Problem 654: Maximum Binary Tree

Problem Information

Difficulty: Medium
Acceptance Rate: 86.20%
Paid Only: No
Tags: Array, Divide and Conquer, Stack, Tree, Monotonic Stack, Binary Tree

Problem Description

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:

![](https://assets.leetcode.com/uploads/2020/12/24/tree1.jpg)

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.

Example 2:

![](https://assets.leetcode.com/uploads/2020/12/24/tree2.jpg)

Input: nums = [3,2,1] **Output:** [3,null,2,null,1]

Constraints:

* `1 <= nums.length <= 1000` * `0 <= nums[i] <= 1000` * All integers in `nums` are **unique**.

Code Snippets

C++:

```
/**
* 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) {

}
};
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public TreeNode constructMaximumBinaryTree(int[] nums) {

}
}
```

Python3:

```
# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def constructMaximumBinaryTree(self, nums: List[int]) -> Optional[TreeNode]:
```