# Binary Search Tree

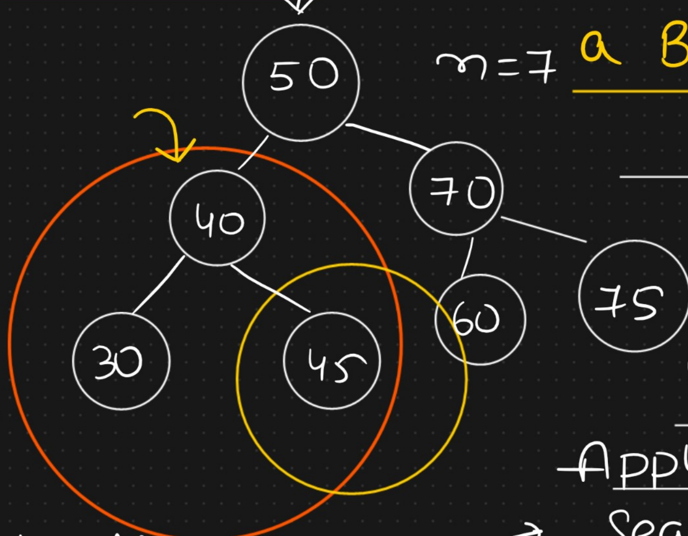
① Left child node val < Parent node val

② Right child node val > Parent node val

No duplicate elements in a BST

H

Binary Search tree




→ Balanced BST

Realtime Application BST

Searching operation

$\log_2 n$ = # Levels

$x = 45$

Time Complexity → $O(\log_2 n)$

Best/Average

Time Complexity → $O(H)$ Height

$O(\# \text{Levels})$

