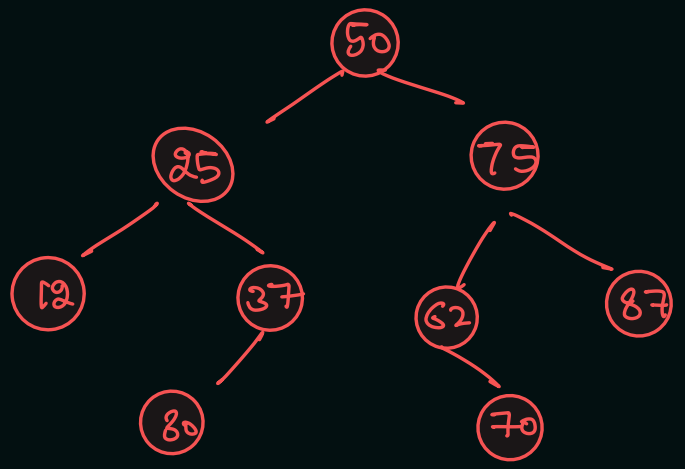
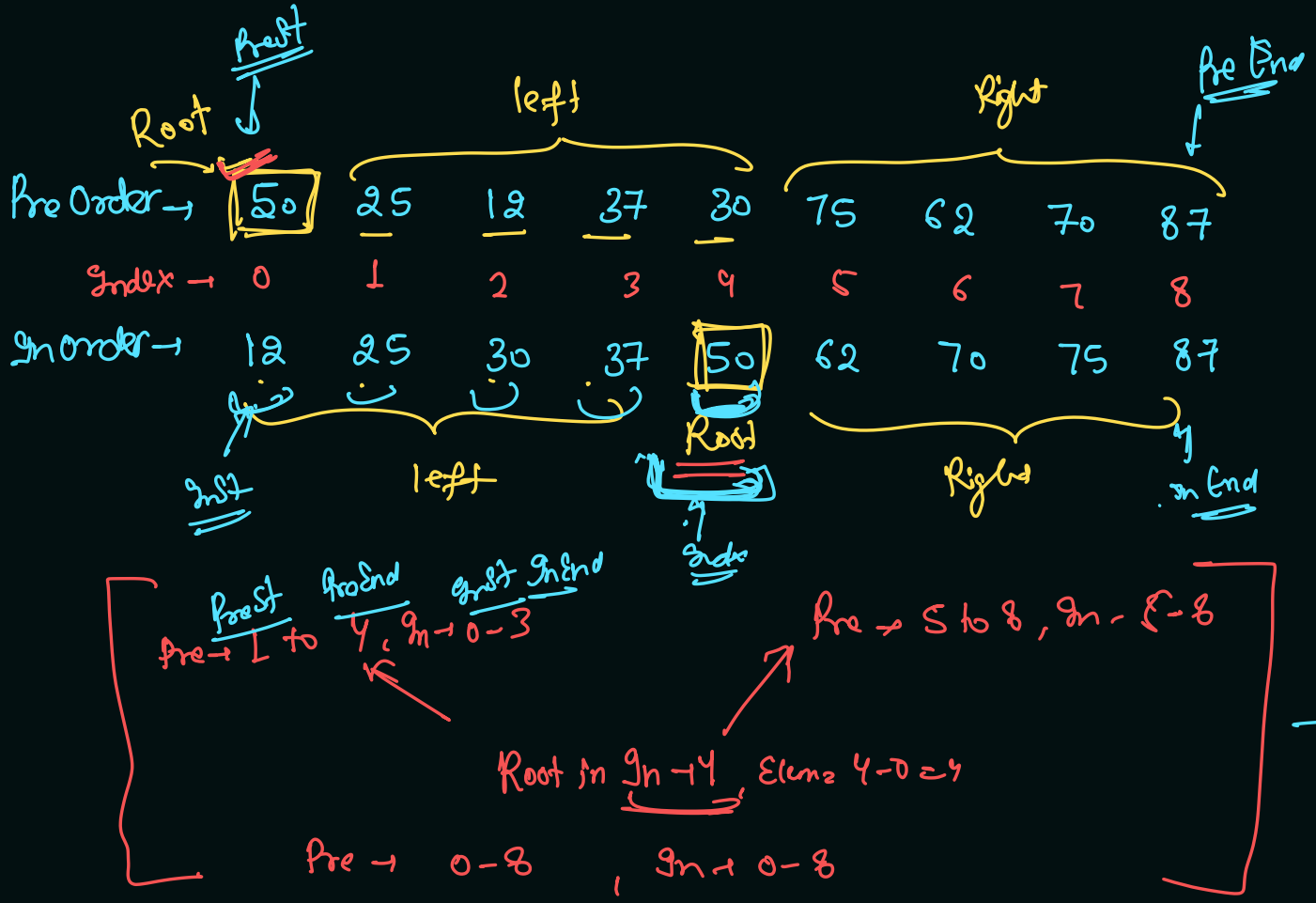
Construct tree using Pre Order and In Order's

