26. You are given an integer array nums with no duplicates. A maximum binary tree can be built recursively from nums using the following algorithm: Create a root node whose value is the maximum value in nums. Recursively build the left subtree on the subarray prefix to the left of the maximum value. Recursively build the right subtree on the subarray suffix to the right of the maximum value. Return the maximum binary tree built from nums.

PROGRAM:

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def constructMaximumBinaryTree(nums):
    if not nums:
        return None
    max_index = nums.index(max(nums))
    root = TreeNode(nums[max_index])
    root.left = constructMaximumBinaryTree(nums[:max_index])
    root.right = constructMaximumBinaryTree(nums[max_index + 1:])
    return root

def print_tree(node, level=0, label="."):
    prefix = " " * (4 * level) + label + ": "
    print(prefix + str(node.val) if node else prefix + "None")
    if node:
        if node.left or node.right:
            print_tree(node.left, level + 1, "L")
            print_tree(node.right, level + 1, "R")

nums = [3, 2, 1, 6, 0, 5]
root = constructMaximumBinaryTree(nums)
```

print_tree(root)

OUTPUT:

```
PS C:\Users\chall\OneDrive\Desktop\DAA> & C:/Users/chall/AppData/Local/Programs/Python/Python312/python.exe
"
.: 6
    L: 3
        L: None
        R: 2
            L: None
            R: 1
    R: 5
        L: 0
        R: None
PS C:\Users\chall\OneDrive\Desktop\DAA>
```

TIME COMPLEXITY:

Time complexity for the above code is O(n2)+O(n)