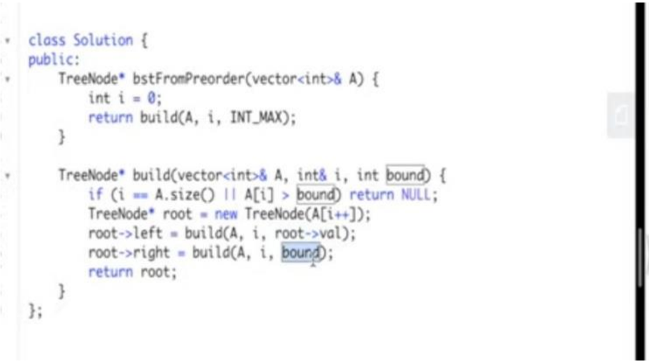
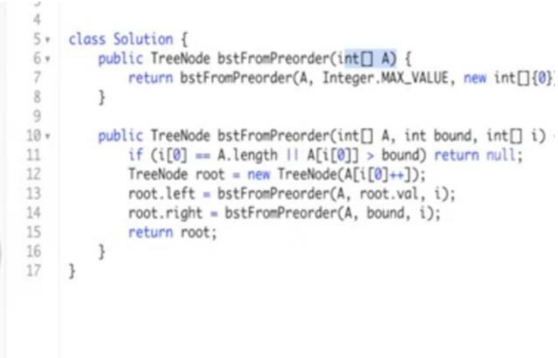
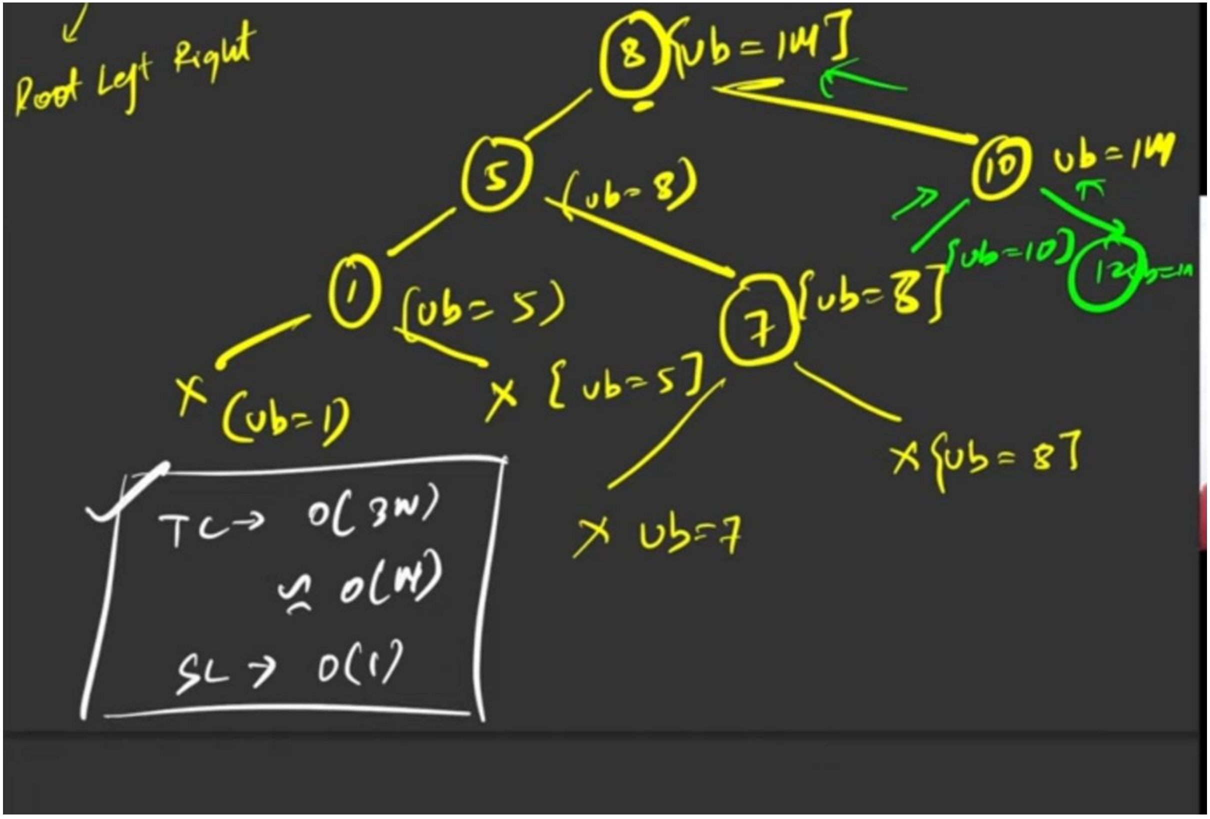
L48. Construct a BST from a preorder traversal | 3 Methods