Pre Order → Root Left Right

Pre Order →	50	25	12	37	30	75	62	70	87
Index →	0	1	2	3	4	5	6	7	8
In order →	12	25	30	37	50	62	70	75	87

In Order → Left Root Right



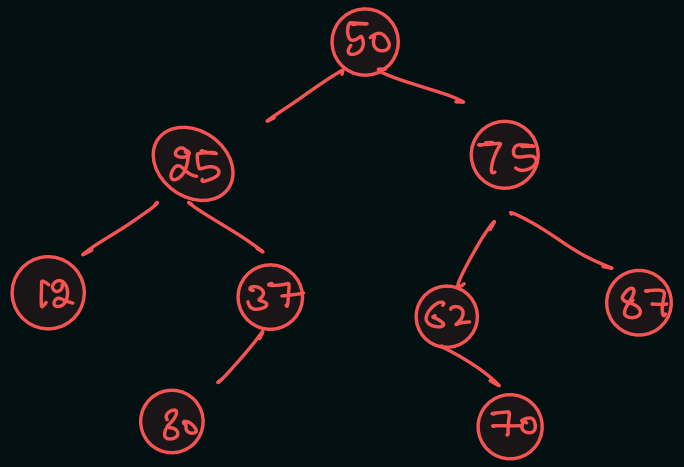


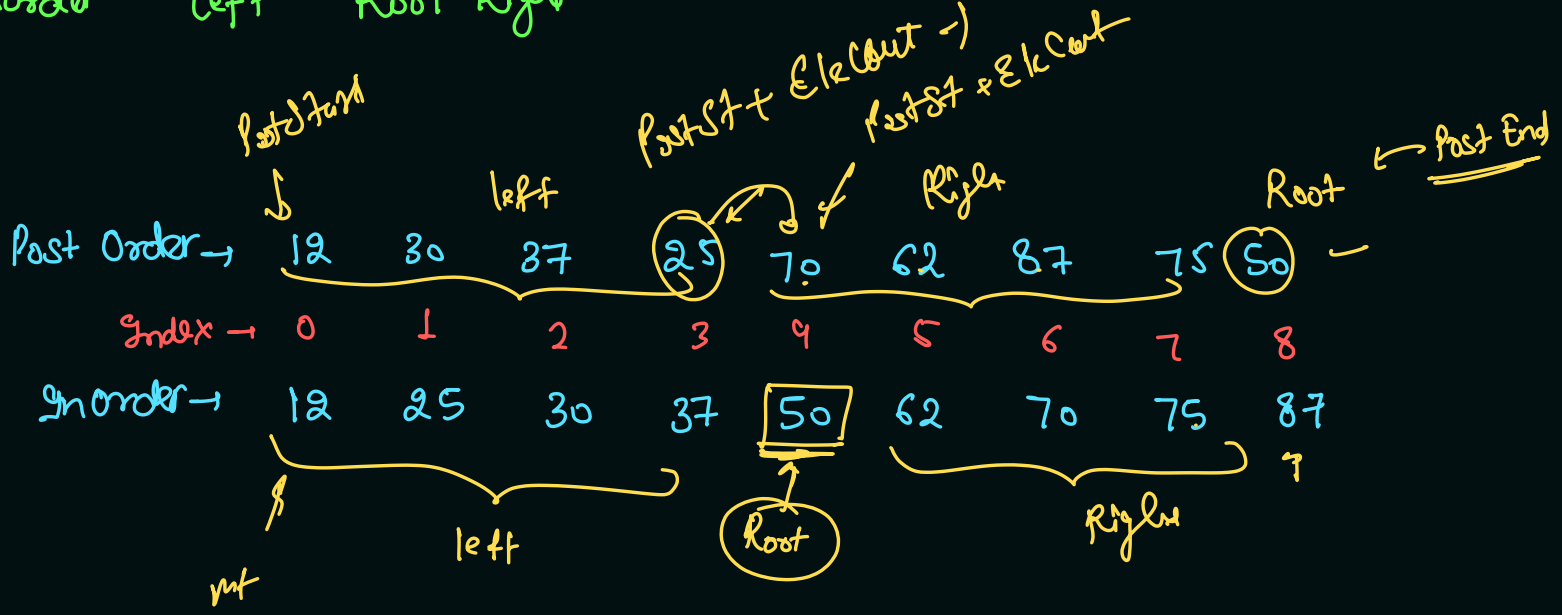
Construct Tree using Post Order and Inorder:

