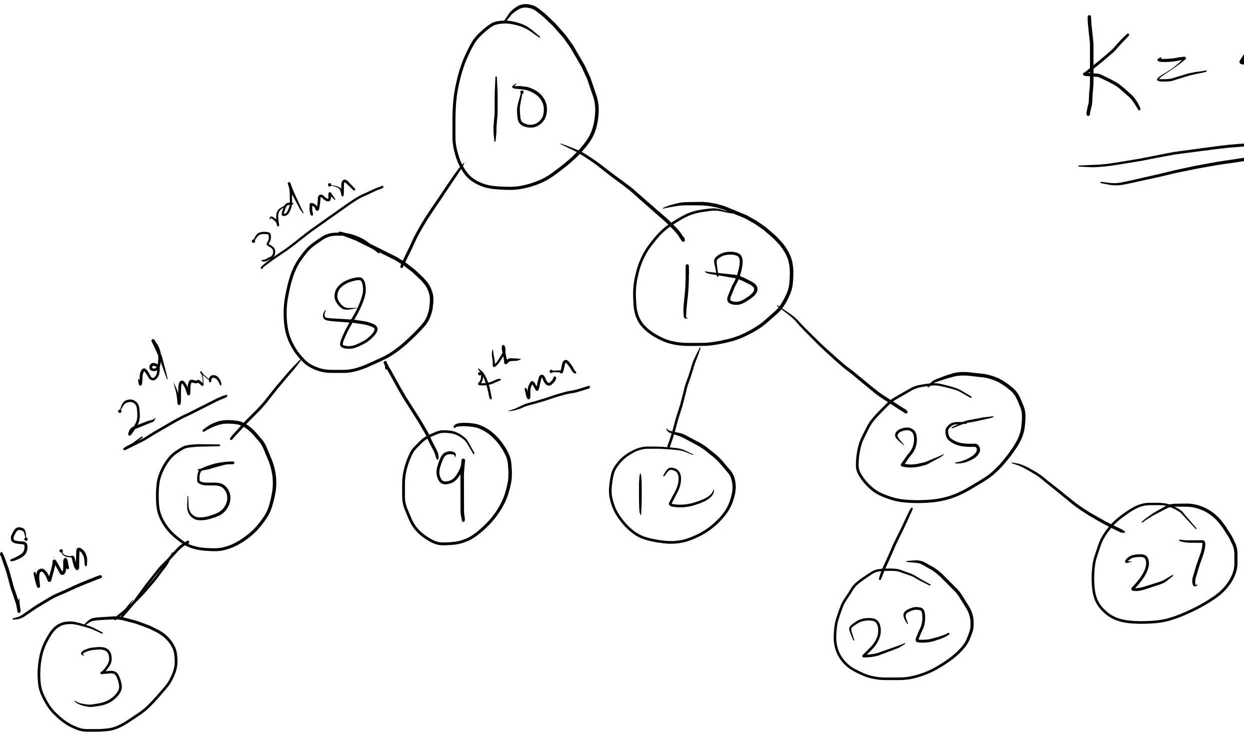
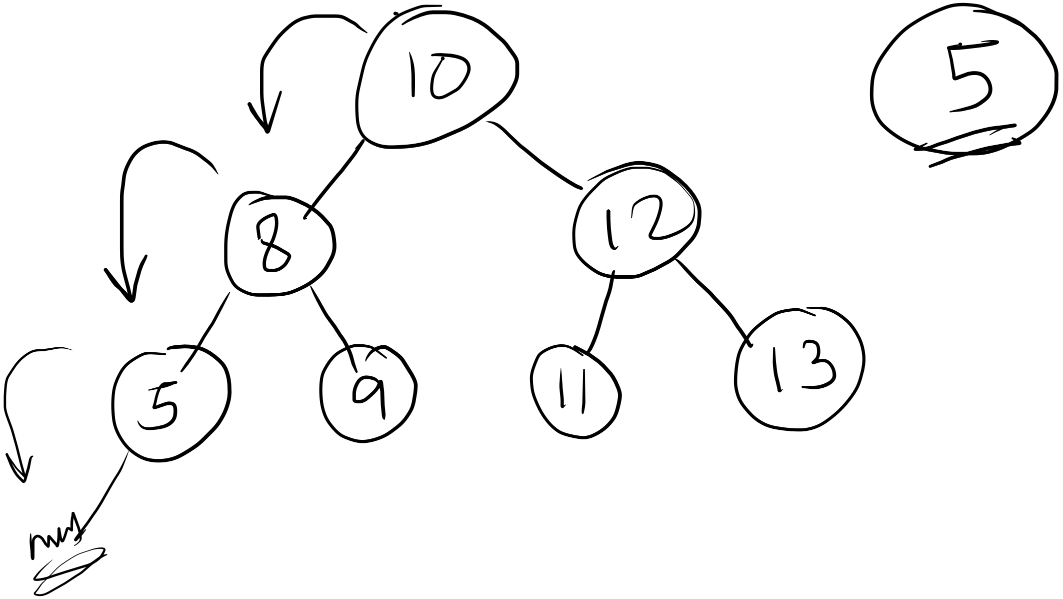
# K th Smallest Element in BST



