Python O(N) Solution - LeetCode Discuss

1

First, we make the first element in the array as the root node.
As we move from left to right, there are three scenarios:

1. The number **n** is so far the smallest. We then make this n the rightmost leaf.
2. The number **n** is smaller than the root node (which means that n will go down the right of the root), but larger than some of the internal nodes or leaves **m**. We then replace the position of **m** with **n** and make **m** the left child node of **n** (since **n** is larger than **m** and **m** appears before **n**).
3. The number **n** is larger than the root node. We make **n** the new root node and make the previous root node the left child node to **n**.

If you are confused, remember:

1. If a node **n** is a left child of another node **p**, it means that **n** comes *before* **p** and **n** is smaller than **p**.
2. If a node **n** is a right child of another node **p**, it means that **n** comes *after* **p** and **n** is smaller than **p**.

```
class Solution(object):
    def constructMaximumBinaryTree(self, nums):
        """
        :type nums: List[int]
        :rtype: TreeNode
        """
        if not nums:
            return None
        root = TreeNode(nums[0])

        for v in nums[1:]:
            if v < root.val:
                tmp = root
                while tmp.right != None and tmp.right.val > v:
                    tmp = tmp.right
                if tmp.right == None:
                    tmp.right = TreeNode(v)
                else:
                    node = TreeNode(v)
                    node.left = tmp.right
                    tmp.right = node
            else:
                node = TreeNode(v)
                node.left = root
```

```
                root = node

        return root
```