You are given n BST (binary search tree) root nodes for n separate BSTs stored in an array trees (0-indexed). Each BST in trees has at most 3 nodes, and no two roots have the same value. In one operation, you can:

Select two distinct indices i and j such that the value stored at one of the leaves of trees[i] is equal to the root value of trees[j].

Replace the leaf node in trees[i] with trees[j].

Remove trees[j] from trees.

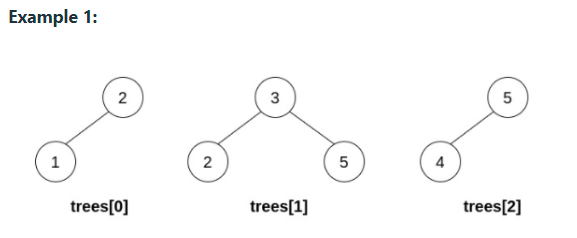
Return the root of the resulting BST if it is possible to form a valid BST after performing n - 1 operations, or null if it is impossible to create a valid BST.

A BST (binary search tree) is a binary tree where each node satisfies the following property:

Every node in the node's left subtree has a value strictly less than the node's value.

Every node in the node's right subtree has a value strictly greater than the node's value.

A leaf is a node that has no children.



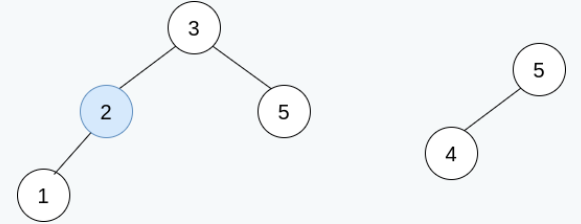
Input: trees = [[2,1],[3,2,5],[5,4]]

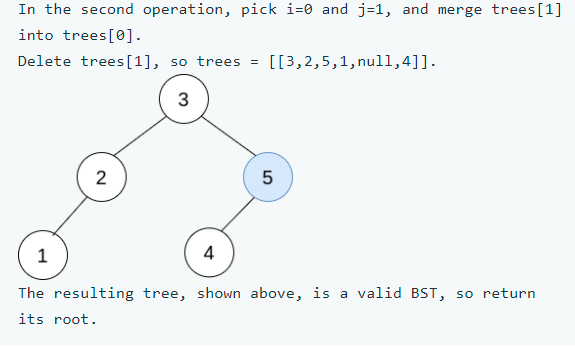
Output: [3,2,5,1,null,4]

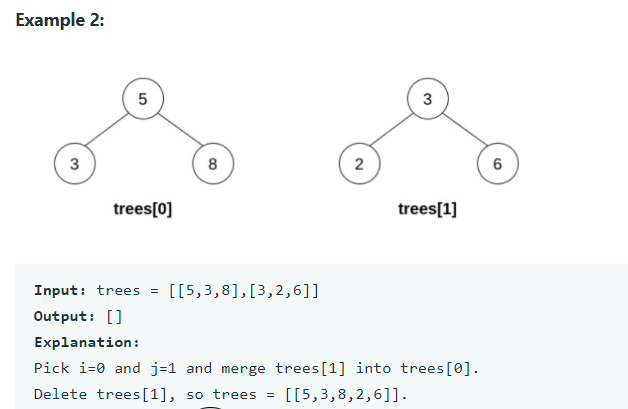
Explanation:

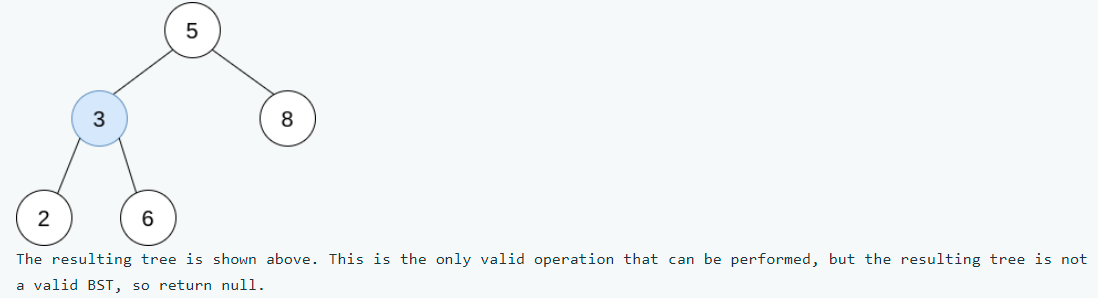
In the first operation, pick i=1 and j=0, and merge trees[0] into trees[1].

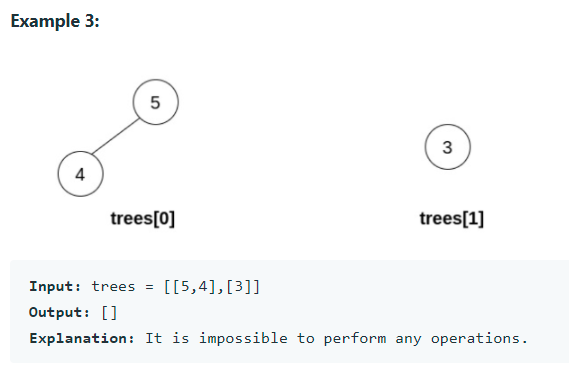
Delete trees[0], so trees = [[3,2,5,1],[5,4]].

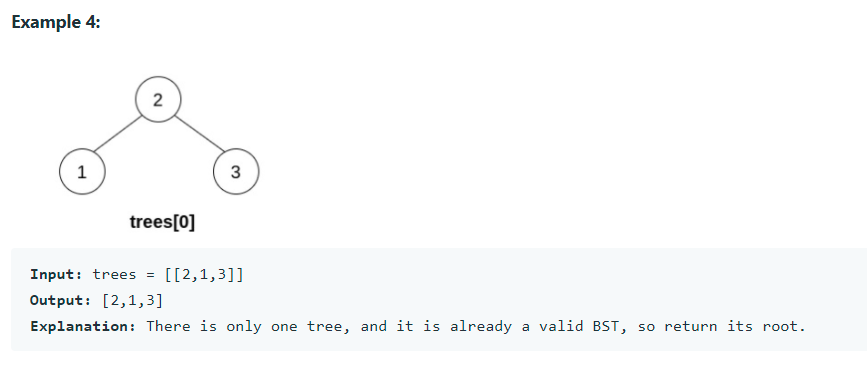












**Constraints:**

n == trees.length

1 <= n <= 5 \* 104

The number of nodes in each tree is in the range [1, 3].

Each node in the input may have children but no grandchildren.

No two roots of trees have the same value.

All the trees in the input are valid BSTs.

1 <= TreeNode.val <= 5 \* 104.