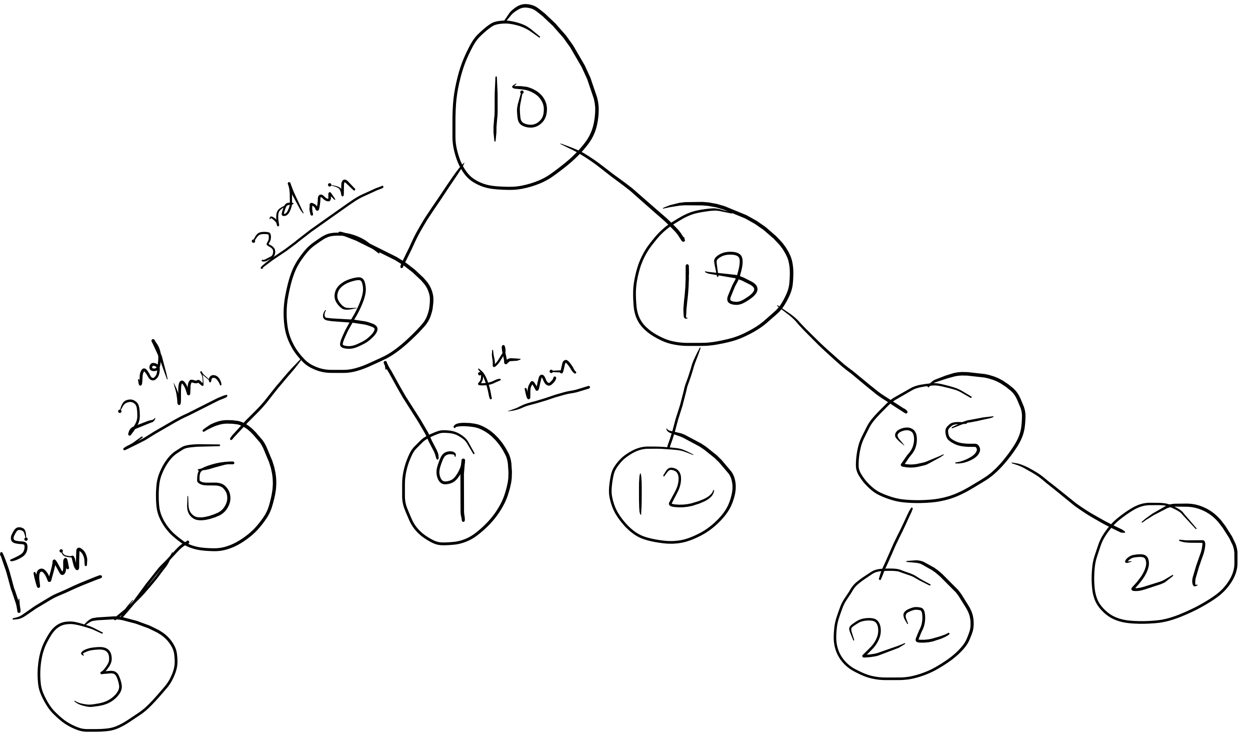
K = 4

# Smallest element in BST

→ Extre left element



# Kth Smallest Element in BST



$K^{th}$ smallest element = $\underline{arr[k-1]}$

i

K = 1 → 0
k = 2 → 1
k = 3 → 2

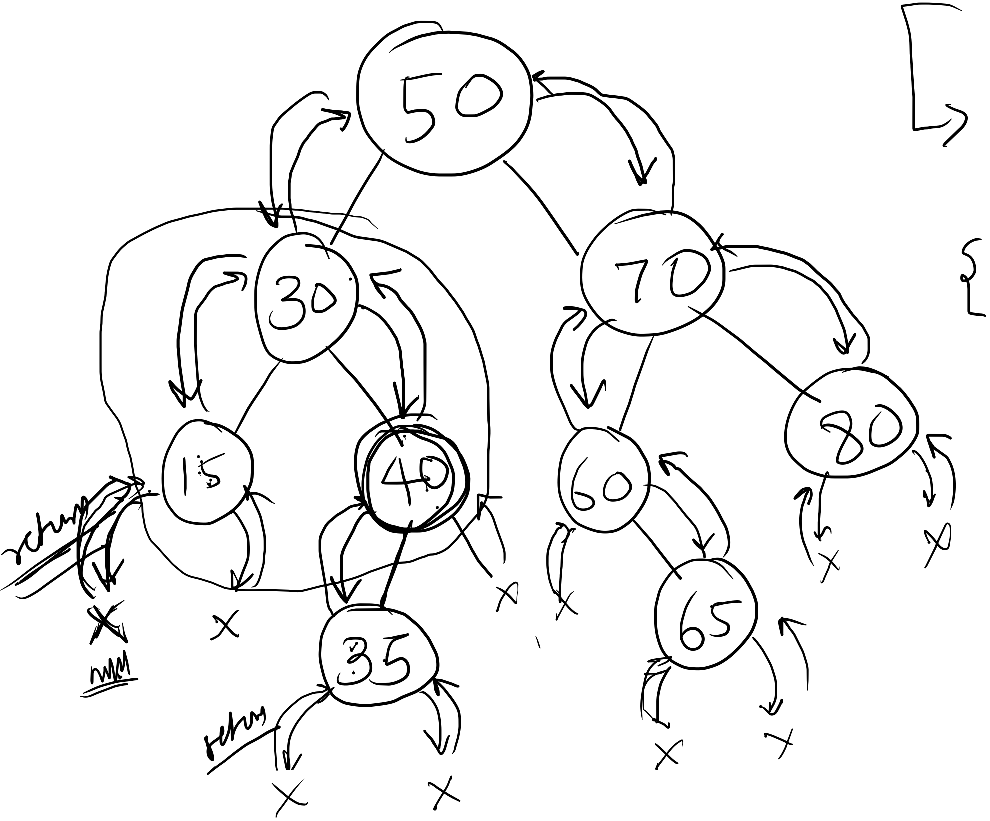
For any BST,

Inorder Traversal → Sorted

BST

↓ Inorder Traversal