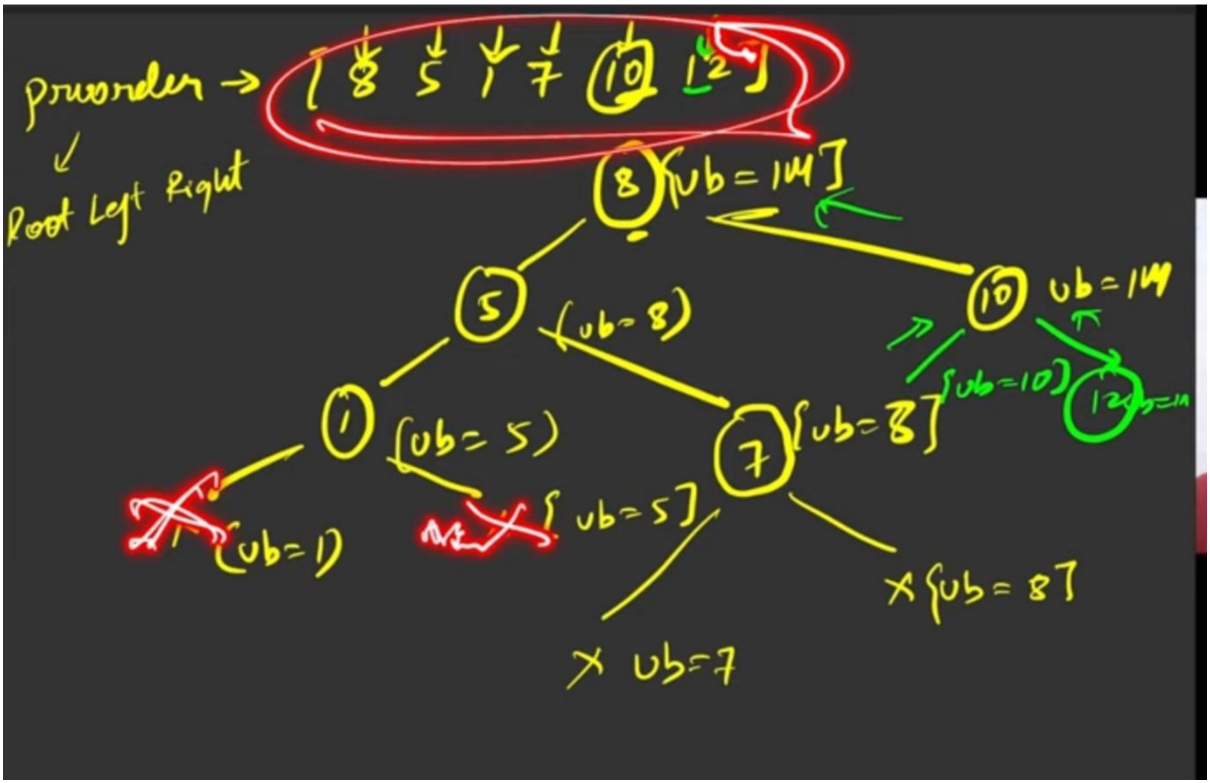


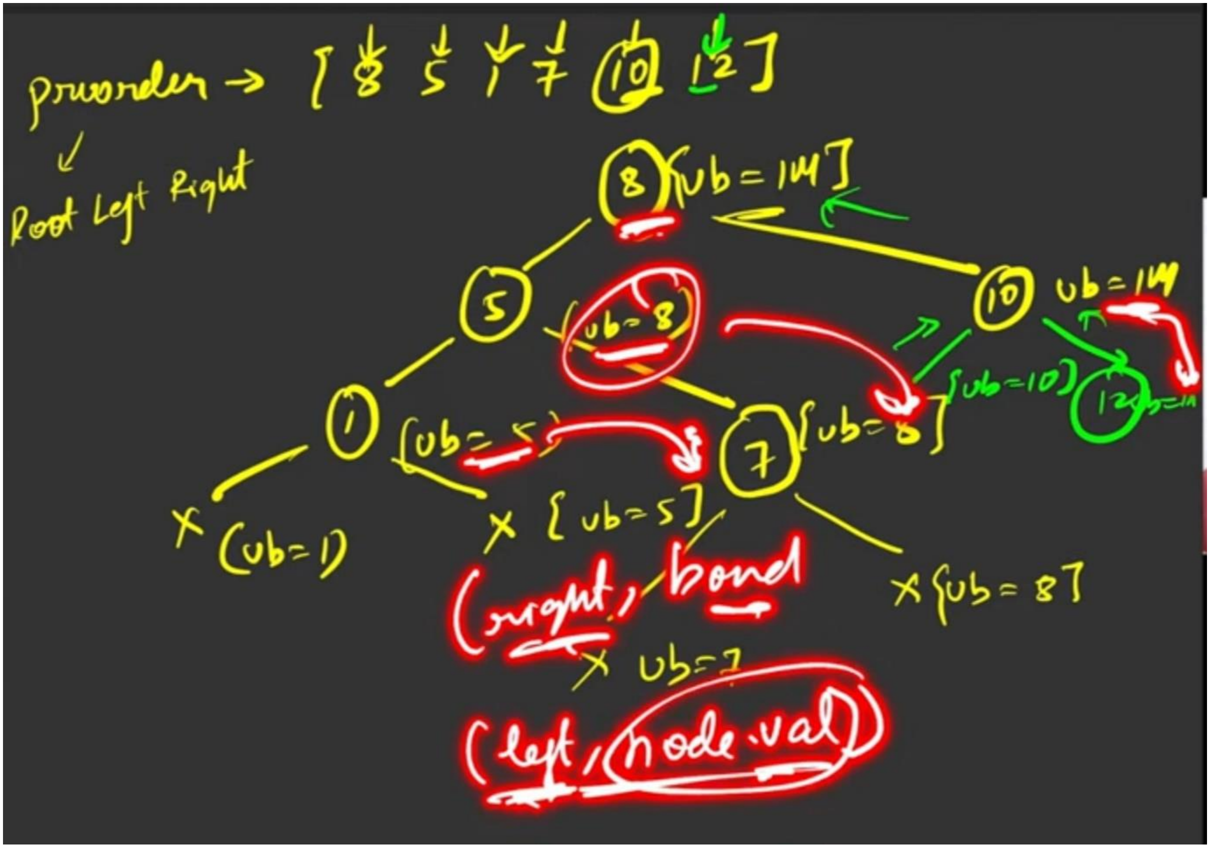
```cpp
class Solution {
public:
    TreeNode* bstFromPreorder(vector<int>& A) {
        int i = 0;
        return build(A, i, INT_MAX);
    }

    TreeNode* build(vector<int>& A, int& i, int bound) {
        if (i == A.size() || A[i] > bound) return NULL;
        TreeNode* root = new TreeNode(A[i++]);
        root->left = build(A, i, root->val);
        root->right = build(A, i, bound);
        return root;
    }
};
```