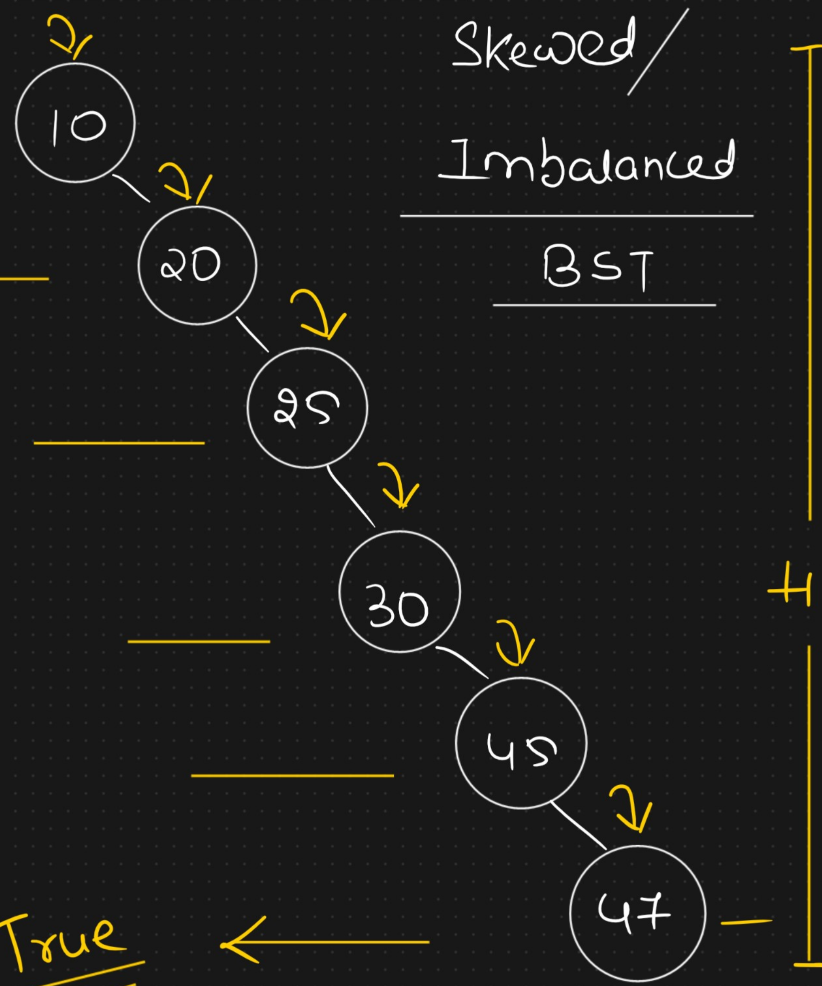
Balanced BST — $O(\log_2 n)$

## Worst case scenario

Skewed / Imbalanced BST

$n = 6$

$x = 47$




True

$$O(H) = O(\#\text{ levels})$$
$$= O(n)$$

# Tree Traversal

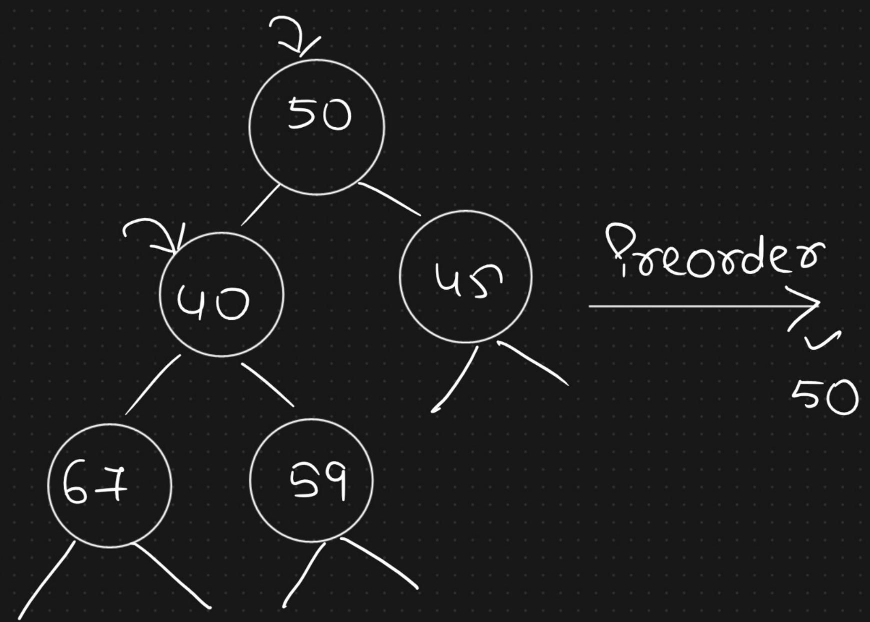
Recursion

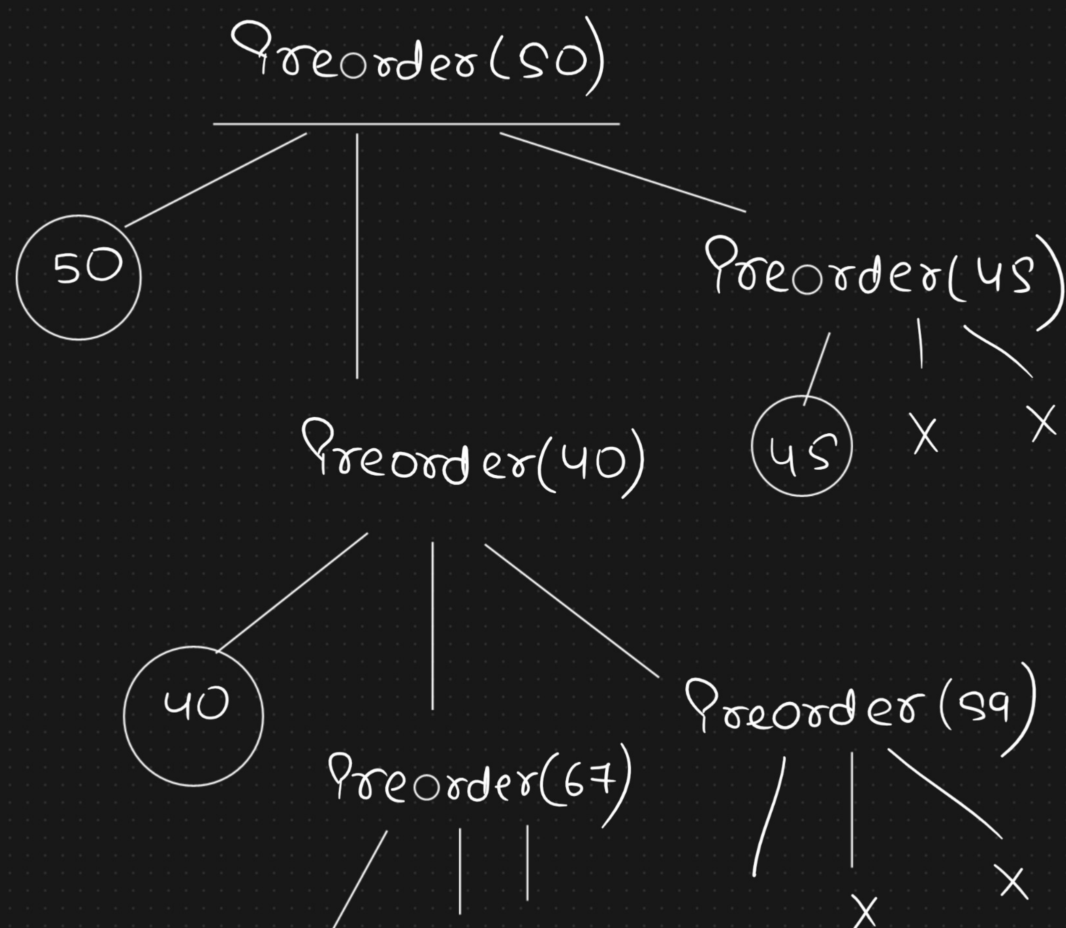
- ↳ Preorder
- ↳ Inorder
- ↳ Postorder

1 — node.val

2 — preorder (node.left)

3 — preorder (node.right)



