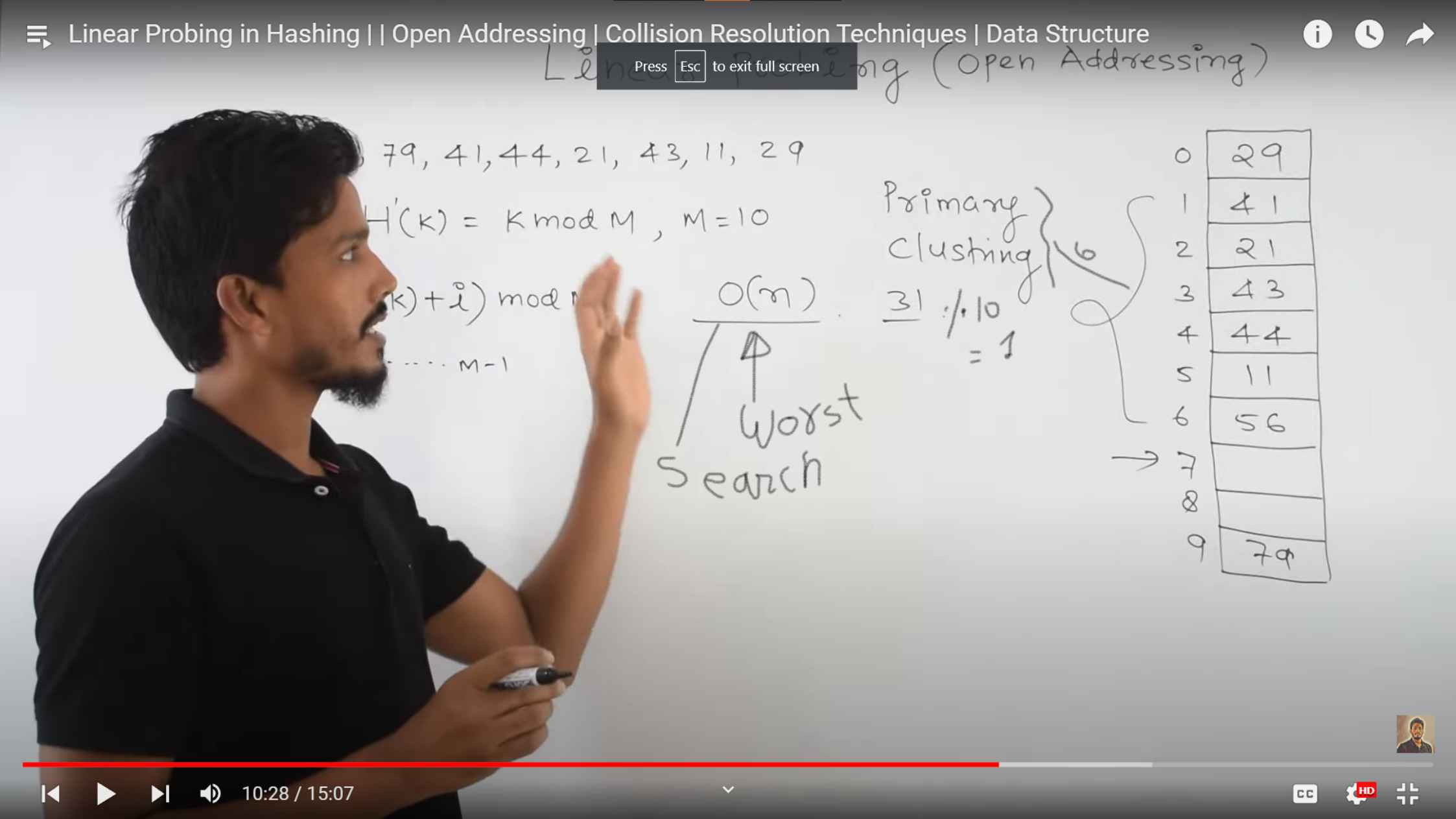
$H(21,1) = (1+1) \bmod 10 = 2$

$H(29,1) = (9+1) \bmod 10 = 0$

4 {
$H(11,0) = (1+0) \bmod 10 = 1$
$H(11,1) = (1+1) \bmod 10 = 2$
$H(11,2) = (1+2) \bmod 10 = 3$
$H(11,3) = (1+3) \bmod 10 = 4$
}

