654. Maximum Binary Tree

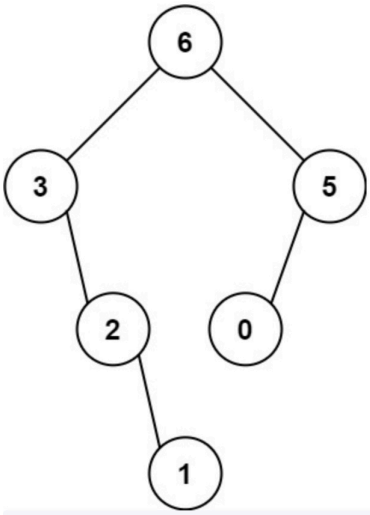
Medium 2330 266 Add to List Share

You are given an integer array `nums` with no duplicates. A **maximum binary tree** can be built recursively from `nums` using the following algorithm:

1. Create a root node whose value is the maximum value in `nums`.
2. Recursively build the left subtree on the **subarray prefix** to the **left** of the maximum value.
3. Recursively build the right subtree on the **subarray suffix** to the **right** of the maximum value.

Return *the **maximum binary tree** built from* `nums`.

Example 1:




```
Input: nums = [3,2,1,6,0,5]
Output: [6,3,5,null,2,0,null,null,1]
Explanation: The recursive calls are as follow:
- The largest value in [3,2,1,6,0,5] is 6. Left prefix is [3,2,1] and right
suffix is [0,5].
    - The largest value in [3,2,1] is 3. Left prefix is [] and right suffix is
[2,1].
        - Empty array, so no child.
        - The largest value in [2,1] is 2. Left prefix is [] and right suffix
is [1].
            - Empty array, so no child.
            - Only one element, so child is a node with value 1.
    - The largest value in [0,5] is 5. Left prefix is [0] and right suffix is
[].
        - Only one element, so child is a node with value 0.
        - Empty array, so no child.
```

For each root node, find the max value and index in nums, and construct child trees using rest of array recursively.

```cpp
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* constructMaximumBinaryTree(vector<int>& nums) {
        TreeNode *root = build(nums, 0, nums.size() - 1);
        return root;
    }

private:
    TreeNode* build(vector<int>& nums, int lo, int hi) {
        // base case
        if (lo > hi) {
            return NULL;
        }

        int rootVal = INT_MIN;
        int rootInd = -1;
        for (int i = lo; i <= hi; i++) {
            if (nums[i] > rootVal) {
                rootVal = nums[i];
                rootInd = i;
            }
        }
        TreeNode *root = new TreeNode(rootVal);

        root->left = build(nums, lo, rootInd - 1);
        root->right = build(nums, rootInd + 1, hi);

        return root;
    }
};
```