```
for(i = 0; i < n ; i++)
{
    i = i + 1
}
```

O(n) ✓

math concept

$O(e^n) < O(n^5) < O(n^2)$

end

true

O() is for Average case only

False

Stack → FILO

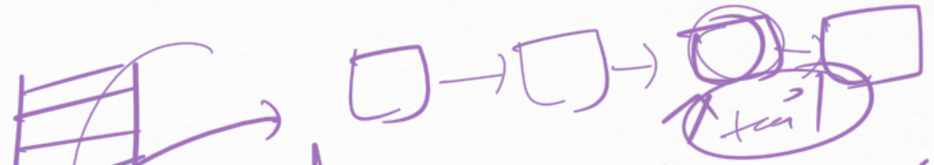
True

Circu Linked List

True Last element → first element

D.C.L.L

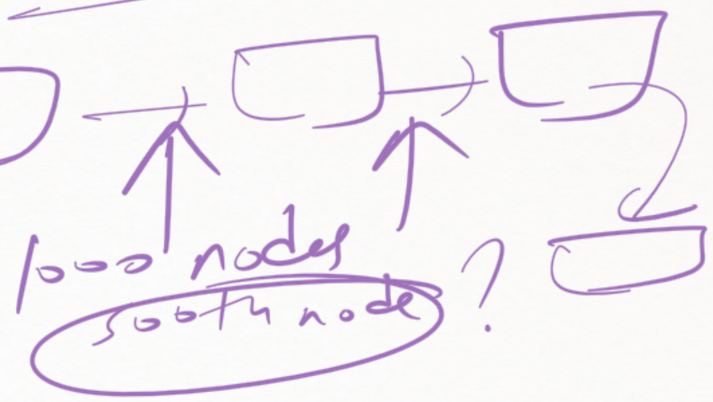
Is same as Above — Sufficient



Append O(n) in Linked list

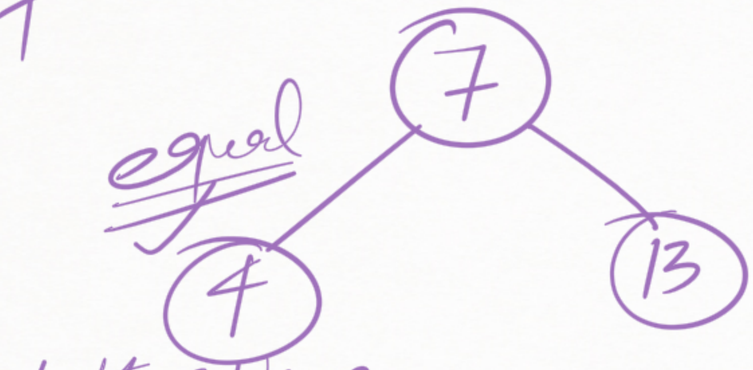
1. → with tail — False — O(1)
2. → with tho tail — True

Insert head tail



→ O(n)

