Problem 1008: Construct Binary Search Tree from Preorder Traversal

Problem Information

Difficulty: Medium
Acceptance Rate: 83.87%
Paid Only: No
Tags: Array, Stack, Tree, Binary Search Tree, Monotonic Stack, Binary Tree

Problem Description

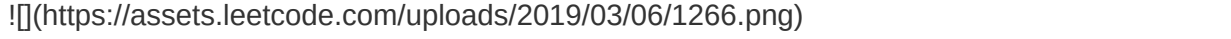
Given an array of integers preorder, which represents the **preorder traversal** of a BST (i.e., **binary search tree**), construct the tree and return _its root_.

It is **guaranteed** that there is always possible to find a binary search tree with the given requirements for the given test cases.

A **binary search tree** is a binary tree where for every node, any descendant of `Node.left` has a value **strictly less than** `Node.val`, and any descendant of `Node.right` has a value **strictly greater than** `Node.val`.

A **preorder traversal** of a binary tree displays the value of the node first, then traverses `Node.left`, then traverses `Node.right`.

Example 1:



Input: preorder = [8,5,1,7,10,12] **Output:** [8,5,10,1,7,null,12]

Example 2:

Input: preorder = [1,3] **Output:** [1,null,3]

Constraints:

* `1 <= preorder.length <= 100` * `1 <= preorder[i] <= 1000` * All the values of `preorder` 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* bstFromPreorder(vector<int>& preorder) {

}
};
```

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 bstFromPreorder(int[] preorder) {

}
}
```

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 bstFromPreorder(self, preorder: List[int]) -> Optional[TreeNode]:
```