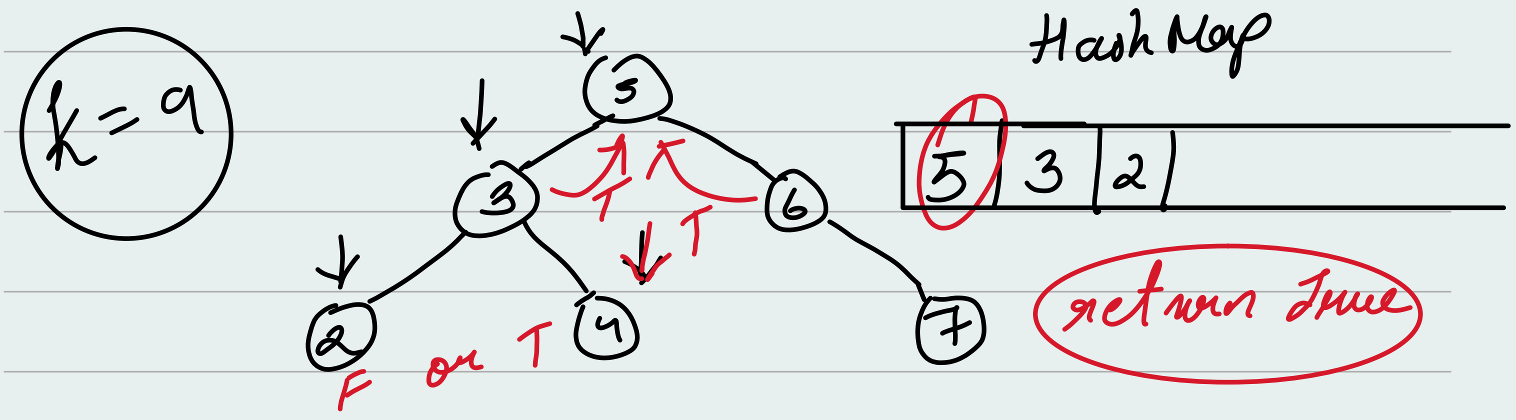
Two Sum :-



Input : root = [5, 3, 6, 2, 4, null, 7]
k = 9
Output = true

→ check (9-5) = 4 whether in HM or not?
No, it's not, put 5 in the HM and keep traversing

Space complexity : O(N).
Time complexity : O(N).

Code:

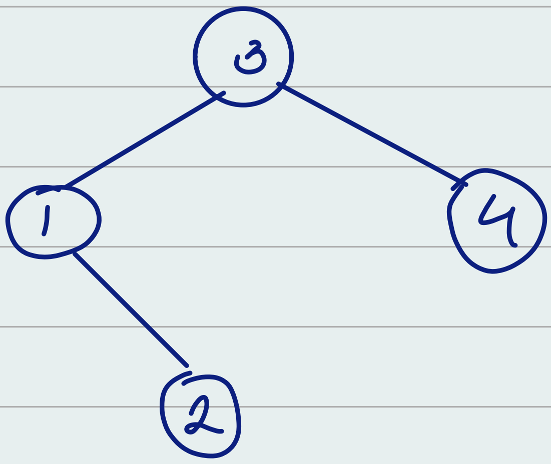
```
class TwoSum {
    public boolean findTarget (TreeNode root,
                                          int k) {
        HashSet < Integer > set = new
                             HashSet<>();
```

```
            return helper(root, k, set);
    }
    private boolean helper(TreeNode node,
                int k, HashSet<Integer> set){

            if (node == null){
                return false;
            }
            if (set.contains(k - node.val)){
                return true;
            }

            set.add(node.val);
            return helper(node.left, k,
            set) || helper(node.right, k, set);
        }
}
```

Kth Smallest Element in BST

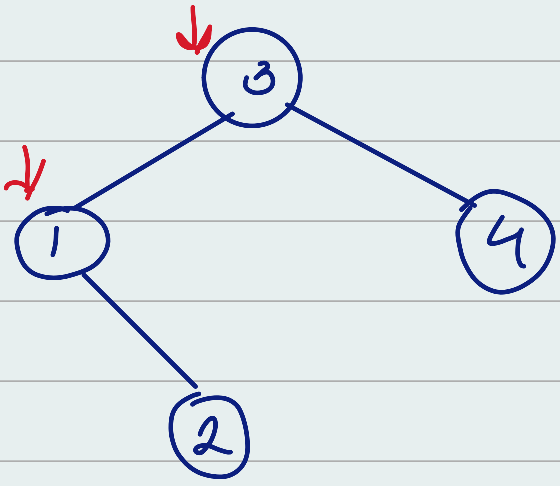
* Whenever kth smallest is given we use Heaps

