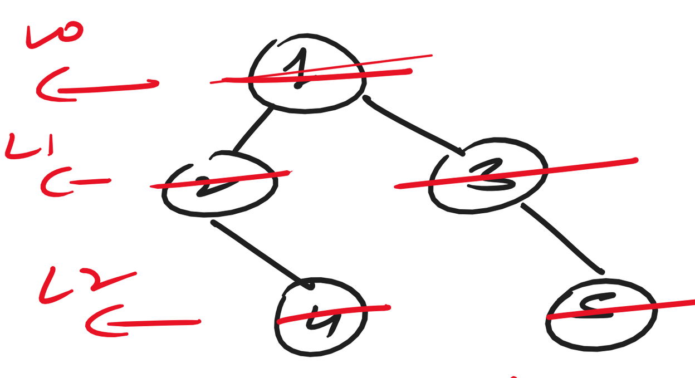
* Important Observation :→

* Constraint :→



what's the left view ?

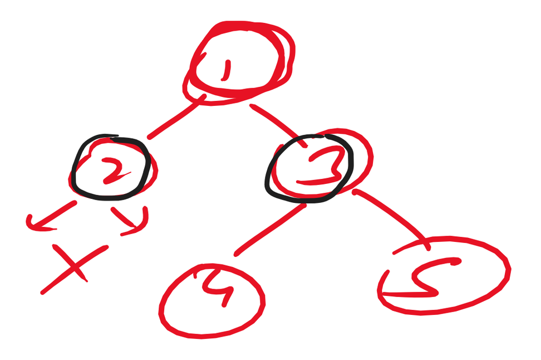
LV → 1, 2, 4

RV → 1, 3, 5

(BFS)

* For the left view always print the first node that you encounter at that level.

* For the right view always print the last node at that level.



L → 1, 2, 4

R → 1, 3, 5

poll

q[1] 0

q[1, 4, 2, 3] 0 (S-1)

(example)

1, 2

1,

q[2, 3] 0 S-1

2

2 q[4, 5] 0, S-1

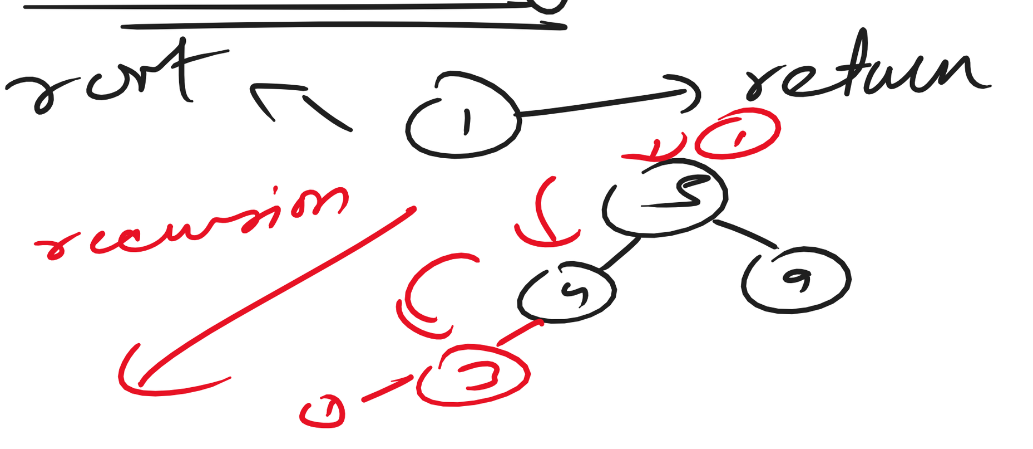
4, 5

Binary Search Tree Implementation

1. insert (root, key)
2. search (root, key)
3. delete (root, key)