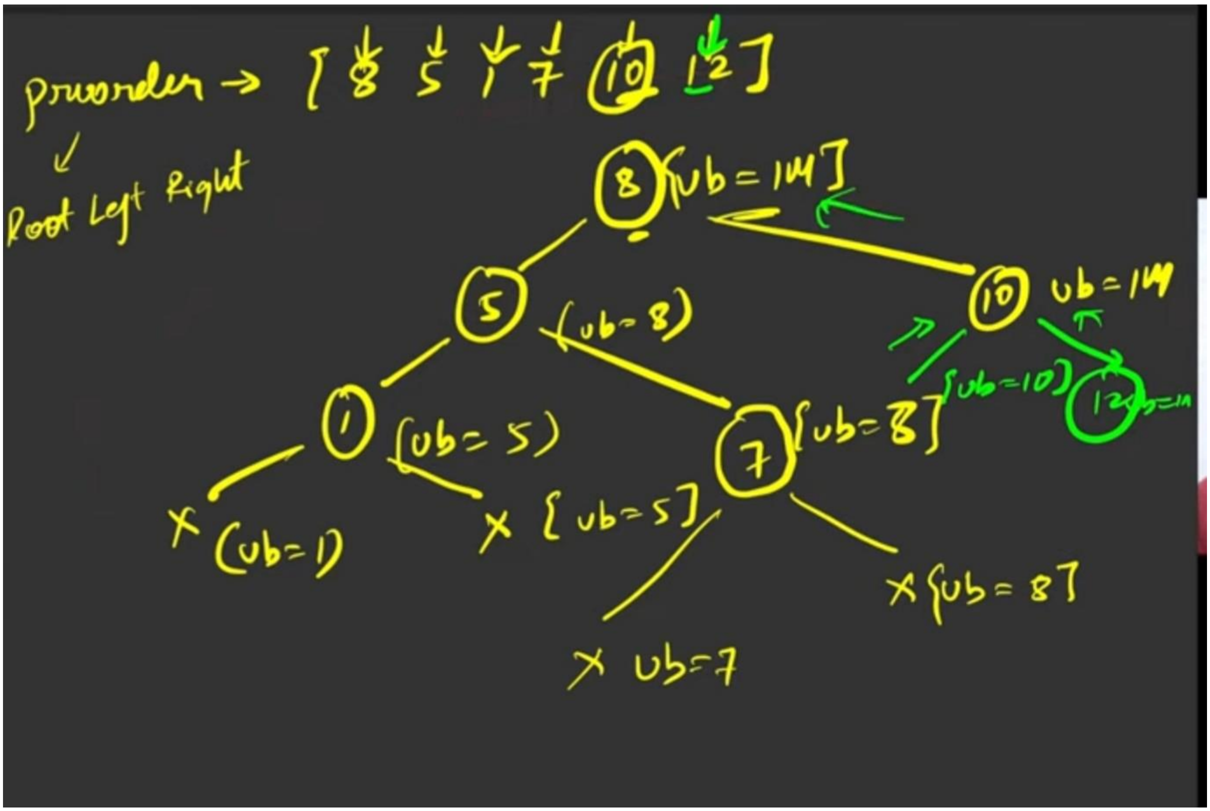


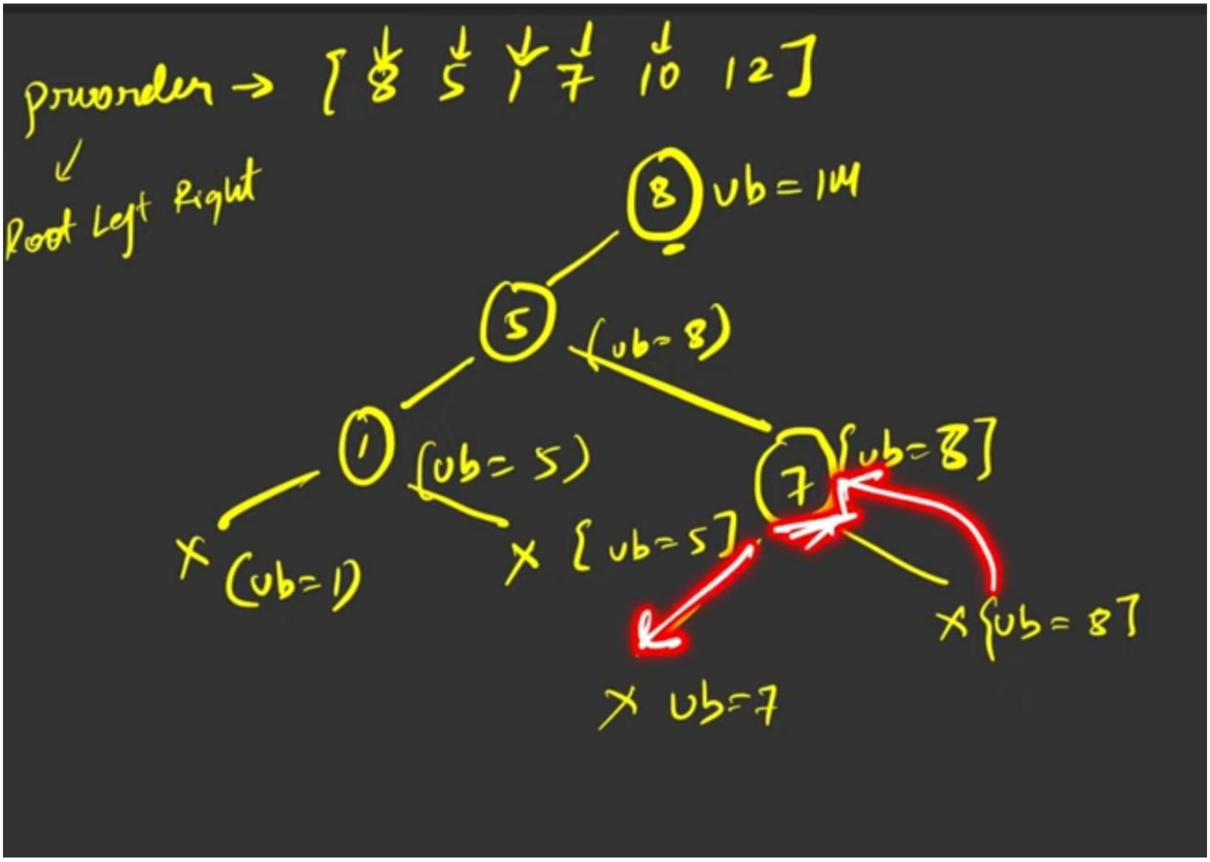
```java
class Solution {
    public TreeNode bstFromPreorder(int[] A) {
        return bstFromPreorder(A, Integer.MAX_VALUE, new int[]{0}

    }

    public TreeNode bstFromPreorder(int[] A, int bound, int[] i)
        if (i[0] == A.length || A[i[0]] > bound) return null;
        TreeNode root = new TreeNode(A[i[0]++]);
        root.left = bstFromPreorder(A, root.val, i);
        root.right = bstFromPreorder(A, bound, i);
        return root;
    }
}
```