#BST ADT

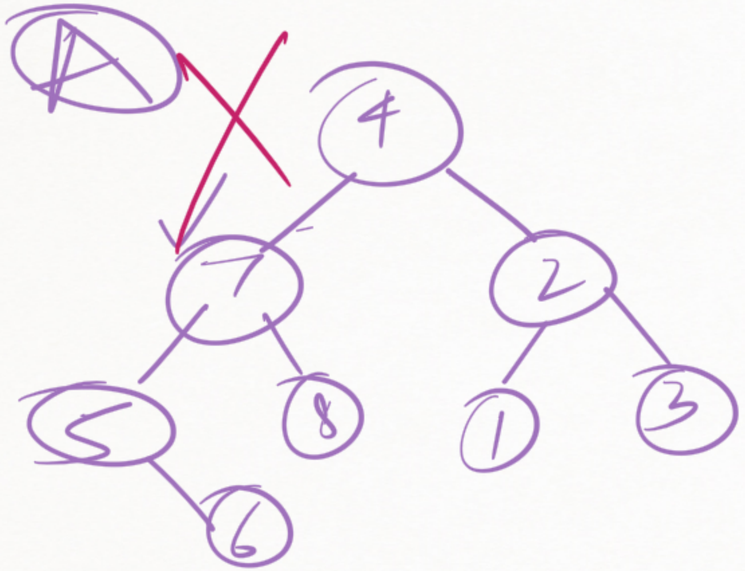


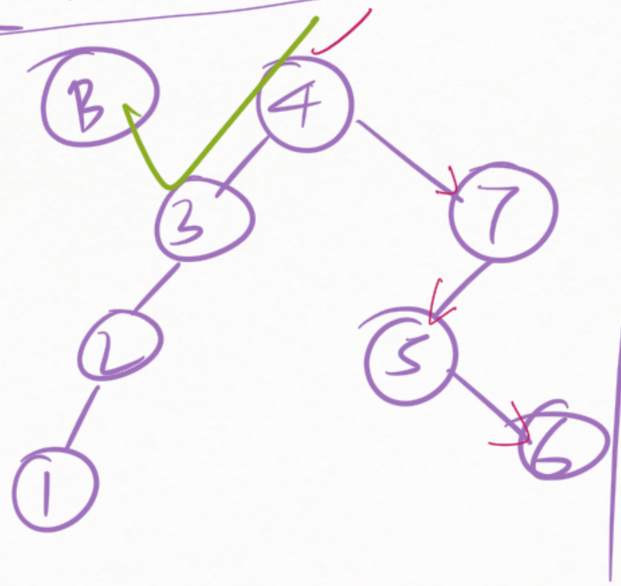
equal

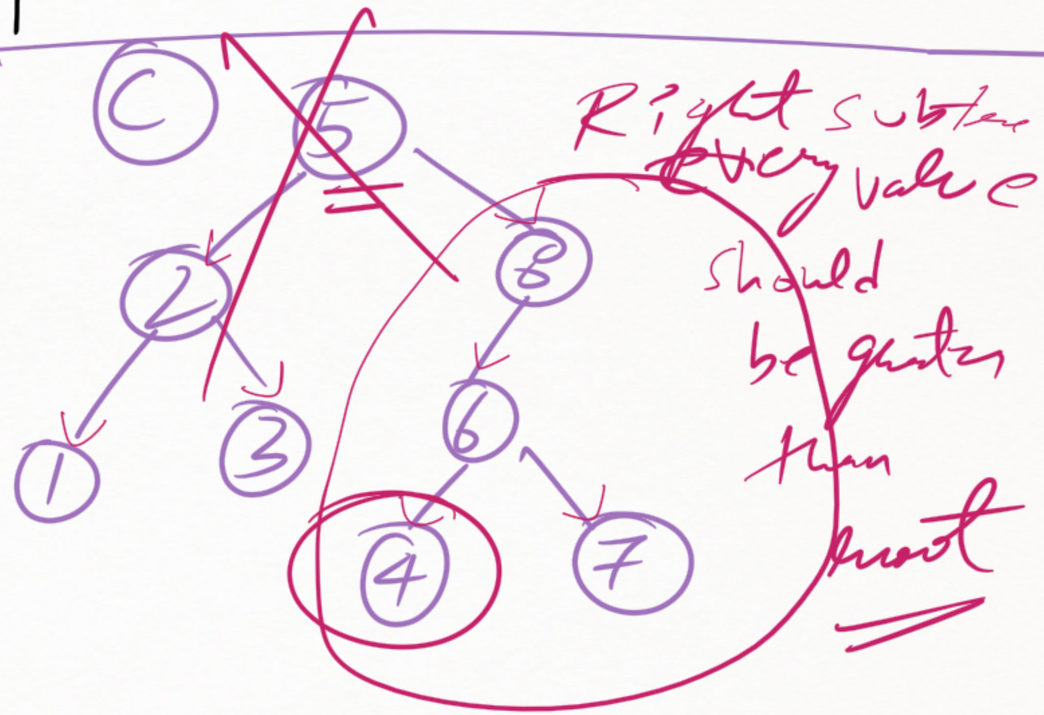
Left subtree smaller values

Right subtree Bigger values

→ Balance
op. search, insert, del.
O(log n) time
very less than linear DS.