$H(11,4) = (1+9) \bmod 10 = 5$

0	
✓ 1	41
2	21
3	43
✓ 4	44
5	11
6	56
7	
8	
✓ 9	79



Linear Probing in Hashing | | Open Addressing | Collision Resolution Techniques | Data Structure

Linear Probing (Open Addressing)

Keys: 56, 7[illegible]4, 21, 43, 11, 29

H[illegible] mod M, M = 10

H(k, i) = (H[illegible] + i) mod M

for i = 0, 1, 2 ·

O(n)

Primary Clustering

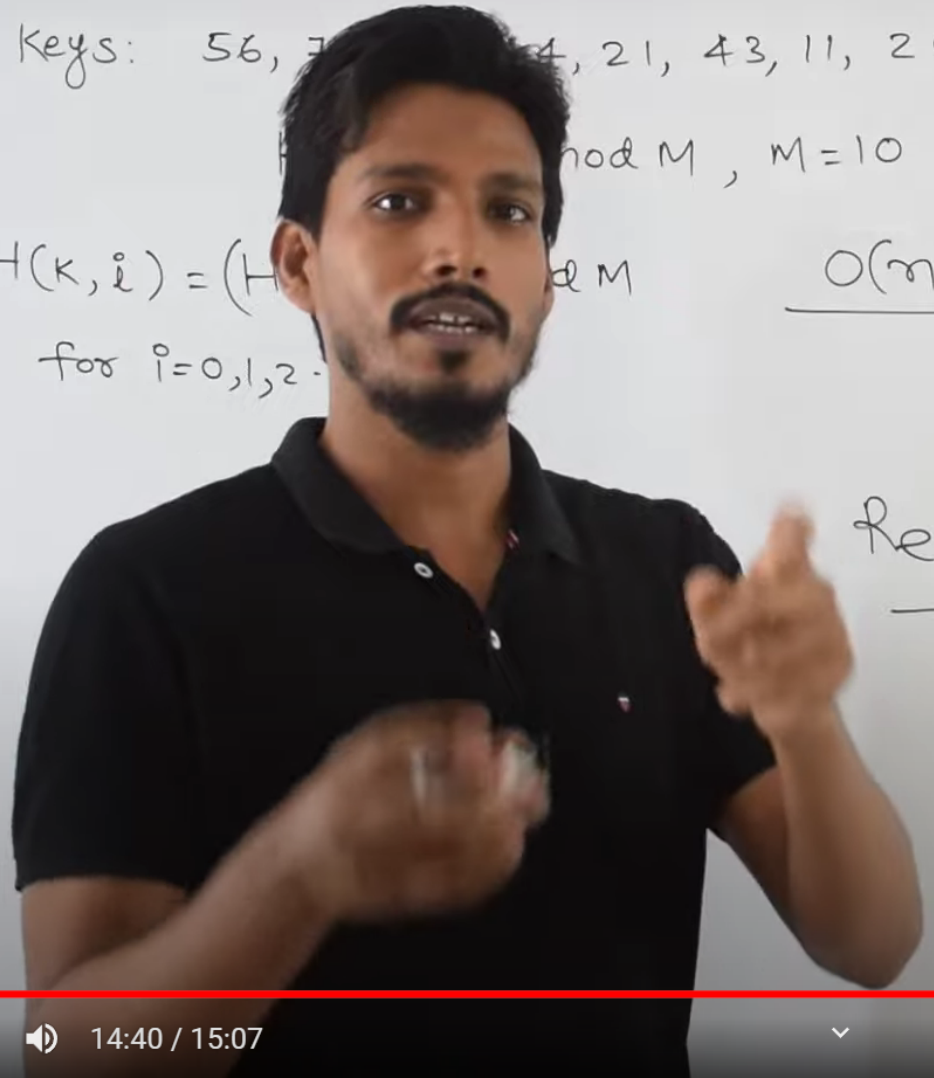
31

44

21

Rehashing

0	29
1	41
2	21
3	43
4	$
5	11
6	56
→ 7	
8	
9	79



14:40 / 15:07

Quadratic Probing in Hashing | | Open Addressing | Collision Resolution Techniques | Data Structure

Press Esc to exit full screen

64
17

Quadratic Probing

49, 58, 69, 29

M = 10

$h(k) = k \bmod M$

$C_1 = C_2 = 1$

$C_2 \neq 0$

$(h'(k) + C_1 i + C_2 i^2) \bmod M$

SC

LB	QP
PC	PC X
SC	SC

$k_1 = 29 \Rightarrow 1, 5, 1, 9, 9, 1, 5, 1, 9$

$k_2 = 19 \Rightarrow 1$

0	58
1	49
2	
3	
4	
5	69
6	
7	
8	18
9	89

$h(49,1) = (9+1+1) \bmod 10 = 1$

$h(58,1) = (8+1+1) \bmod 10 = 0$