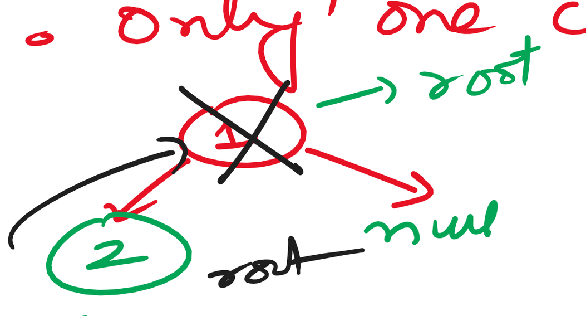
Tree is empty 1 key

root ← 1 → return root



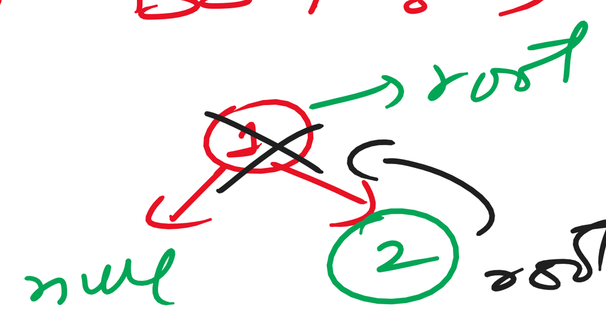
* Delete operation in a BST :→

1. Only one child




```
if (root.right == null)
{
  Tn = r.l
  root = null;
} return temp;
```

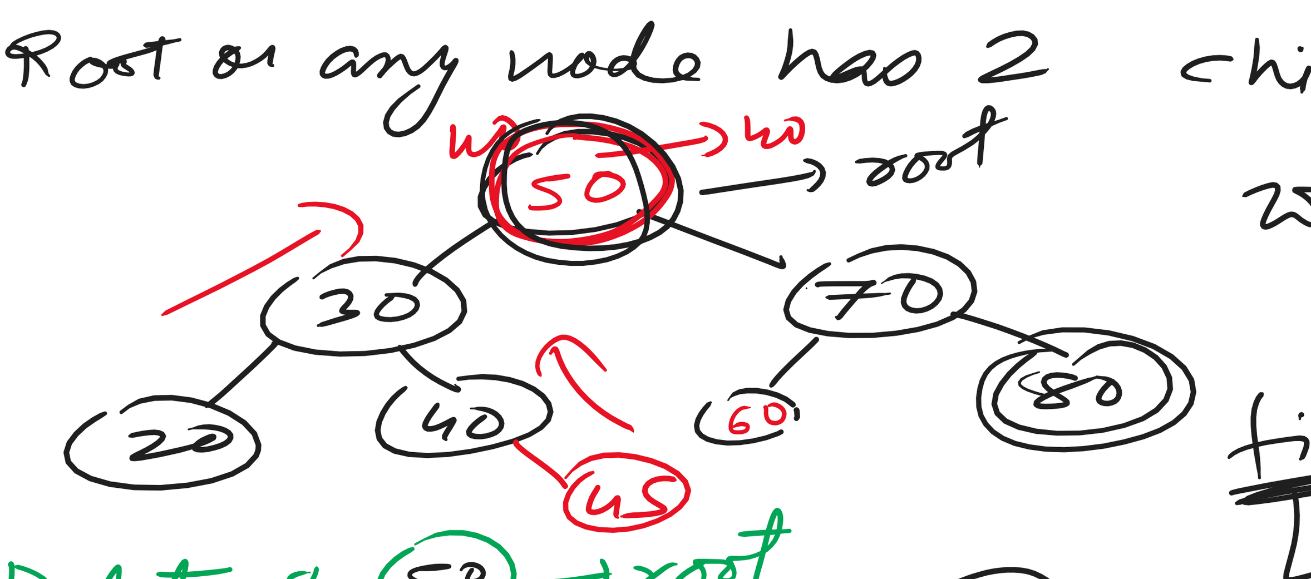
Right child doesn't exist




```
if (root.left == null)
{
  Tn = r.r
  root = null
}
return temp;
```

Left child doesn't exist

* Root or any node has 2 children :



