1008. Construct Binary Search Tree from Preorder Traversal

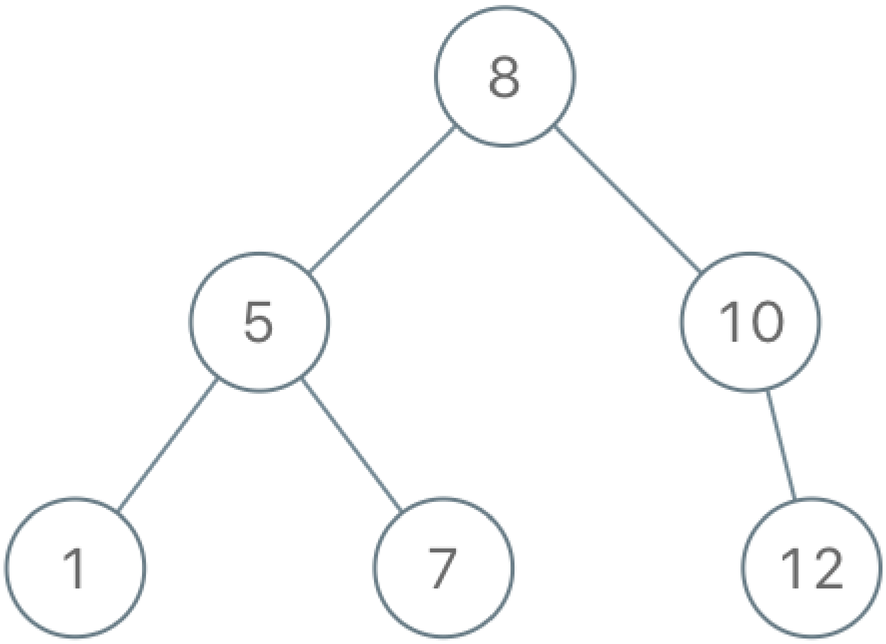
Given an array of integers preorder, which represents the **preorder traversal** of a BST (i.e., **binary search tree**), construct the tree and return *its root*.

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**.

```
import sys
class Solution:
    def bstFromPreorder(self, preorder: List[int]) -> Optional[TreeNode]:
        low = -sys.maxsize
        hi = sys.maxsize
        self.idx = 0
        return self.bstFromPreOrderHelper(preorder,low,hi)



    def bstFromPreOrderHelper(self,preorder,low,hi):
        if self.idx>=len(preorder) or preorder[self.idx]<low or 
preorder[self.idx]>hi:
            return None
        node = TreeNode(preorder[self.idx])
        self.idx+=1
        node.left = self.bstFromPreOrderHelper(preorder,low,node.val)
        node.right = self.bstFromPreOrderHelper(preorder,node.val,hi)
        return node
```

```
class Solution:
    def bstFromPreorder(self, preorder: List[int]) -> TreeNode:
        inorder = sorted(preorder)
        root = self.helper(preorder,inorder)
        return root



    def helper(self,preorder,inorder):
        if len(inorder)==0:
            return None
        node = TreeNode(preorder[0])
        idx = inorder.index(preorder[0])
        node.left = self.helper(preorder[1:idx+1],inorder[:idx])
        node.right = self.helper(preorder[idx+1:],inorder[idx+1:])
        return node
```

```python
class Solution:
    def bstFromPreorder(self, preorder: List[int]) -> TreeNode:
        if not preorder:
            return None
        root = TreeNode(preorder[0])
        i = 1
        while i<len(preorder) and  preorder[i] < root.val:
            i+=1
        root.left = self.bstFromPreorder(preorder[1:i])
        root.right = self.bstFromPreorder(preorder[i:])
        return root
```