Sorted Array



Inorder

left , root , right

{ 15, 30, 35, 40, 50, 60, 65,
70, 80 }

$K = 5$

Inorder

↳ left, root, right

Sorted list

L { 15, 30, 35, 40, 50, 60, 65, 70, 80 }

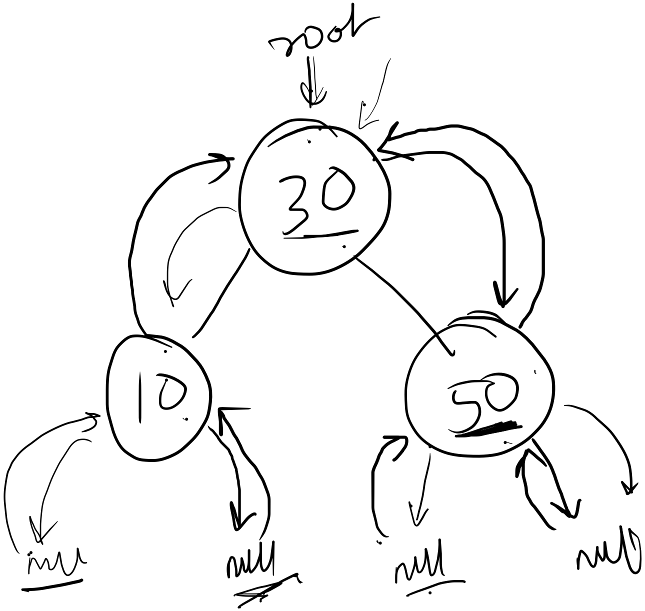
| 15 | 30 | 35 | 40 | 50 | 60 | 65 | 70 | 80 |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