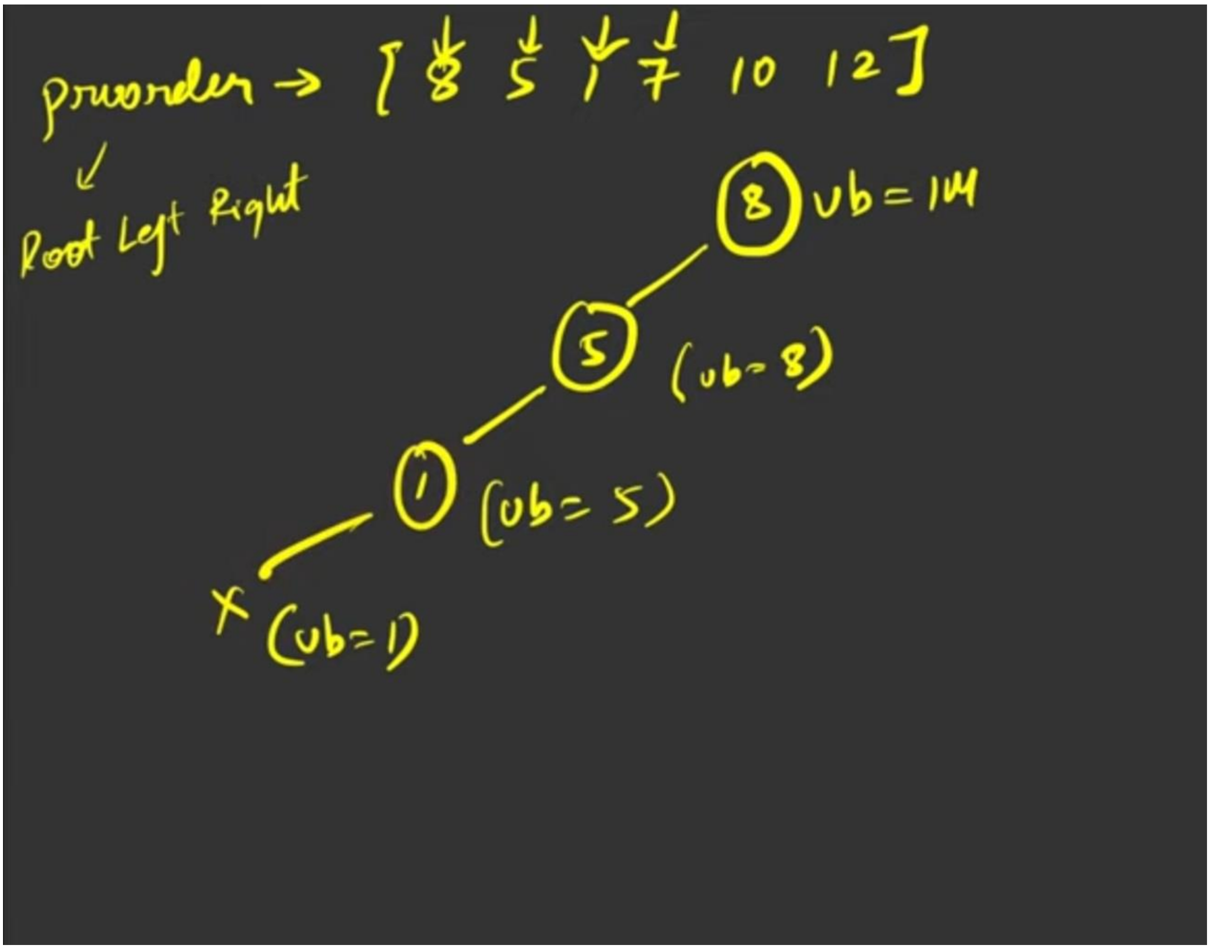



preorder → [8 5 1 7 10 12]
Root Left Right
8 {ub = 14]
5 {ub = 8)
1 {ub = 5)
7 {ub = 8]
10 ub = 14
{ub = 10]
12 ub = 14
{ub = 1)
{ub = 5]
× {ub = 8]
× ub = 7


preorder → [8 5 1 7 10 12]
Root Left Right
8 {ub = 14]
5 {ub = 8)
1 {ub = 5)
7 {ub = 8]
10 ub = 14
{ub = 10]
12 ub = 14
× (ub = 1)
× {ub = 5]
× {ub = 8]
× ub = 7
(right, bond)
(left, node.val)

