

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 printLevelOrder(root):
```

```
    if not root:
```

```
        return
```

```
    queue = [root]
```

```
    while queue:
```

```
        node = queue.pop(0)
```

```
        print(node.val, end=" ")
```

```
        if node.left:
```

```
            queue.append(node.left)
```

```
        if node.right:
```

```
            queue.append(node.right)
```

```
nums = [3, 2, 1, 6, 0, 5]
root = constructMaximumBinaryTree(nums)
print("Constructed Maximum Binary Tree:")
printLevelOrder(root)
```

OUTPUT:

```
Constructed Maximum Binary Tree:
6 3 5 2 0 1
```

TIME COMPLEXITY: $O(n \log n)$