67 × × 59

## Preorder traversal

$$\boxed{50, 40, 67, 59, 45}$$

Balanced Tree

$T(n)$

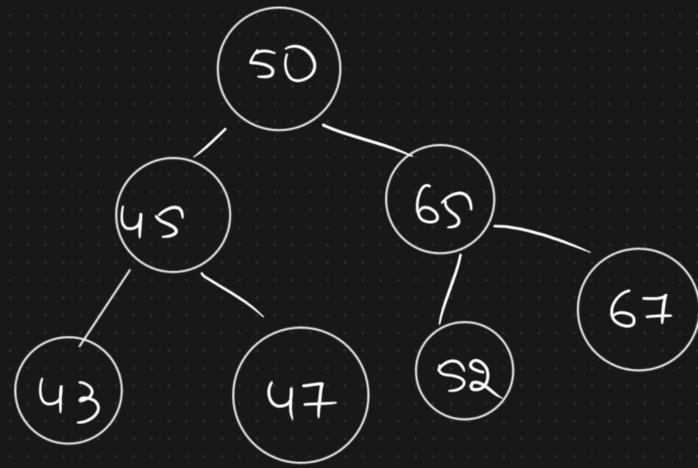
Inorder

1. Inorder(node.left) $T(n/2)$
2. print(node.val) $c$
3. Inorder(node.right) $T(n/2)$

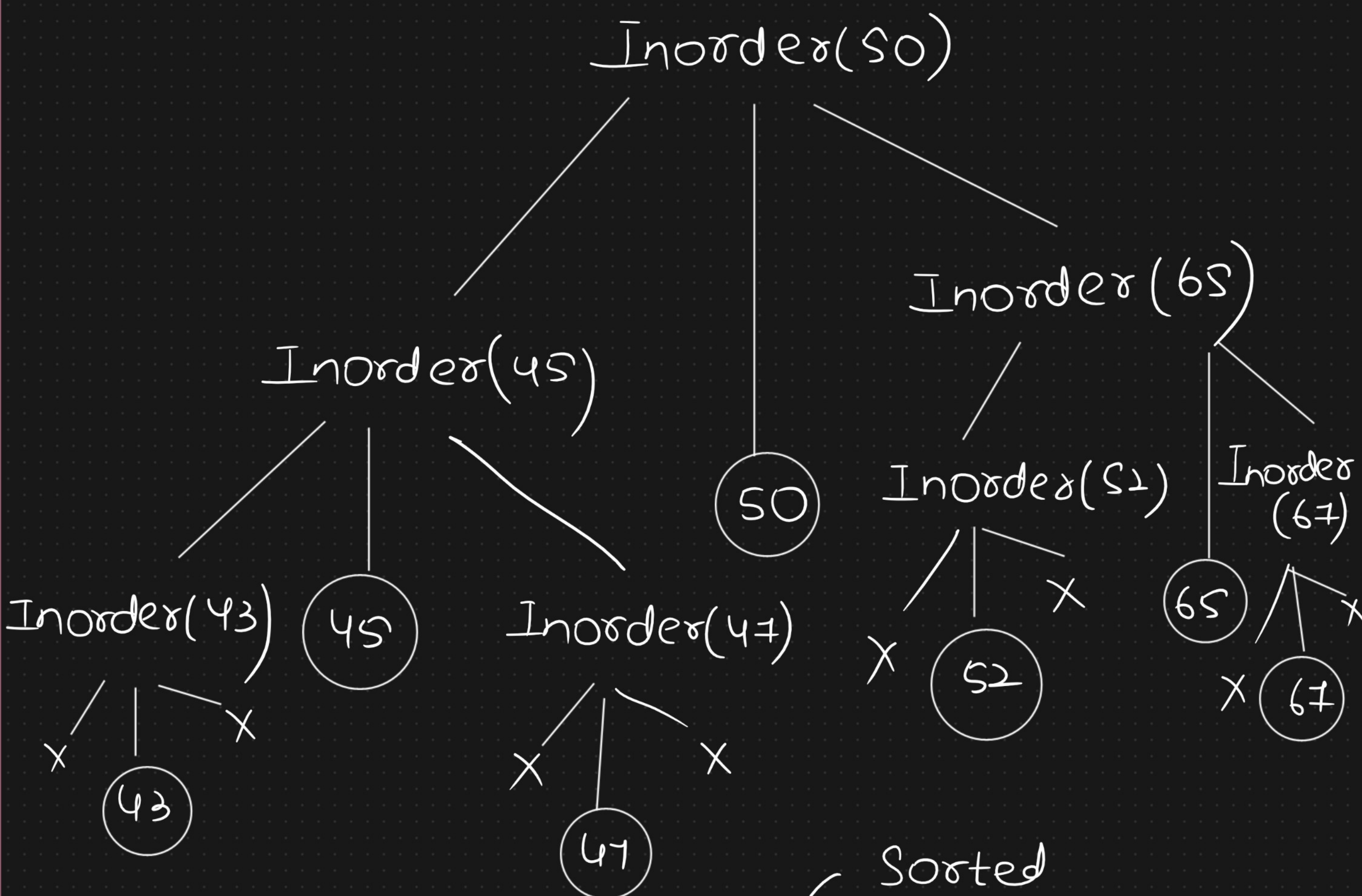
Postorder

1. Postorder(node.left)
2. Postorder(node.right)
3. Print(node.val)

BST




## Inorder traversal




Sorted manner

43, 45, 47, 50, 52, 65, 67

$O(n)$

Property

→ BST

→ Inorder traversal

→ Sorted array (Ascending order)