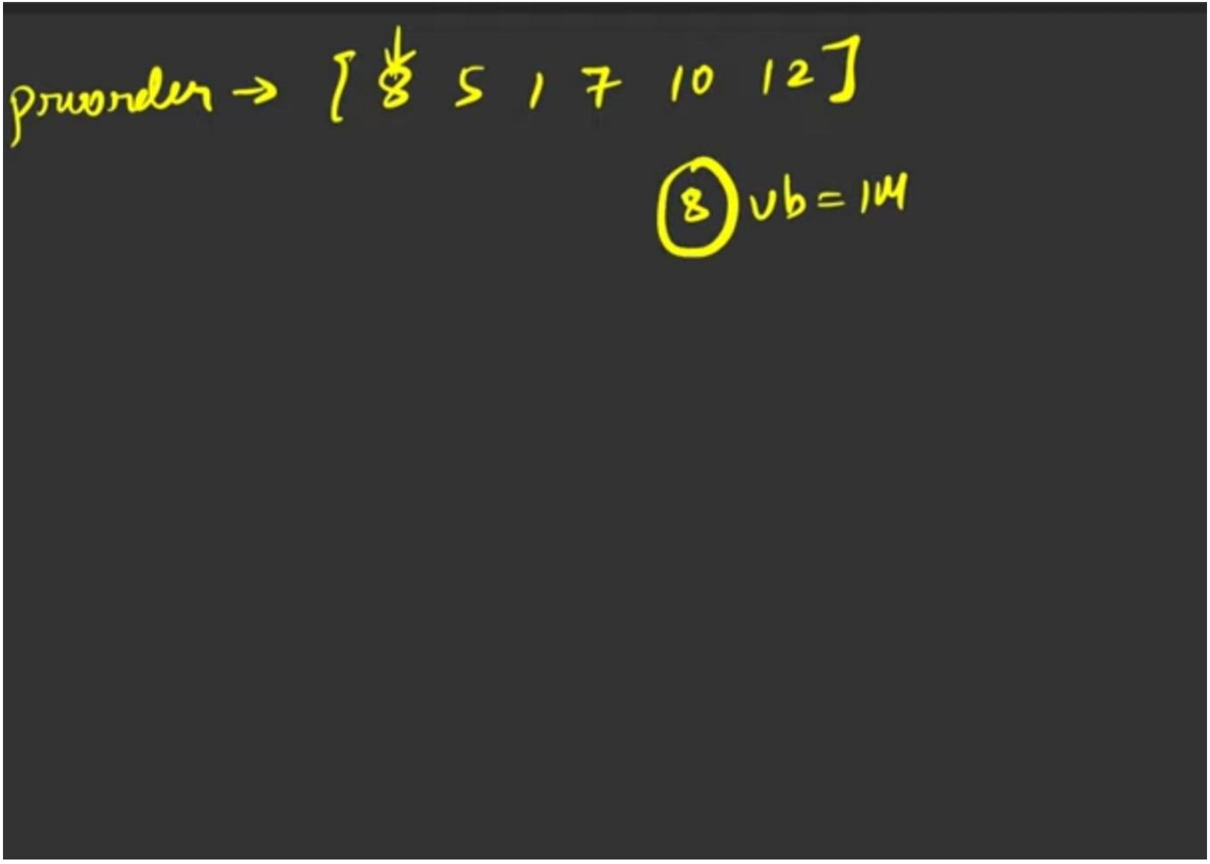

preorder → [8 5 1 7 10 12]
Root Left Right
8 {ub=14]
5 (ub=8)
10 ub=14
1 {ub=5)
7 {ub=8]
{ub=10]
12 ub=14
X (ub=1)
X [ub=5]
X {ub=8]
X ub=7


preorder → [8 5 1 7 10 12]
Root Left Right
8 ub=14
5 (ub=8)
1 {ub=5)
7 {ub=8]
X (ub=1)
X [ub=5]
X {ub=8]
X ub=7