PostOrder → Left Right Root

Post Order →	12	30	37	25	70	62	87	75	50
Index →	0	1	2	3	4	5	6	7	8
Inorder →	12	25	30	37	50	62	70	75	87

Inorder Left Root Right



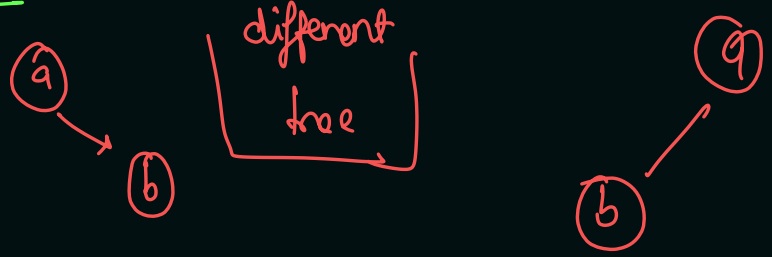


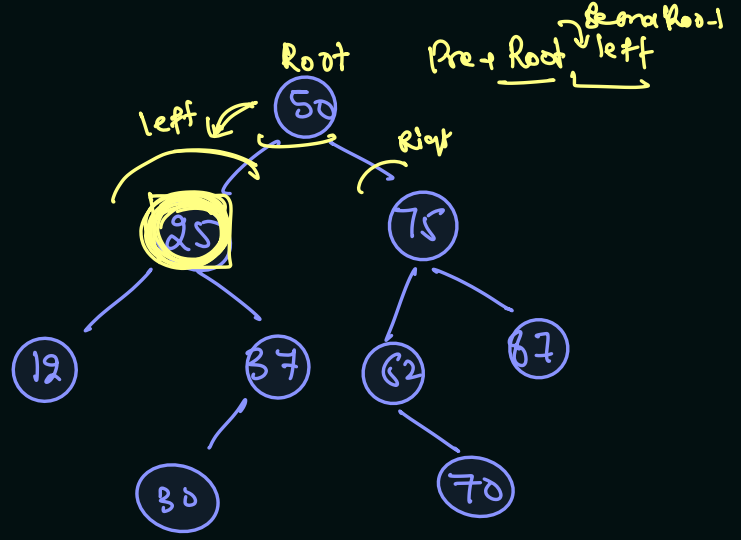
Recursion
fail &
Exception

Ek. Ex 4

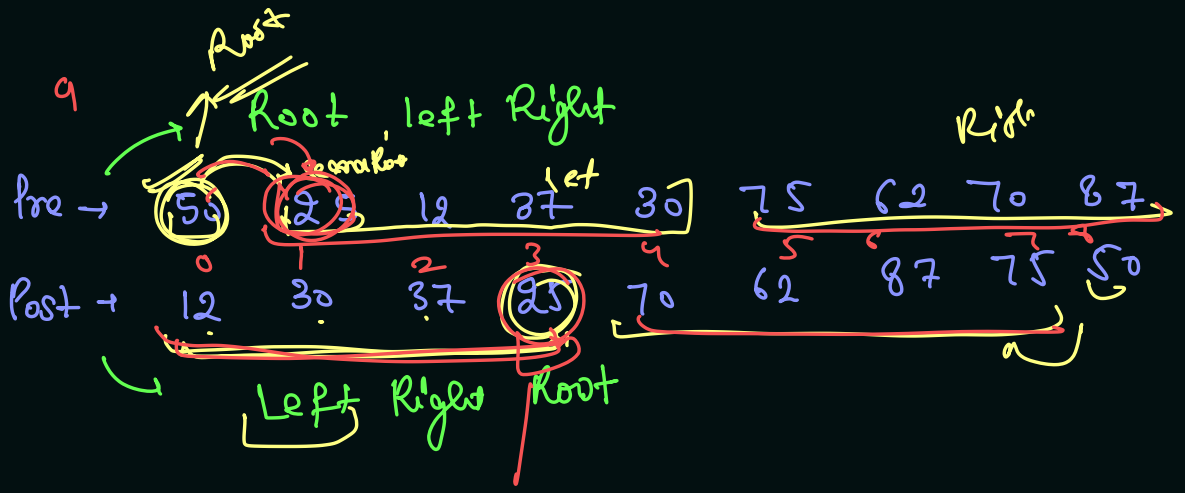
Construct tree using Pre Order and Post Orders

there can be more than one answer.





Pre → a b
Post → b a
⇐ Same →
Pre → a b
Post → b a



Complete binary Tree have always unique PreOrder & Post Order.