$K^{th}$ smallest = $L.get(\overset{5-1}{K-1}) = L.get(4)$

$= 50$

① Inorder Traversal
( sorted list )

② get $\underset{\text{(index)}}{\underline{(K-1)}}^{th}$ element from the list



ans {10, 30, 50}

T.C = $O(n)$

SC = $O(n)$

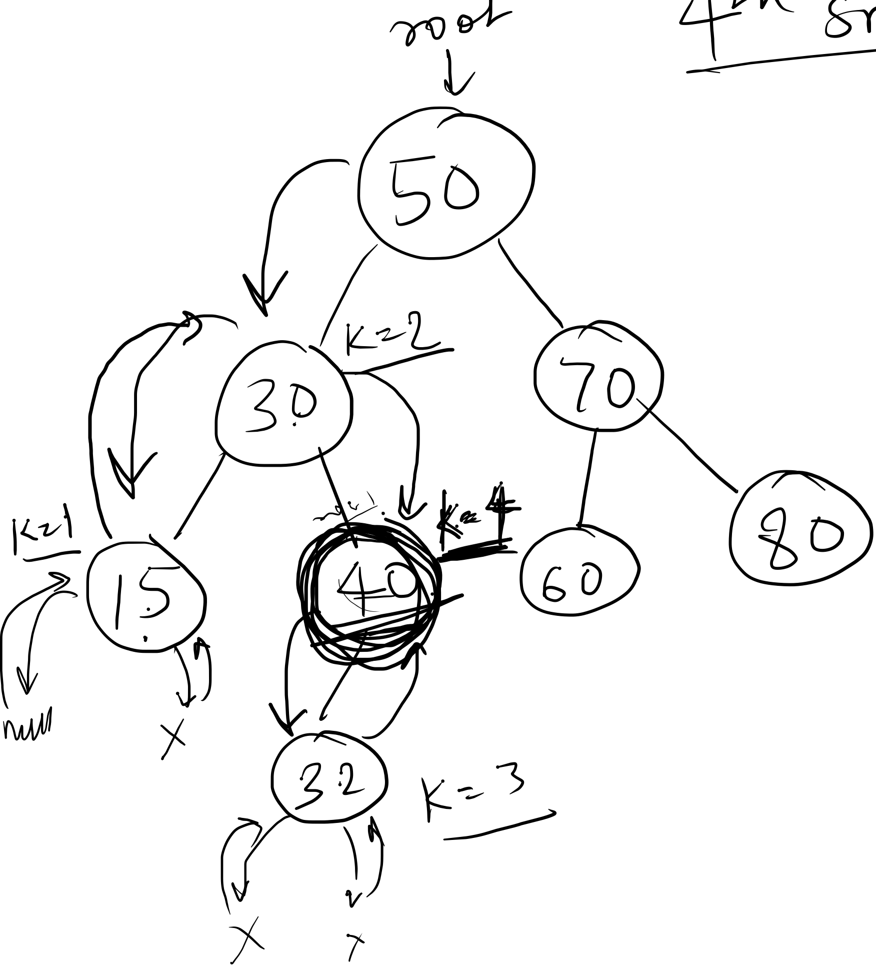
T.C = $O(n)$

SC = $O(n)$

# Optimization

1. No need of list
2. Stop when ans found (inorder)

# 4th smallest element



K = 4

count = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

Inorder

left, root, right

```
if (count == k)
    ans = root.val
```

TC = $\underline{O(n)}$

Aux space → $\underline{\text{recursive stack } O(n)}$

Aux space → $\underline{\text{active space : } O(1)}$