# 题目

给定一个不含重复元素的整数数组nums。一个以此数组直接递归构建的 最大二叉树 定义如下：

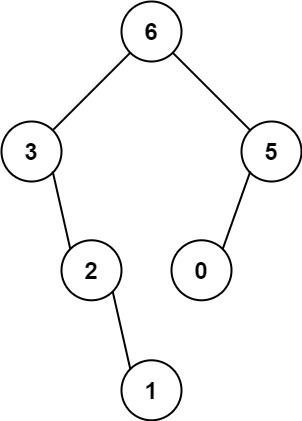
二叉树的根是数组nums中的最大元素。

左子树是通过数组中 最大值左边部分 递归构造出的最大二叉树。

右子树是通过数组中 最大值右边部分 递归构造出的最大二叉树。

返回有给定数组nums构建的最大二叉树 。

示例 1：



输入：nums = [3,2,1,6,0,5]

输出：[6,3,5,null,2,0,null,null,1]

解释：递归调用如下所示：

- [3,2,1,6,0,5] 中的最大值是 6 ，左边部分是 [3,2,1] ，右边部分是 [0,5] 。

- [3,2,1] 中的最大值是 3 ，左边部分是 [] ，右边部分是 [2,1] 。

- 空数组，无子节点。

- [2,1] 中的最大值是 2 ，左边部分是 [] ，右边部分是 [1] 。

- 空数组，无子节点。

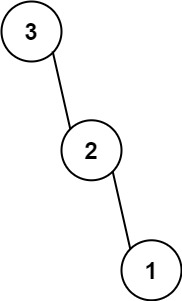
- 只有一个元素，所以子节点是一个值为 1 的节点。

- [0,5] 中的最大值是 5 ，左边部分是 [0] ，右边部分是 [] 。

- 只有一个元素，所以子节点是一个值为 0 的节点。

- 空数组，无子节点。

示例 2：



输入：nums = [3,2,1]

输出：[3,null,2,null,1]

提示：

1 <= nums.length <= 1000

0 <= nums[i] <= 1000

nums 中的所有整数互不相同

# 分析

class Solution {

public:

TreeNode\* constructMaximumBinaryTree(vector<int>& nums) {

return dfs(nums, 0, nums.size() - 1);

}

TreeNode\* dfs(vector<int>& nums, int l, int r) {

if (l > r) return nullptr;

int idx = l;

for (int i = l + 1; i <= r; ++i)

if (nums[i] > nums[idx]) idx = i;

TreeNode\* root = new TreeNode(nums[idx]);

root->left = dfs(nums, l, idx - 1);

root->right = dfs(nums, idx + 1, r);

return root;

}

};