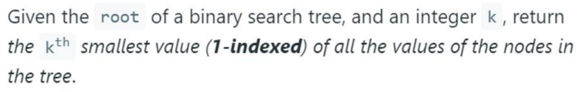
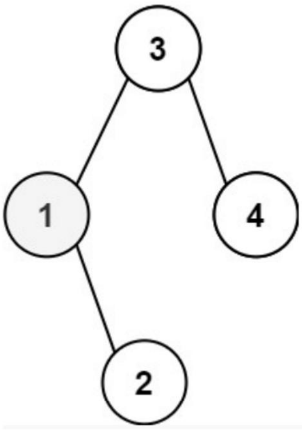
230. Kth Smallest Element in a BST

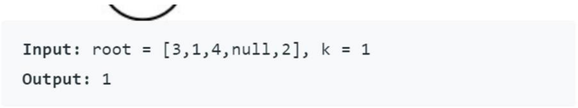
02 April 2022 05:23 PM



Given the `root` of a binary search tree, and an integer `k`, return *the* `kth` *smallest value (**1-indexed**) of all the values of the nodes in the tree.*

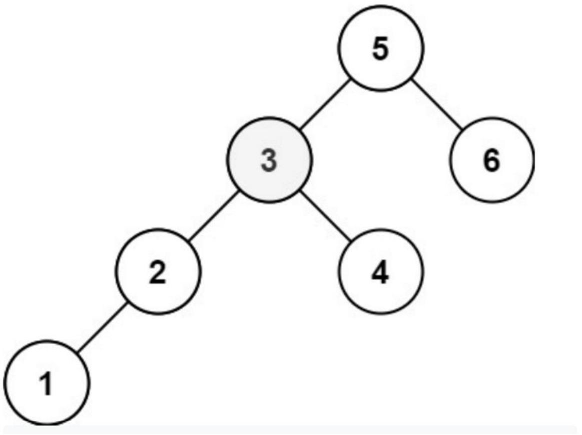
Example 1:

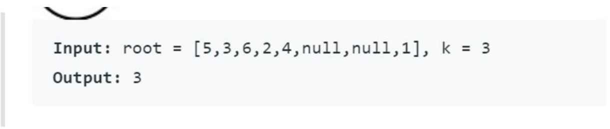




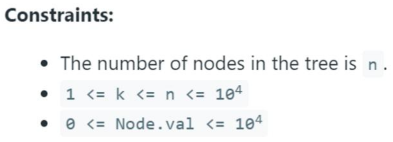
```
Input: root = [3,1,4,null,2], k = 1
Output: 1
```

Example 2:



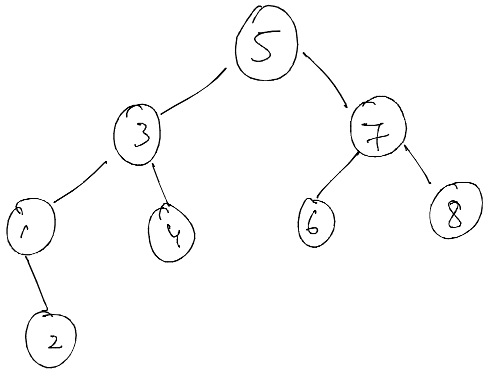


```
Input: root = [5,3,6,2,4,null,null,1], k = 3
Output: 3
```



Constraints:

- The number of nodes in the tree is `n`.
- `1 <= k <= n <= 10^4`
- `0 <= Node.val <= 10^4`



k = 3

this is the kth smallest element

1 2 3 4 5 6 7 8

DFS approach (Pre, Post, level, Inorder)

* nodes
↳ Vector/List
↳ Sort

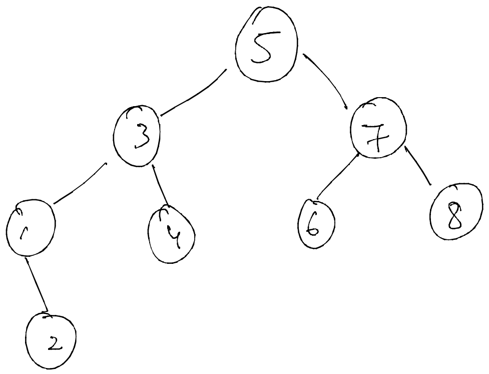
→ Vector/ List
→ Sort
→ Remove the k^{th}

T.C ⇒ $O(N) + O(N \log N)$

S.C ⇒ $O(N)$

* <u>Efficient Approach</u>

Inorder (Left Root Right)



* The <u>inorder</u> of any given <u>Binary Search Tree</u> is always <u>in sorted order</u>

1 2 3 4 5 6 7 8

S.C → $O(N)$

To avoid space, you can keep counter = 0, wherever you visit the node, do the counter++ and the moment counter == k.

```
if (cnt == k)
    ans = node
```

Recursive T.C → $O(N)$ S.C → $O(N)$

Recursive   T.C → O(N)   S.C → O(N)

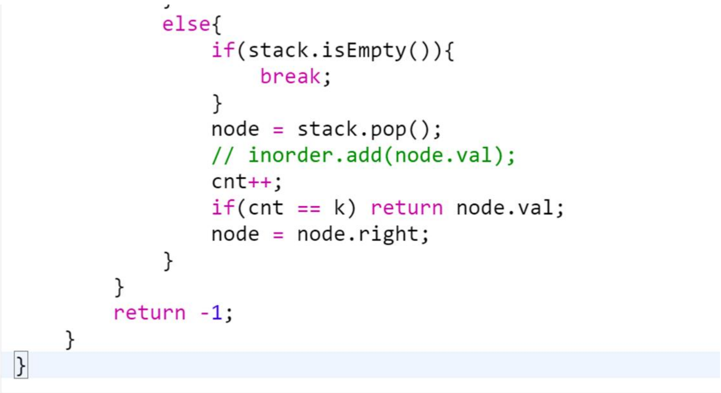
Iteration ↗

Morris Traversal   S.C ⇒ O(1)

Now for k^{th} largest :

one traversal ⟶ Ⓝ

k^{th} largest = ($N - k^{th}$) smallest



```
            else{
                if(stack.isEmpty()){
                    break;
                }
                node = stack.pop();
                // inorder.add(node.val);
                cnt++;
                if(cnt == k) return node.val;
                node = node.right;
            }
        }
        return -1;
    }
}
```