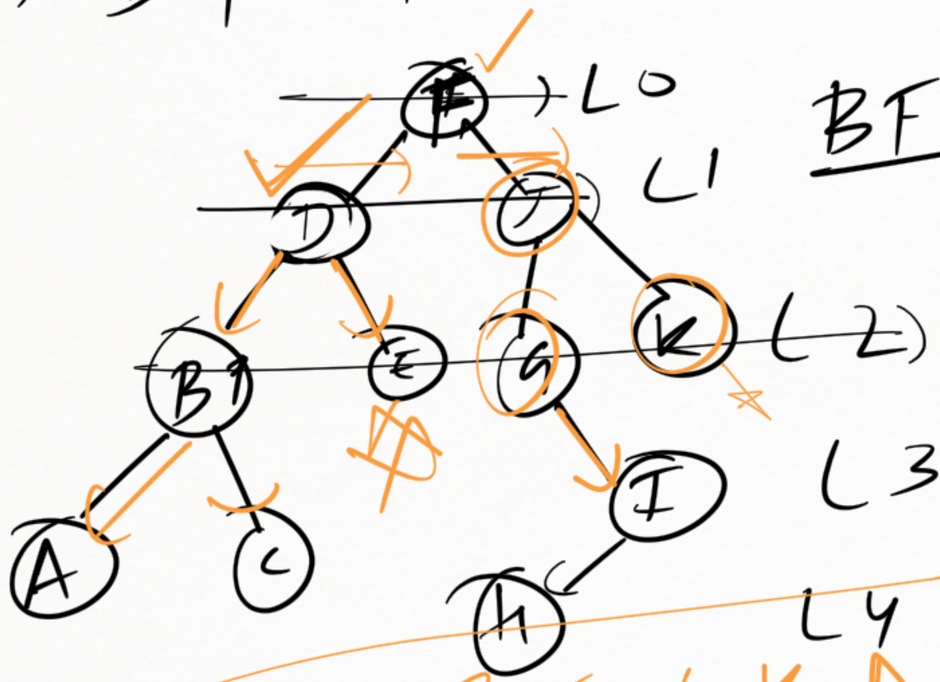


Right subtree every value should be greater than root

Traversals in BST → Going & printing each value of the tree.

→ Breadth First

→ Depth First



BFS : F, D, J, B, E, G, K, A, C, I, H

Queue → FIFO

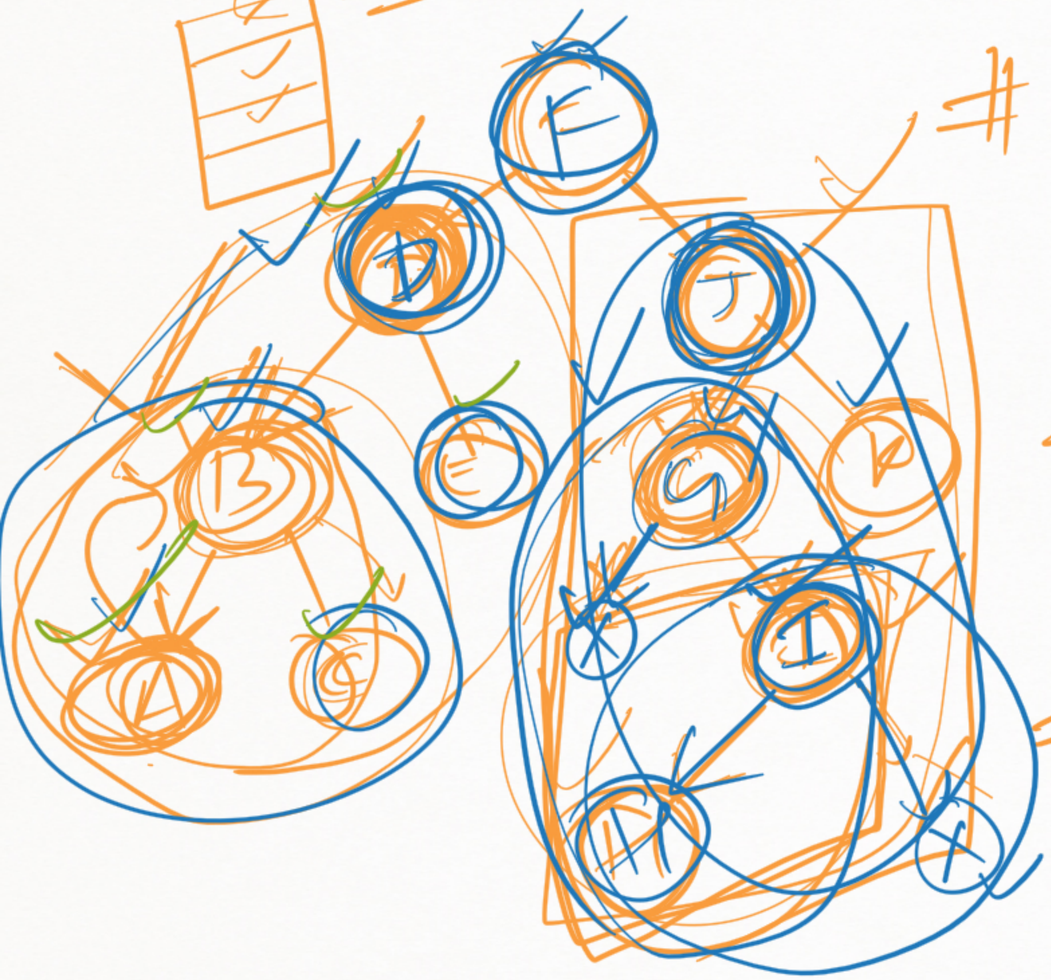
F, D, J, B, E, G, K, A, C, I, H

① print de, Popfront

② push it's children from back

Depth First
return first

- Inorder → left → root → right
- Preorder → root → left → right
- Postorder → left → right → root