→ In Order traversal for sorted order



Input: root = [3, 1, 4, null, 2],
k = 1
output = 1

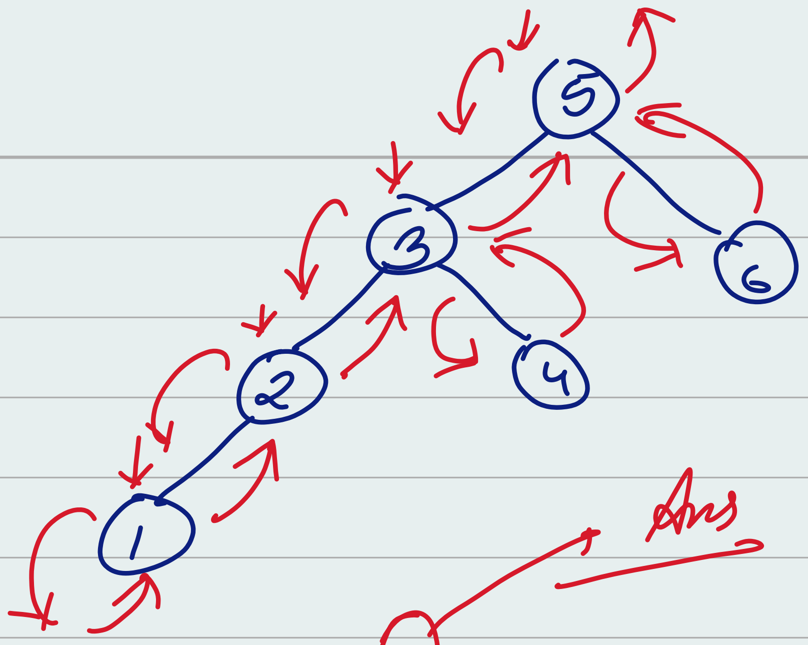
InOrder : L - N - R



$\boxed{1 - 2 - 3 - 4}$

↓

Ans



k = 3

Third smallest element

1 – 2 – 3 – 4 – 5 – 6

heap

1	2	3	4	5	6

remove k elements :

ans = 0
i < k,
ans = minHeap.poll();

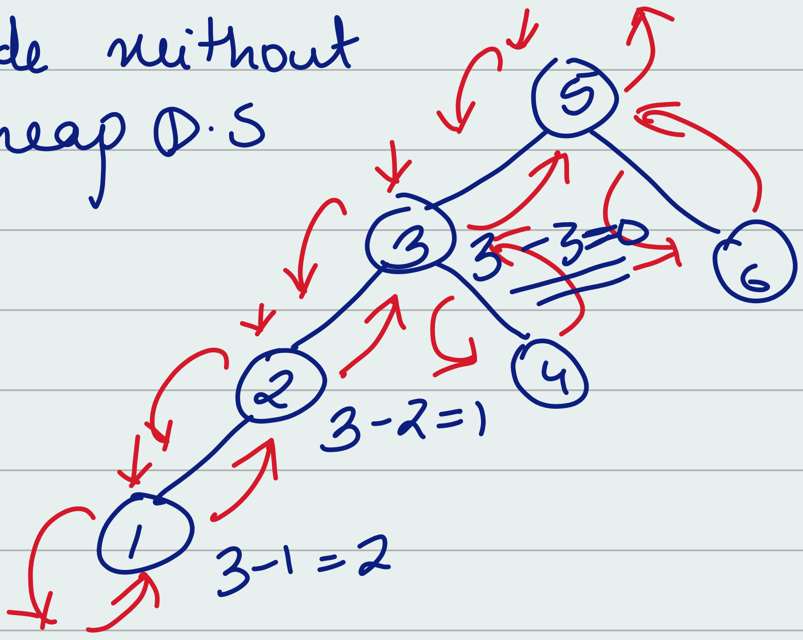
~~1~~, ~~2~~, 3 Ans

Code Using Heap data structure :-

```
class kthSmallest{

  public int kthSmallest(TreeNode root, int k){

    PriorityQueue<Integer> minHeap = new
                     PriorityQueue<>();

    helper( root, minHeap, k);
    // Remove k elements
```

```
int ans =0;
for (int i=0, i< k, i++){

   ans = minHeap.poll();

}
 return ans
}
private void helper (TreeNode node,
        PriorityQueue <Integer> minHeap,
          int k ){
   if (node == null){
      return;
   }
   helper (node.left, minHeap, k);

   minHeap offer (node.val);


   helper( node right, minHeap, k);
```

Code without
heap D.S



k =3

```
class kthSmallest{

    private int k;
     Private int ans;
public int kthSmallest(TreeNode root, int k){

        this.k = k;
        helper ( root ),
        return ans;
  }

  private void helper (TreeNode node){
      if (node == null){
          return;
      }
        helper (node. left ) ;
        k--;
        if (k==0){
            ans = node.val ,
            return;
        }
       helper (node.right);
}
```