Best/Average scenario

Reccrrence Relation

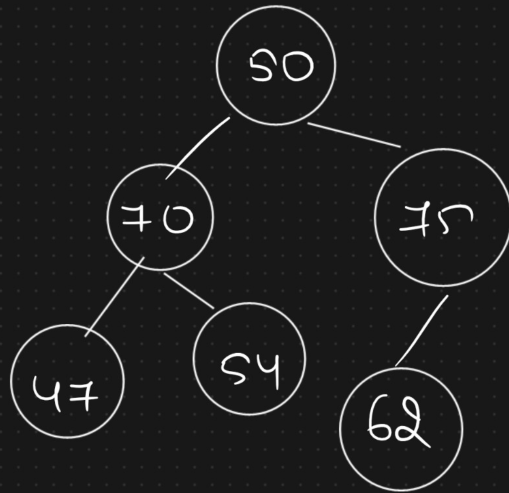
Substitution

Master's Theorem

$T(n) = 2T\left(\frac{n}{2}\right) + c \rightarrow O(n)$

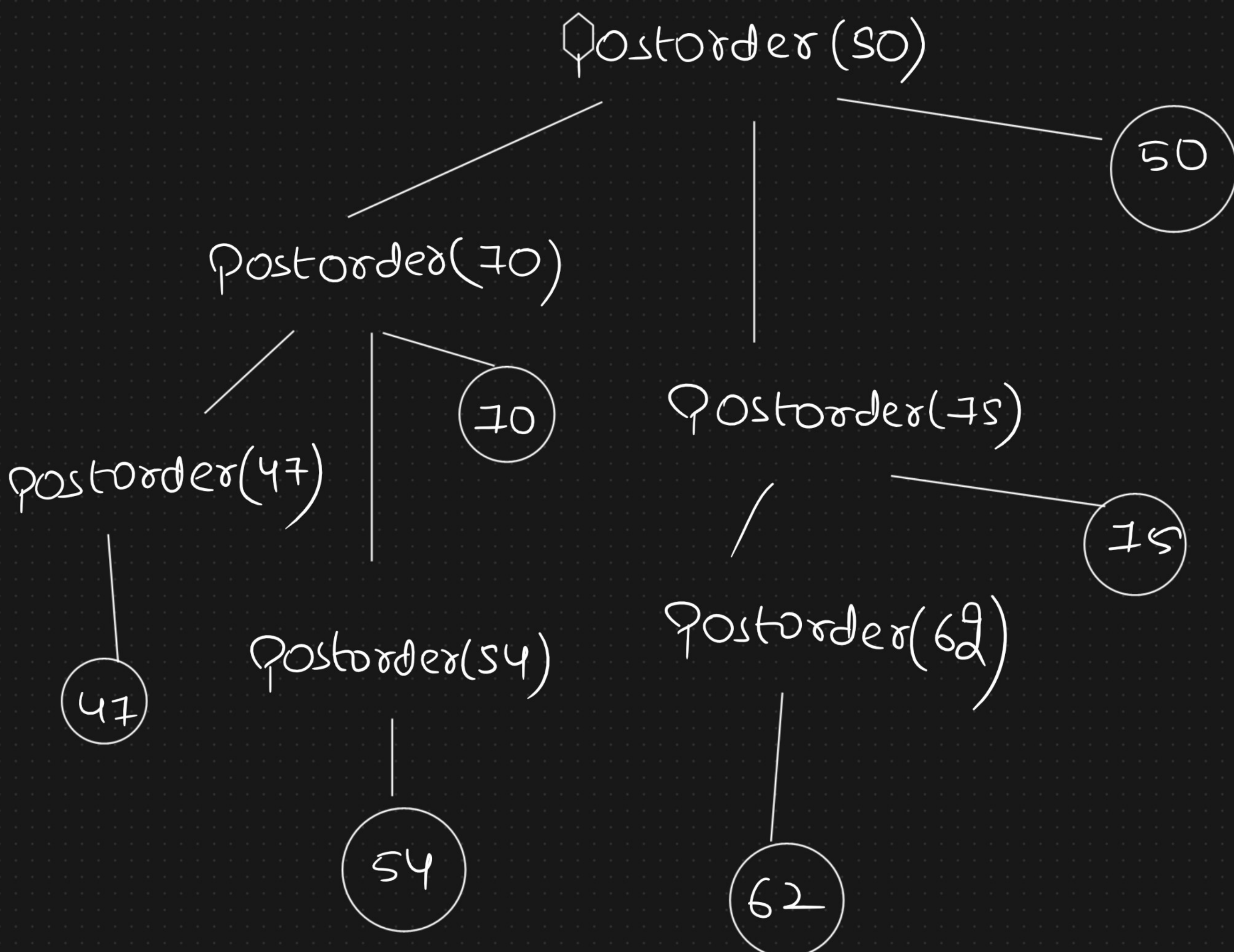
$T(n/2)$ Left subtree Recursion