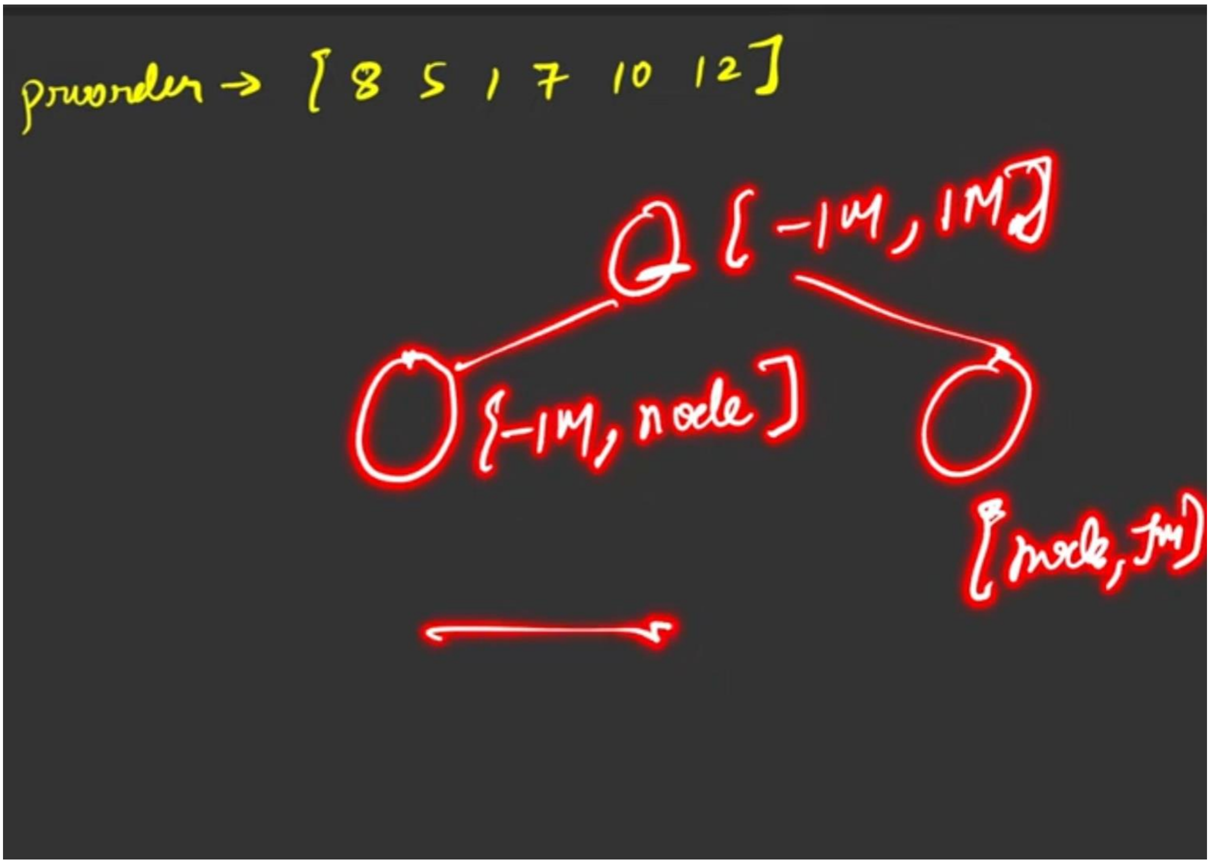


preorder → [8 5 1 7 10 12]

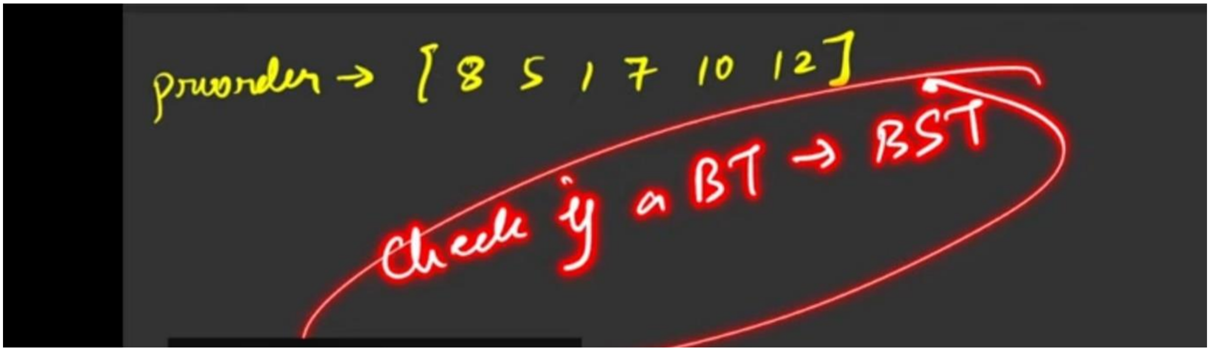
Root Left Right



preorder → [8 5 1 7 10 12]

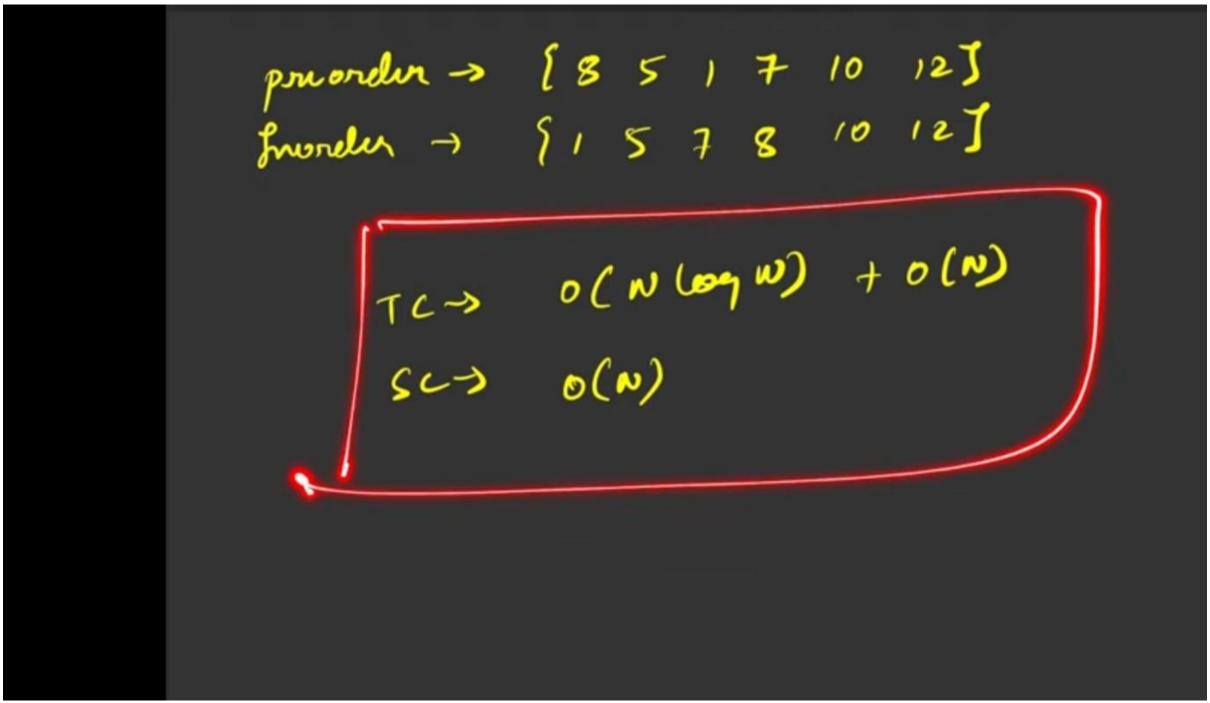


preorder → [8 5 1 7 10 12]



preorder → [8 5 1 7 10 12]