why ? 20 → 50, 70

findMin → 60

Delete : 50 → root

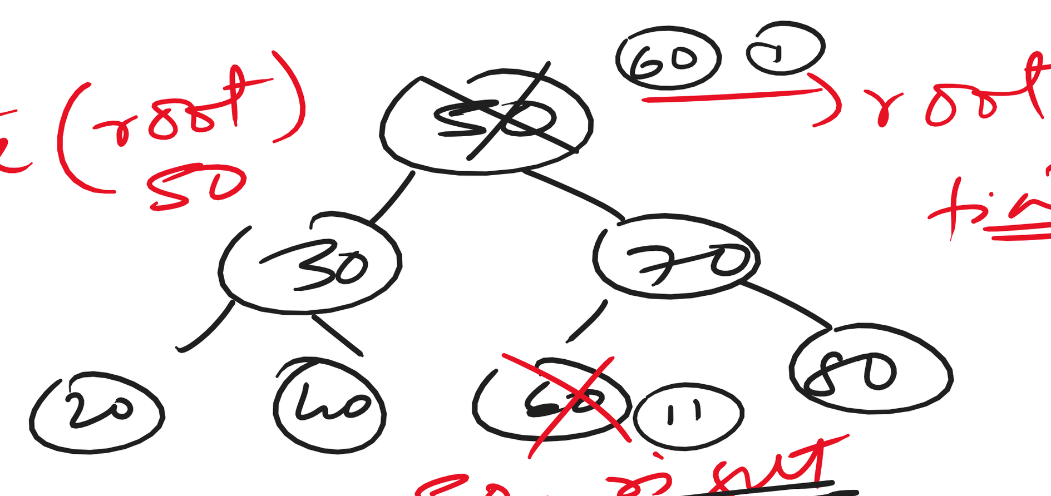
In-order : 20, 30, 40, 50, 60, 70, 80

(In Order Successor) → 60

Least on the RHS

root.right = 70

delete (root) 50



find Min

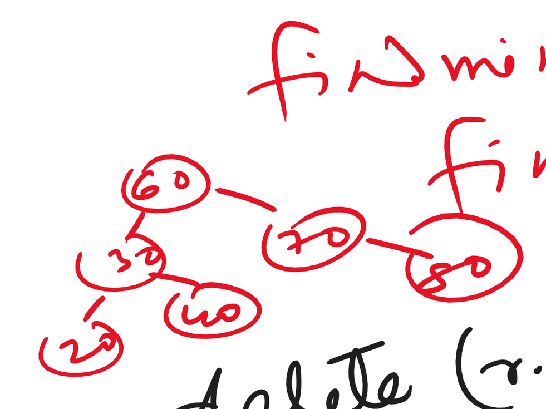
temp = 60

50.right

findmin (root.right) → 70

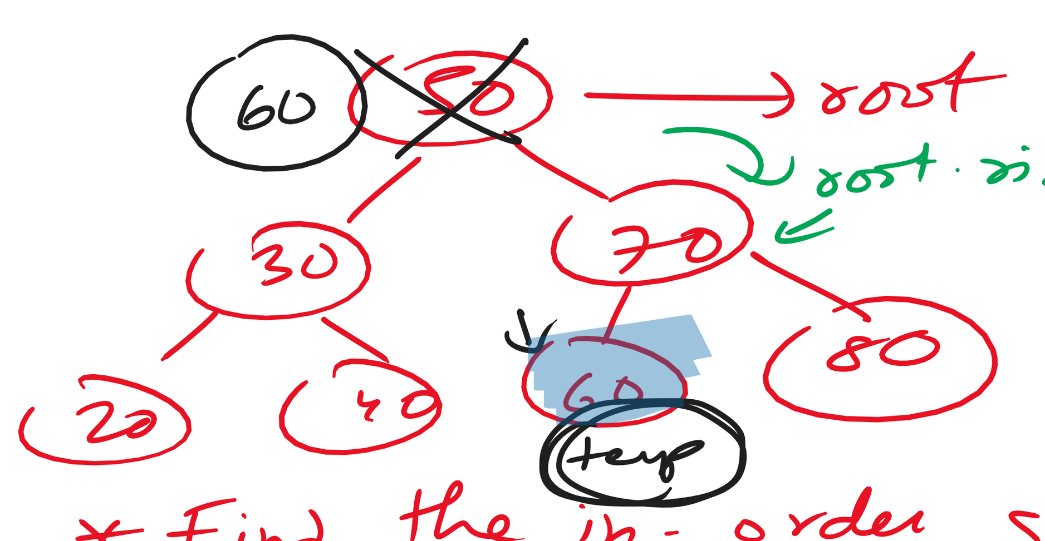
findmin (70)

60



delete (r.r, temp) (70, 60)

replacing 50 with 60



delete (50)

* Find the in-order successor

↳ Least value @ root.right

findMin (root.right)

TreeNode temp = fM (70)

temp = ↳ 60;

root = temp

delete (root.right, temp);

* 1 Problem on BST *
* Heaps → Greedy Algos + Graphs Algos
* Tries

↓

Max Heap + Min Heap

O(1)

Heap Sort → nlogn