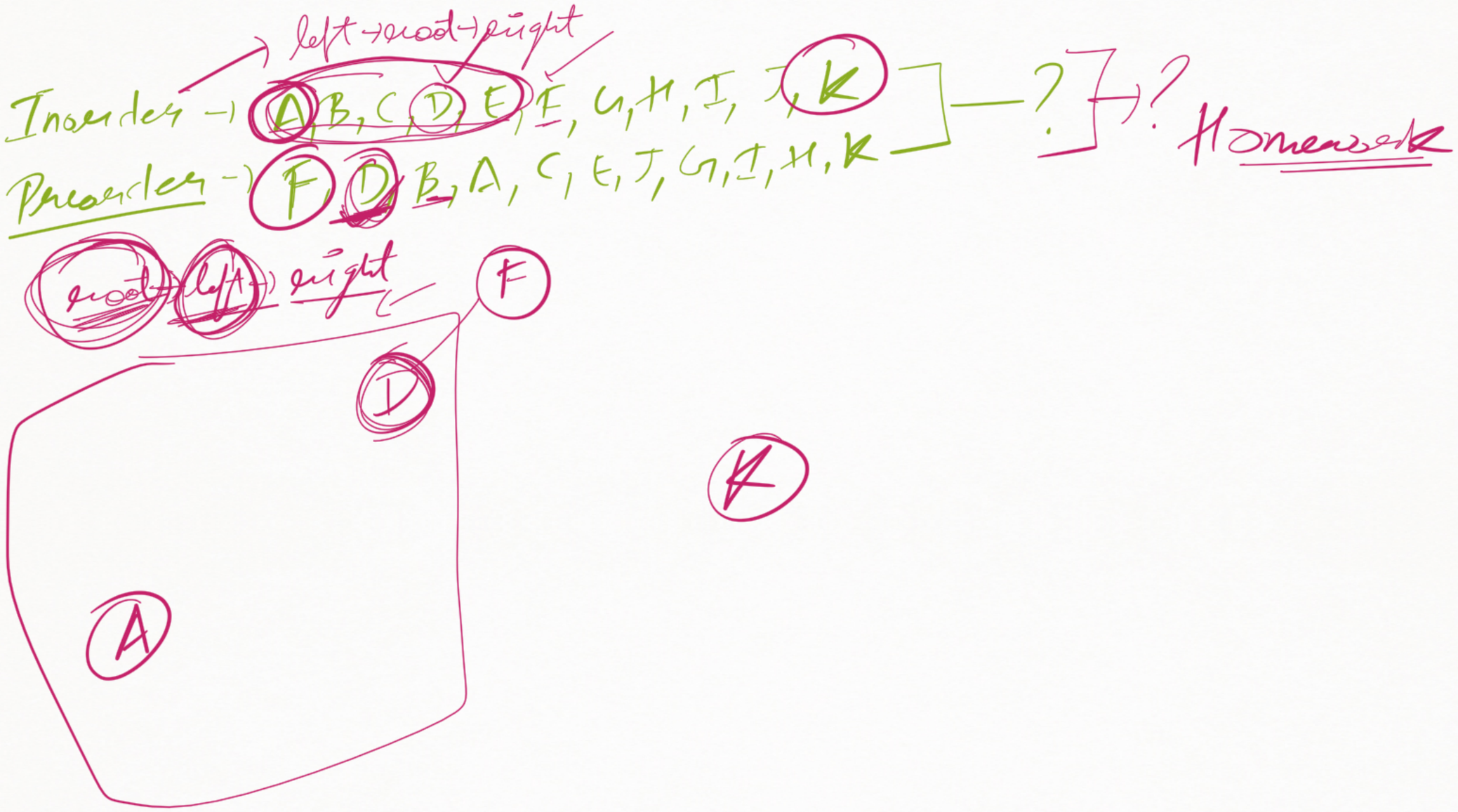
Recursively traverse

Sorted Order

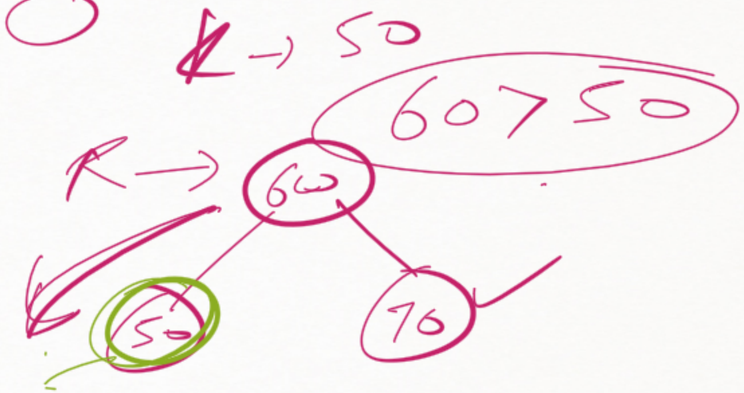
Inorder =) A, B, C, D, E, F, G, H, I, J, K

Preorder =) F, D, B, A, C, E, J, G, I, H, K

Postorder =) A, C, B, E, D, H, I, G, K, J, F


left→root→right
Inorder →
A, B, C, D, E, F, G, H, I, J, K
Preorder →
F, D, B, A, C, E, J, G, I, H, K
? → ?
Homework
root left right
F
D
K
A

Basic op.



