Brute Force

-> Construct the whole ordered array [1,2,3....]
-> return array[k]
-> Time: O(n)
-> Space: O(n)

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def kthSmallest(self, root: Optional[TreeNode], k: int) -> int:
        def buildArray(root):
            if not root:
                return []
            if not root.left and not root.right:
                return [root.val]
            return buildArray(root.left) + [root.val] + buildArray(root.right)

        return buildArray(root)[k - 1]
```

Inorder Iterator

-> Perform inorder traversal
-> if we reah an "action(node)" section. we check the value of k
-> if k > 1: k -= 1
-> else: return current.val
-> Time: O(k) ?
-> Space: O(k) ? Correct me If i am wrong.

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def kthSmallest(self, root: Optional[TreeNode], k: int) -> int:
        def inorder(root, k):
            if root:
                for p in inorder_subtree(root):
                    if k == 1:
                        return p.val
                    k -= 1

        def inorder_subtree(root):
            if root:
                for other in inorder_subtree(root.left):
                    yield other
                yield root
                for other in inorder_subtree(root.right):
                    yield other

        return inorder(root, k)
```