Element count = index - postL + 1

Construct tree using LevelOrder and InOrders

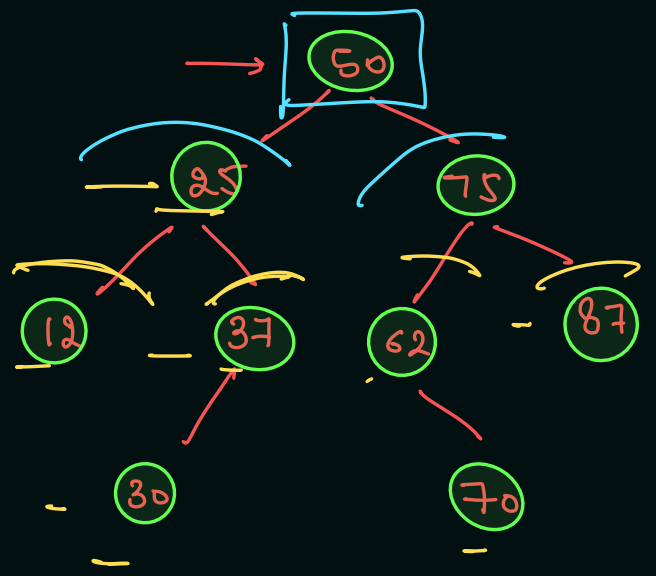
left Root Right

InOrder → 12 25 30 37 **50** 62 70 75 87

LevelOrder → 50 25 75 12 37 62 87 30 70

↳ level wise

Root of Current level



50

left

In - 12 **25** 30 37

level **25** 12 37 30

25

In → 12

level → 12

In 30 37

level → 37 30

Right

In → 62 70 **75** 87

level → **75** 62 87 70

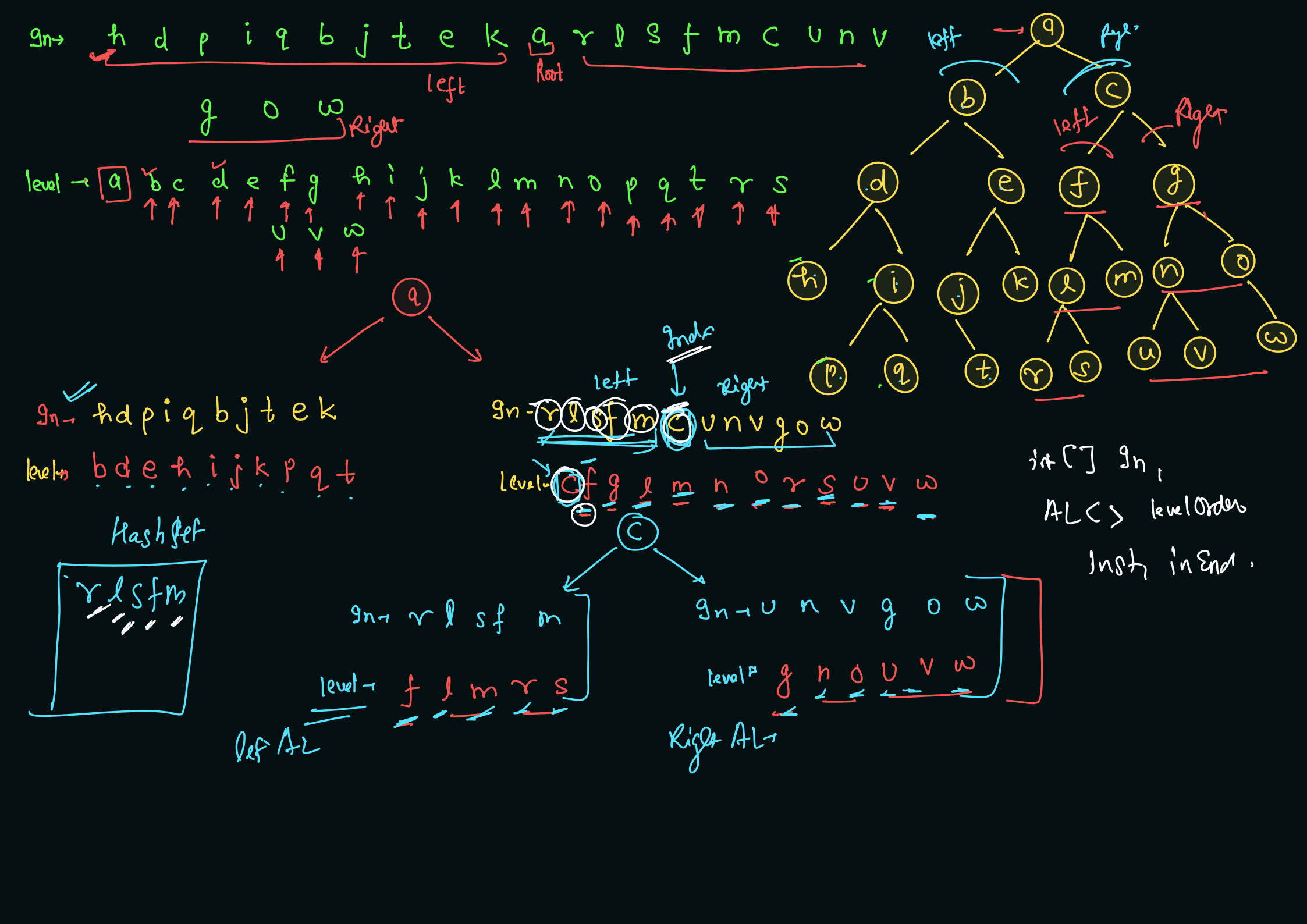
75

In → 62 70

lev - 62 70

In → 87

lev 87


In→ h d p i q b j t e k a r l s f m c u n v
left
Root
g o w
Right
level → a b c d e f g h i j k l m n o p q t r s
u v w
q
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
t
r
s
u
v
w
left
Right
In→ h d p i q b j t e k
level→ b d e h i j k p q t
Hash Set
r l s f m
In→ r l s f m c u n v g o w
level→ c f g l m n o r s u v w
Index
left
Right
c
In→ r l s f m
level→ f l m r s
Left AL
In→ u n v g o w
level→ g n o u v w
Right AL
int [] In,
AL<> levelOrder
Insty in End.

BST using Inorder:

NOTE: Inorder of BST is always sorted

middle element
Root

	0	1	2	3	4	5	6	7	8	9
In →	10	20	30	40	[50]	60	70	80	90	100

lo = 0
hi = 9

mid = $\frac{lo + hi}{2}$

Root = In[mid]

50
left: 0 to mid-1
right: mid+1 to length-1

BST (val)
left max
right min

left max < val < Right min] for every Node