① Find : Input: Key K, Root R

Output → Node in the tree with K

Find(K, R)

if R.key = K
  return R

else if R.key > K:
  return Find(K, R.left)

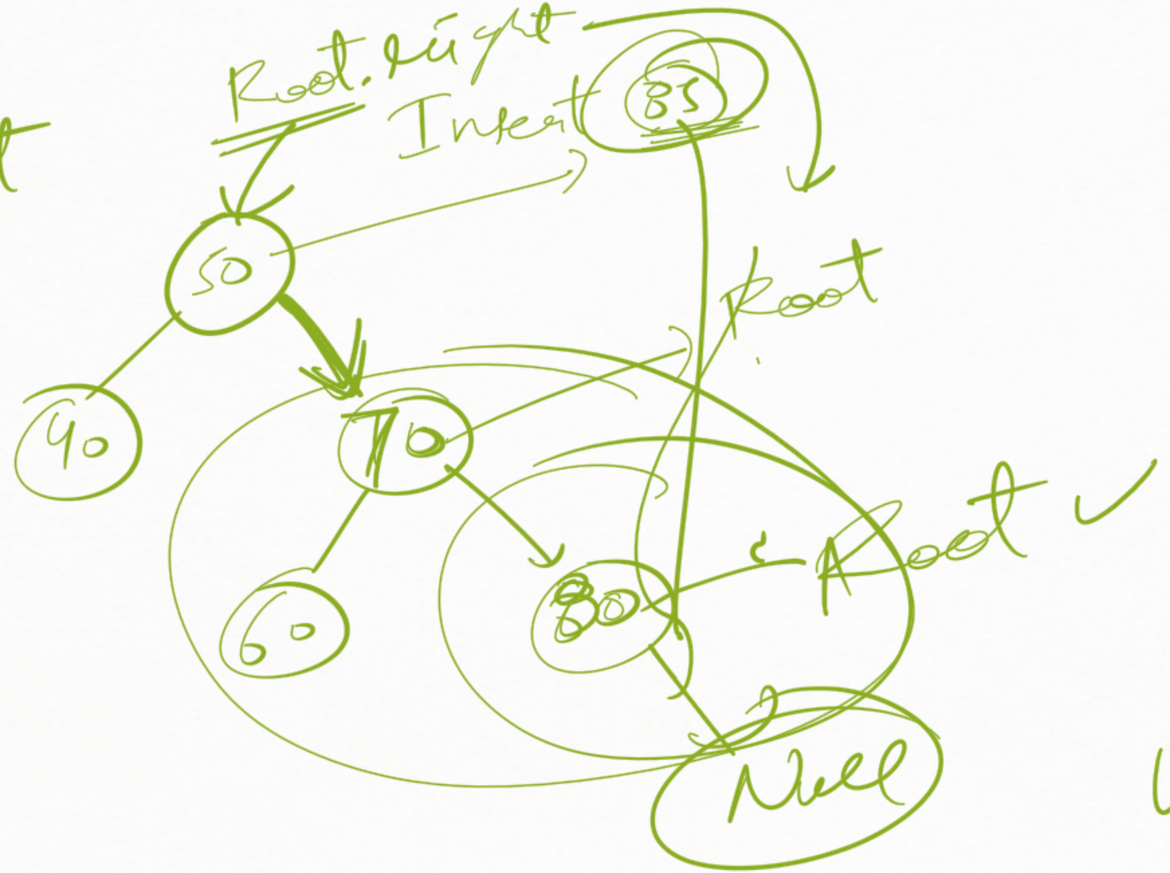
else if R.key < K:
  return Find(K, R.right)

✓ Ending Rec. condition.

What if the value is not in the tree?