$h(69,1) = (9+1+1) \bmod 10 = 1$

$h(69,2) = (9+2+4) \bmod 10 = 5$

$h(29,1) = (9+1+1) \bmod 10 = 1$

$h(29,2) = (9+2+4) \bmod 10 = 5$

$h(29,3) = (9+3+9) \bmod 10 = 1$

$h(29,4) = (9+4+16) \bmod 10 = 9$

$(M-1)$

$\bmod 10 = 9$

$+36) \bmod 10 = 1$

$-49) \bmod 10 = 5$

$4) \bmod 10 = 1$

$81) \bmod 10 = 9$

13:00 / 13:25

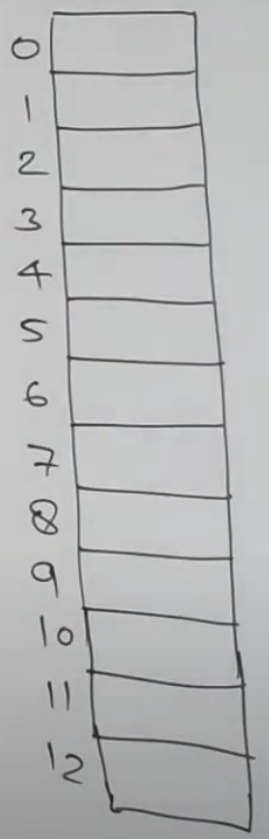
Double Hashing

79, 69, 98, 72, 14, 50 $\quad M = 13$

$h_1(k) = k \bmod M$ (M → 13), $\quad h_2(k) = 1 + k \bmod (M-2)$ (M-2 → 11)

$h(k, i) = (h_1(k) + i\, h_2(k)) \bmod M$

$i = 0, 1, 2 \ldots\ldots (M-1)$



Double Hashing

50 mod 13 = 11

79, 69, 98, 72, 14, 50

M = 13

$h_1(k) = k \bmod M$, $h_2(k) = 1 + k \bmod$

$h(k, i) = (h_1(k) + i\, h_2(k)) \bmod M$

$i = 0, 1, 2 \ldots\ldots (M-1)$

	LB	QP	DH
PC	✓	X	X
SC	✓	✓	X

$h(72,1) = (7 + (1 + 72 \bmod 11)) \bmod 13 \longrightarrow 72$

$= (7 + 7) \bmod 13$

$h(72,2) = (7 + 2(1+$

$= (7 + 2(7)$

$h(14,1) = 7 + 14 =$

$= (1 + 4) mo$

0	
1	79
2	
3	
4	69
5	14
6	
7	98
8	72
	50