Check if a BT → BST

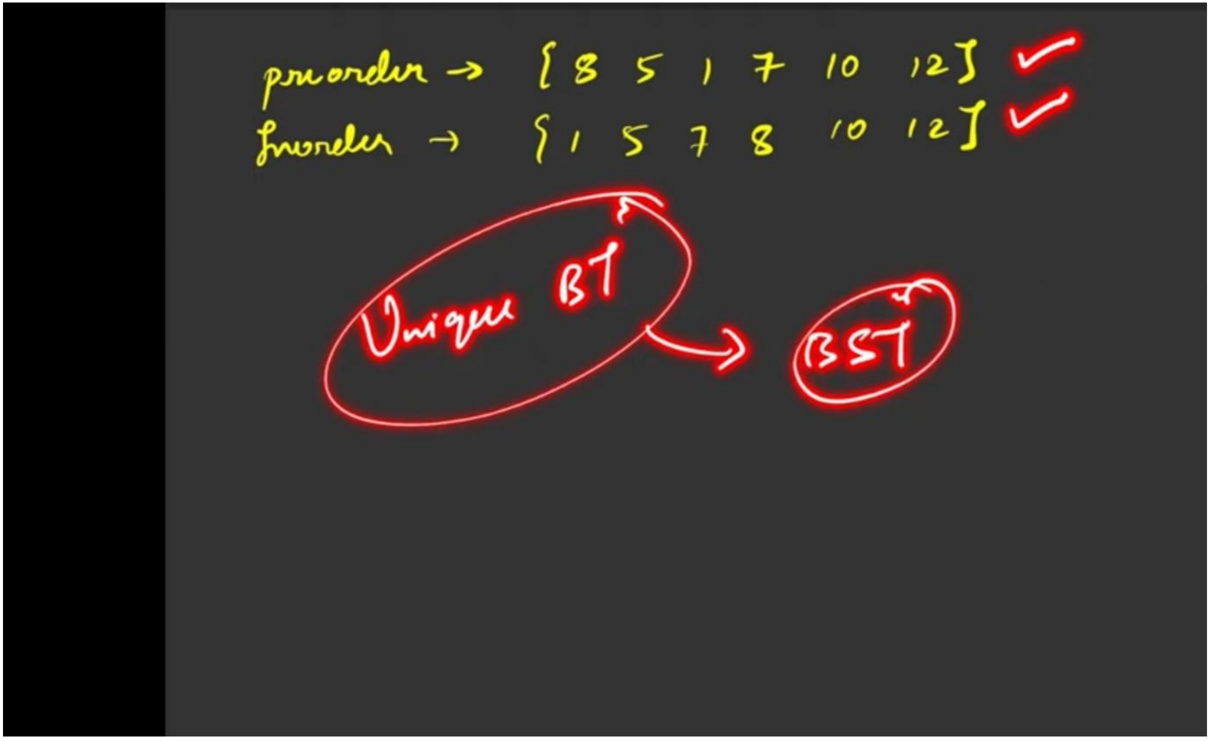


preorder → [8 5 1 7 10 12]

Inorder → [1 5 7 8 10 12]