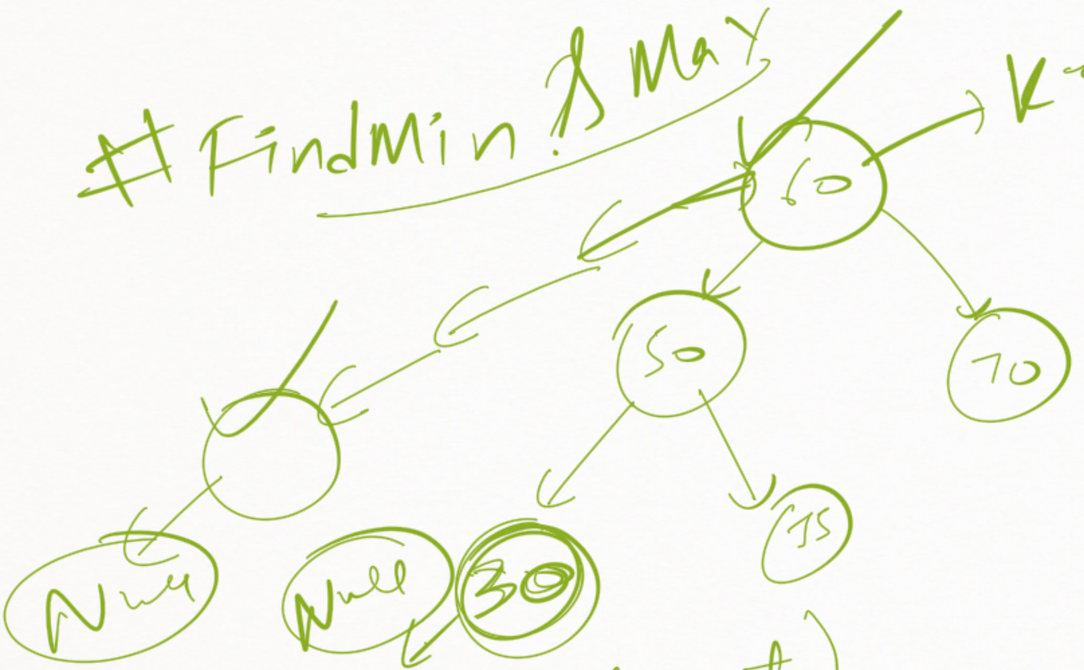
Homework: Error Report for Above Scenario

Insert



Written Code
already in JS
flow → Convert into PY

FindMin & Max



Key = 60

Property1: Node with minimum key will not have left child.

Property2: Node with maximum key will not have right child.

```
FindMin (root)
{ if (root.left == null)
    return root.key
  else
    FindMin (root.left)
}
```

Min is present on the left most side

Max is present on the right most side

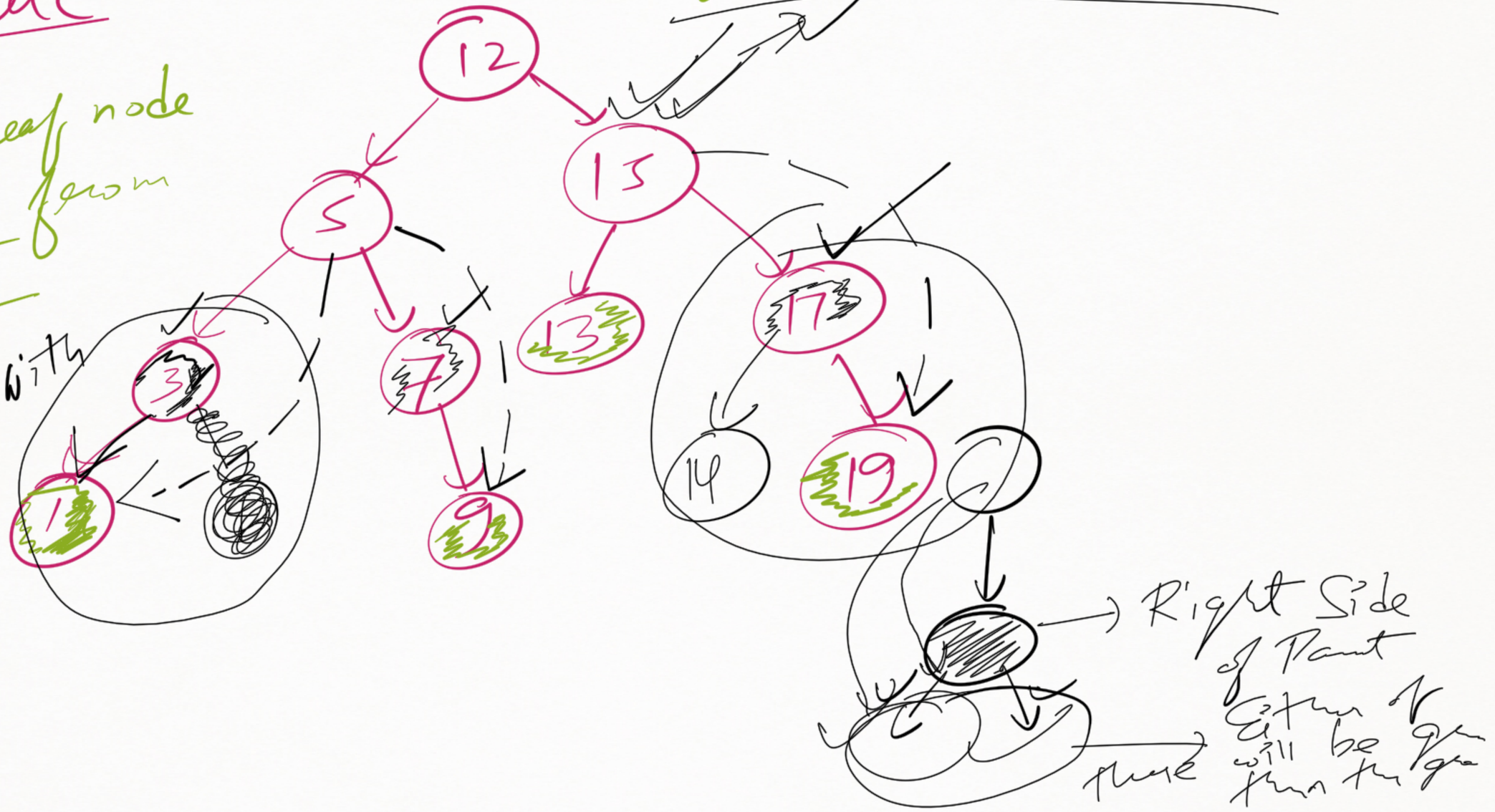
Find Max (HW)