$T(n/2)$ Right subtree Recursion




↳ Postorder(node.left)
Postorder(node.right)
print(node.val)

## Postorder Traversal




$\boxed{47, 54, 70, 62, 75, 50}$

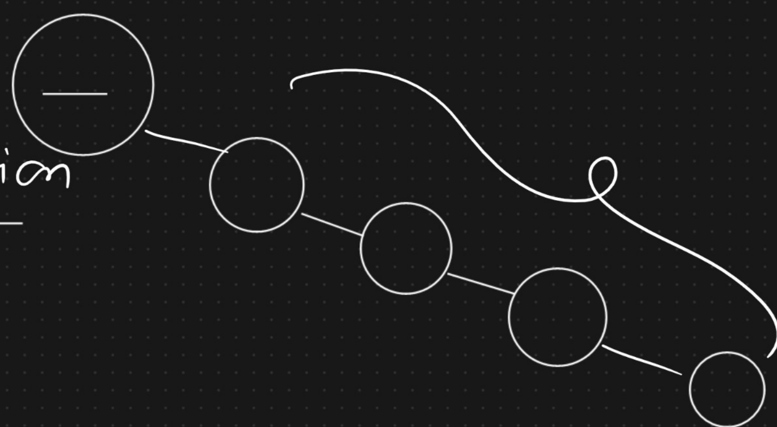
Root Node

Preorder (1st Pos)