TC → $O(N \log N) + O(N)$

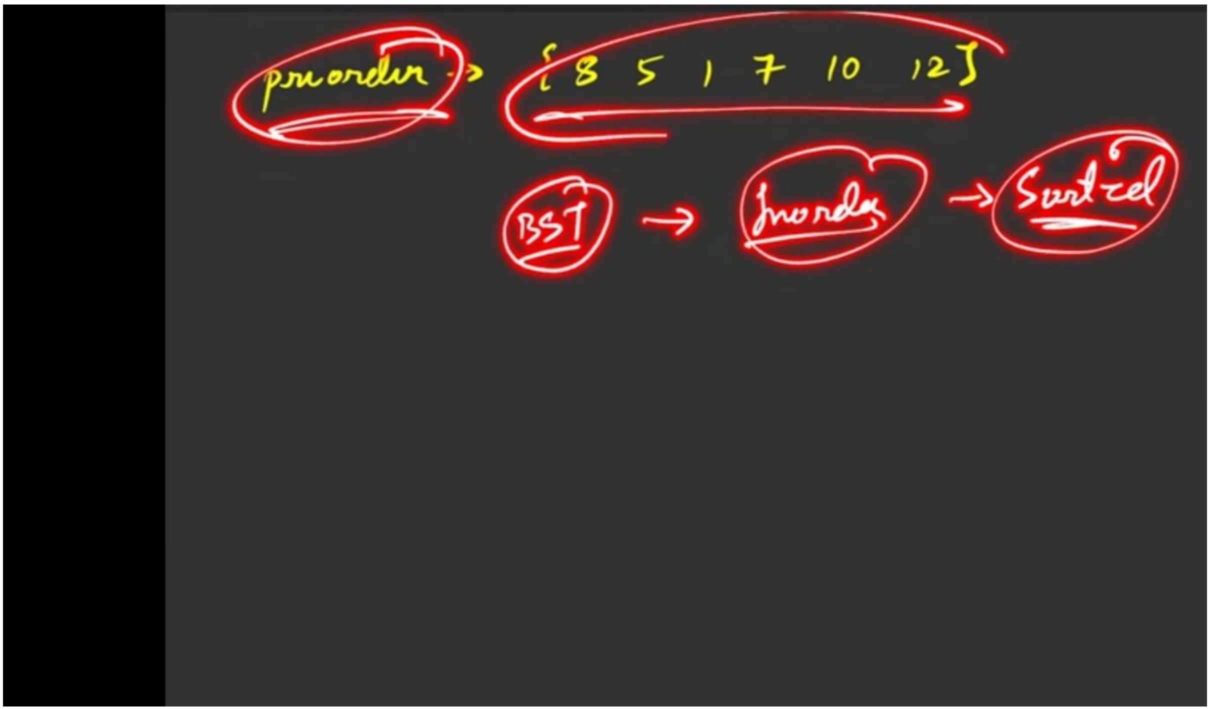
SC → $O(N)$



preorder → [8 5 1 7 10 12] ✓

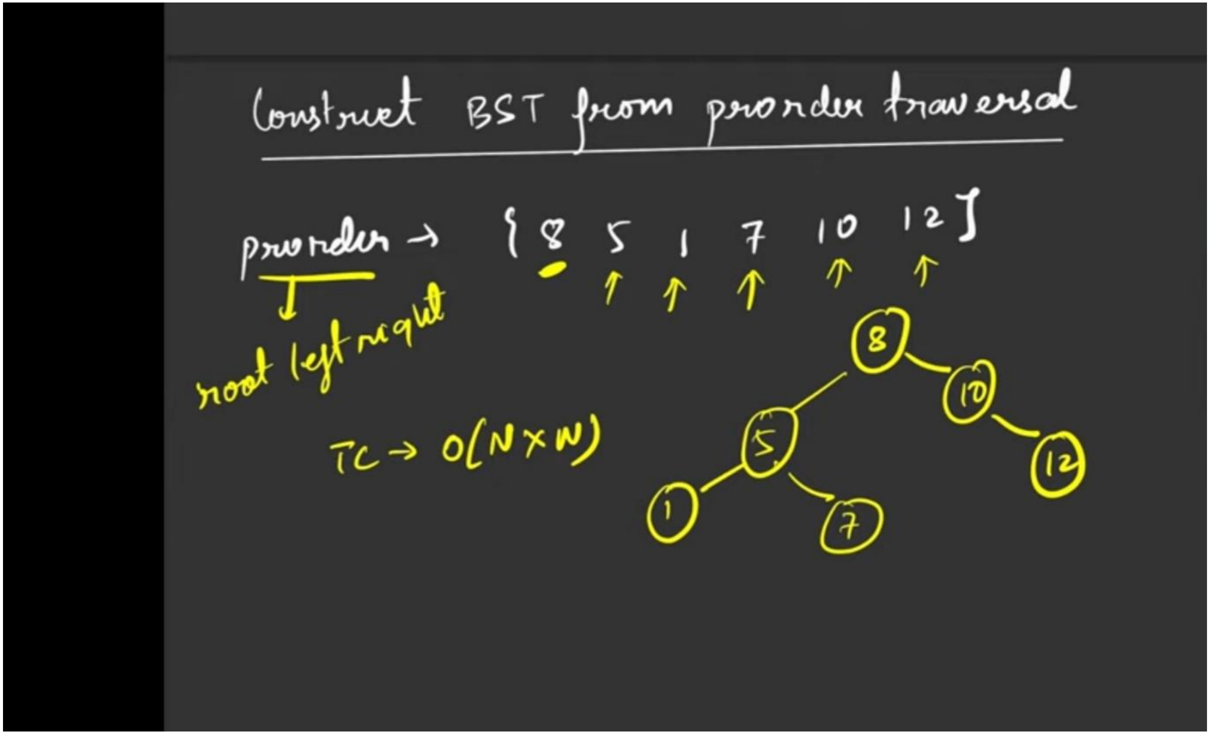
Inorder → [1 5 7 8 10 12] ✓

Unique BT → BST



preorder → [8 5 1 7 10 12]

BST → Inorder → Sorted

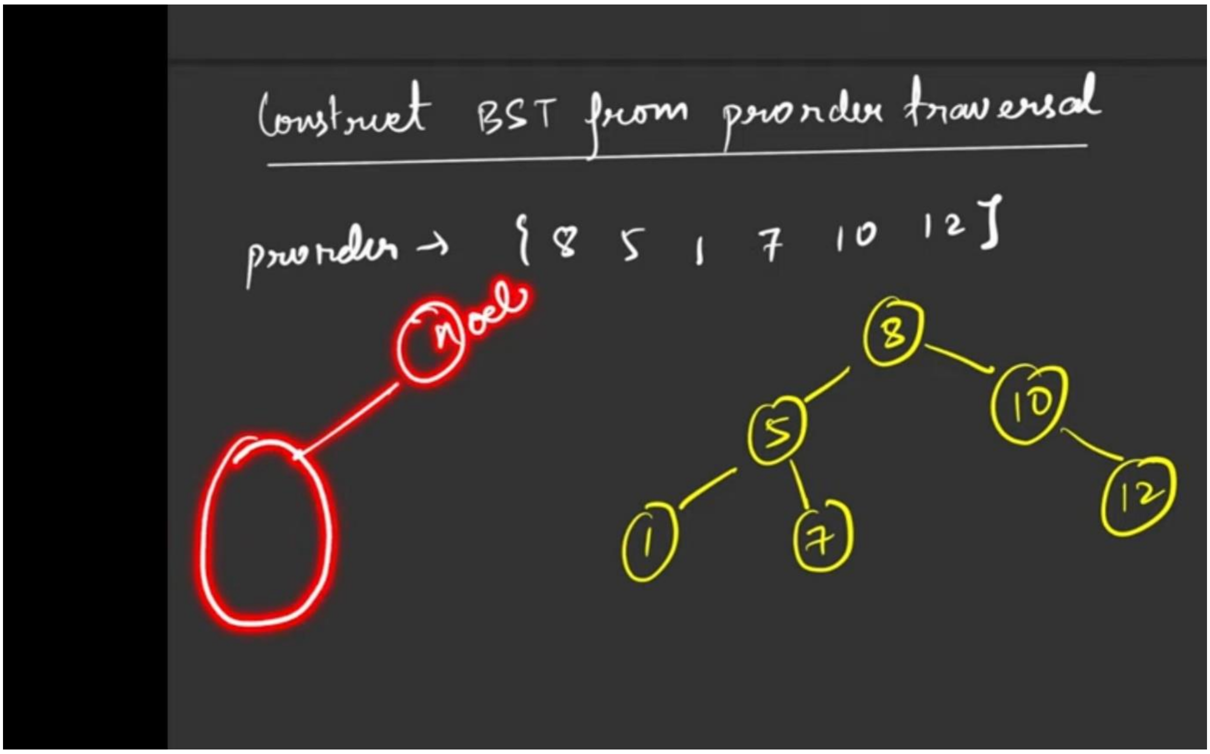


Construct BST from preorder traversal

preorder → [8 5 1 7 10 12]

root left right

TC → O(N×N)



Construct BST from preorder traversal

preorder → [8 5 1 7 10 12]