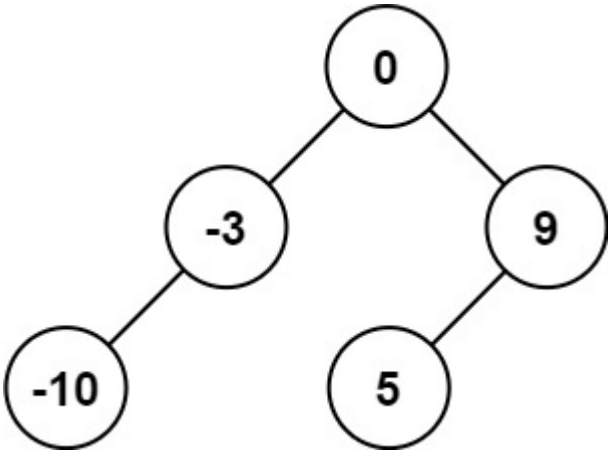
108. Convert Sorted Array to Binary Search Tree

Given an integer array `nums` where the elements are sorted in **ascending order**, convert *it to a **height-balanced** binary search tree*.

A **height-balanced** binary tree is a binary tree in which the depth of the two subtrees of every node never differs by more than one.



Input: nums = [-10,-3,0,5,9]
Output: [0,-3,9,-10,null,5]
Explanation: [0,-10,5,null,-3,null,9] is also accepted:

```
def sortedArrayToBST(self, nums: List[int]) -> TreeNode:
        root = self.helper(nums)
        return root


    def helper(self,nums):
        if len(nums)==0:
            return
        idx = len(nums)//2
        node = TreeNode(nums[idx])
        node.left = self.helper(nums[0:idx])
        node.right = self.helper(nums[idx+1:])
        return node
```

Not a good solution as slice takes O(n) time so it is O(nlogn) TC.

A better approach would be to just have start and end indices.

```
def sortedArrayToBST(self, nums: List[int]) -> TreeNode:
        root = self.helper(nums,0,len(nums))
        return root


    def helper(self,nums,lo,hi):
        if lo==hi:
            return
        idx = (lo+hi)//2
        node = TreeNode(nums[idx])
        node.left = self.helper(nums,lo,idx)
        node.right = self.helper(nums,idx+1,hi)
        return node
```

TC is O(n)