Delete

BST Property should be valid
left <= root < right

Case 1: leaf node
Unlink from parent

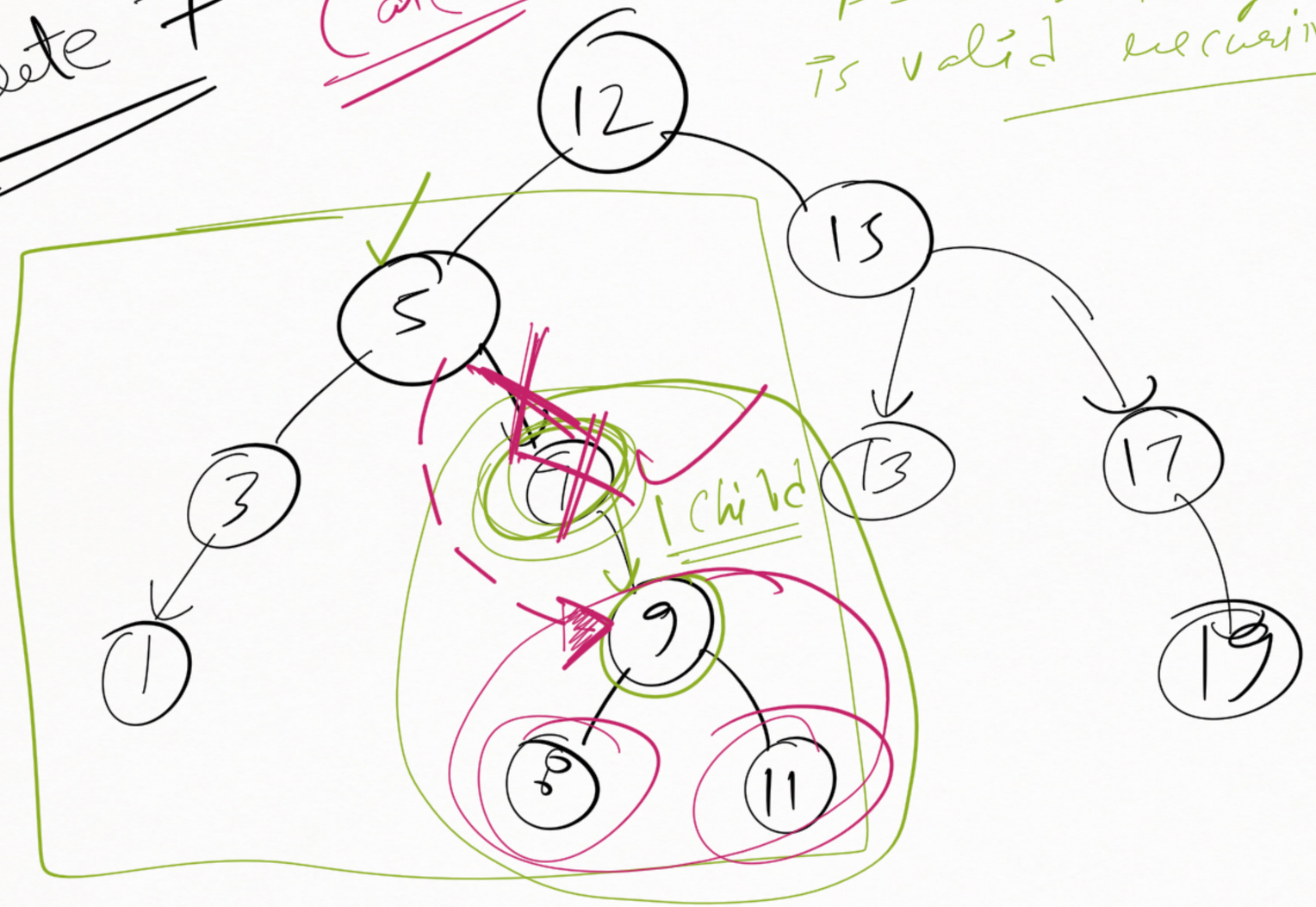
Case 2: With 1 child



Delete 7

Case 2

BST Property is valid recursively

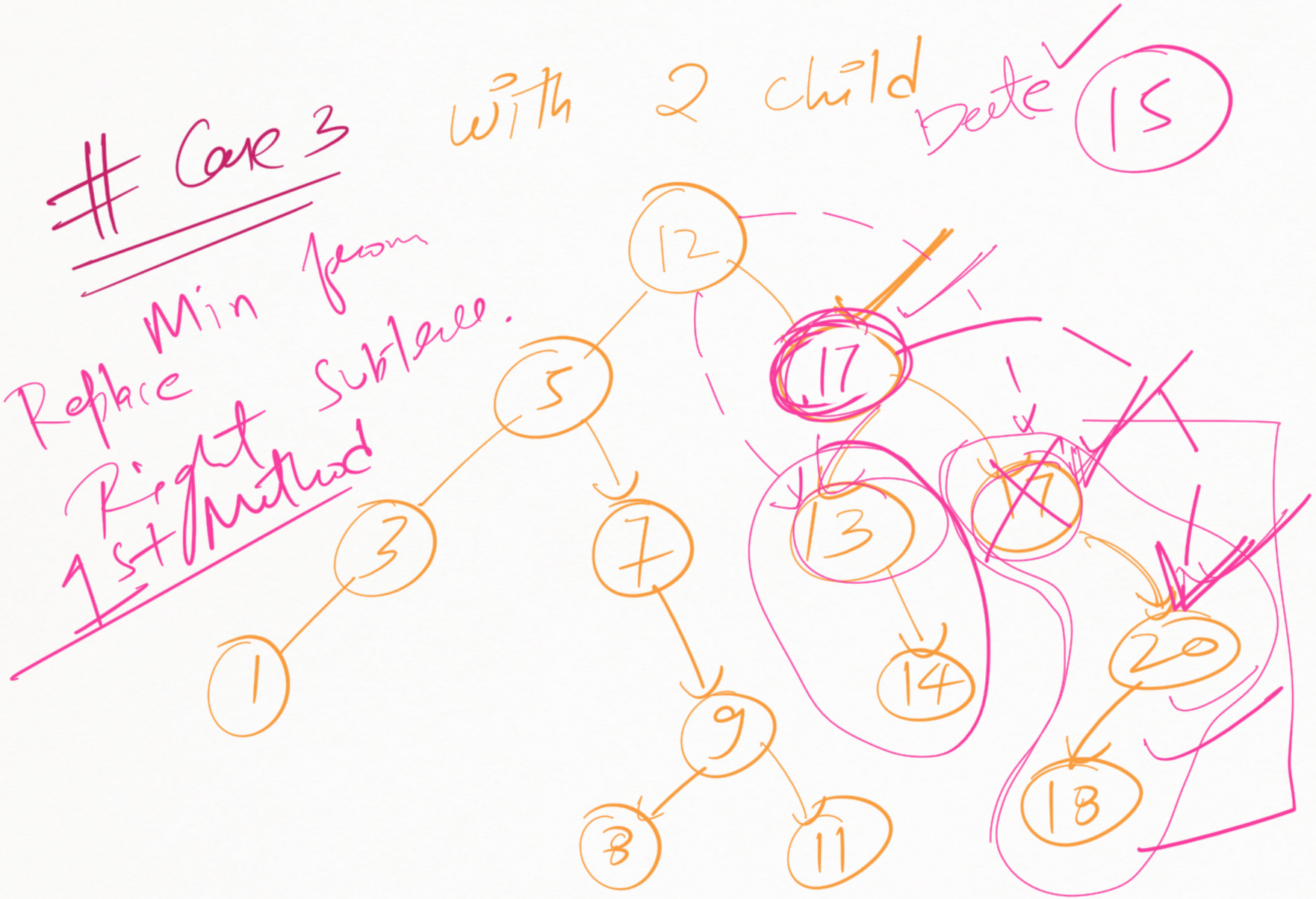


Case 3

With 2 child Delete ✓ 15

Min from Right subtree.

Replace 1st Method



Replace Max from left subtree

Delete is 2ND Method

Case 2

Valid BST Property

Delete Algo

1st method

1. Find min in right
2. Copy the value in the targetted node
3. Delete duplicate from right subtree

Recursive Call

2nd method

Delete 13

Delete 14

Case 2

Delete 13.5

Case 1