Postorder (Last Pos)

Inorder ($n/2$ Pos)

## Worst case scenario



Recurrence Relation

$$T(n) = T(n-1) + c$$

$\hookrightarrow O(n)$

| Best/Avg case | Worst case |
| --- | --- |
| $T(n) = 2T(n/2) + c$ | $T(n) = T(n-1) + c$ |
| $T(n) = O(n)$ | $T(n) = O(n)$ |

time complexity = O(1)
(constant)

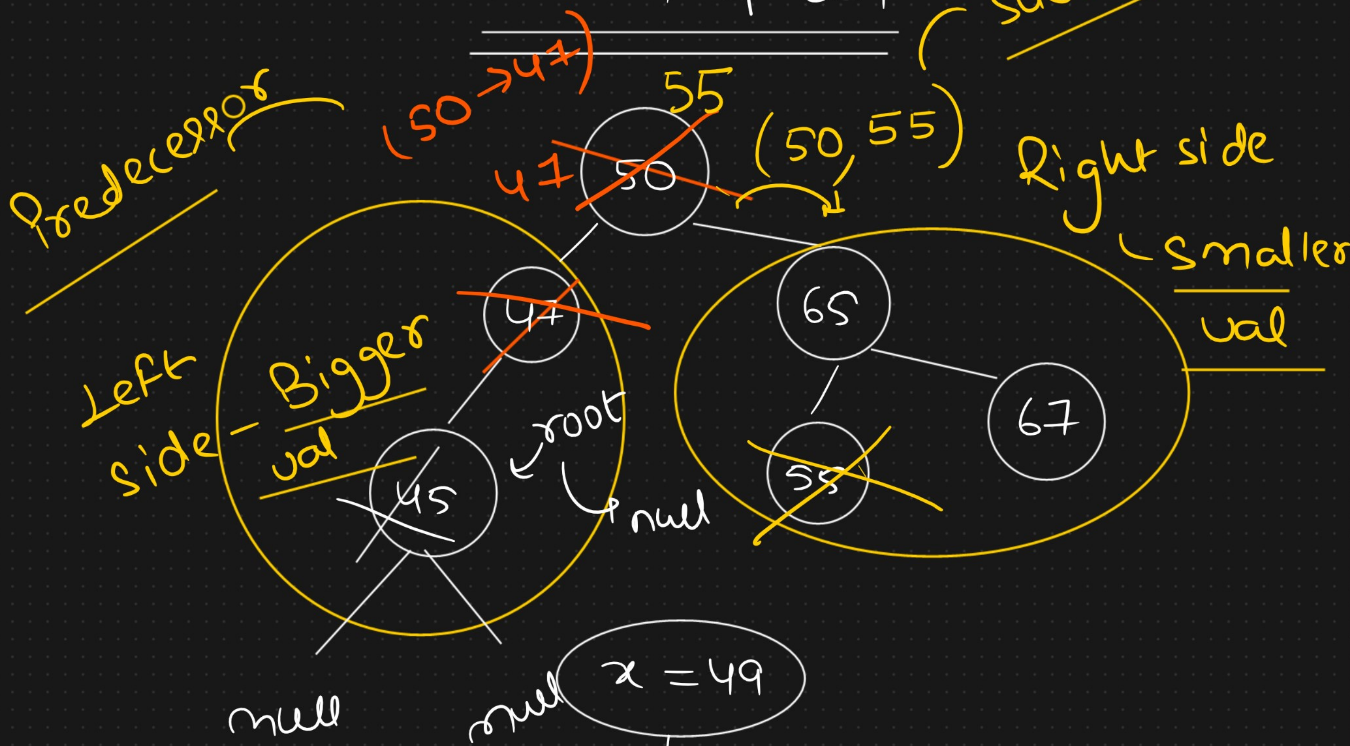
Kth Smallest (BST)

Psuedo code

1. Inorder Traversal
   Sorted array

2. return arr(k-1);
   final answer

# Deletion of BST




$x = 49$ → ① Search $x$

② Case 1

↳ $x$ - Leaf Node

↙

Assign = null

$x = 50$ → 2 child node