Construction Of Trees

If we are given two traversal sequences, can we construct the binary tree uniquely?

It depends on what traversals are given. If one of the traversal methods is inorder then the tree can be constructed uniquely, otherwise not.

Construction of a tree from Preorder traversal and inorder traversal

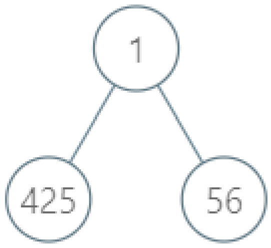
Let's understand it by an example

Preorder traversal: 1 2 4 5 3 6

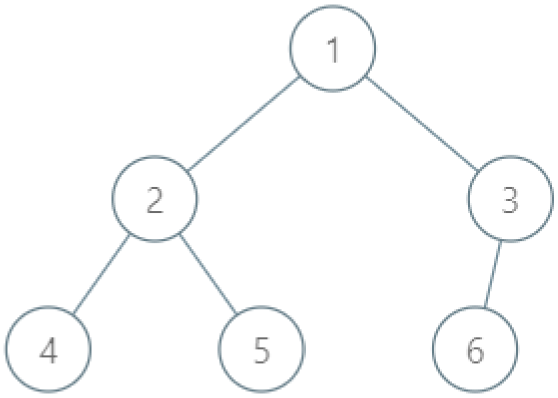
Inorder Traversal: 4 2 5 1 6 3

We know that in preOrder traversal root node comes first, therefore the root of the tree will be 1. Later on, searching 1 in Inorder traversal we will get to know the nodes in the left and the right subtree of the root node. So to optimize searching we must create a HashMap to keep track of numbers Vs indexes.

After the first step, the tree will look like



Repeating the same steps we will get the tree as shown below



Algorithm

constructTree()

- Initialize preIndex = 0 to iterate over the preorder traversal.
- Pick an element from preOrder traversal and increment the preIndex to pick the element in the next recursive call.
- Create a newNode and set newNode's data as the picked element.
- Find the picked element in inorder traversal and let the index be idx.
- Call constructTree for elements before idx and make the constructTree as a left subtree of newNode.
- Call constructTree for elements after idx and make the constructTree as the right subtree of newNode.
- return newNode.

Construction of a tree from PostOrder traversal and Inorder traversal

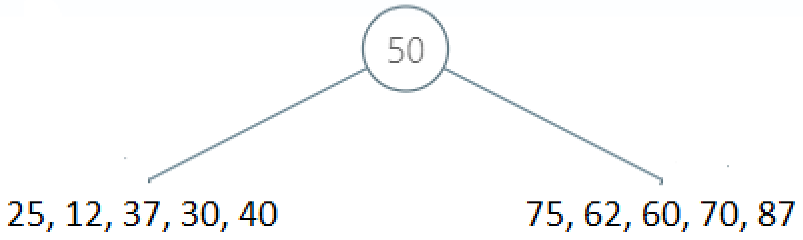
Given traversals:

Postorder: 12, 30, 40, 37, 25, 60, 70, 62, 87, 75, 50

Inorder: 12, 25, 30, 37, 40, 50, 60, 62, 70, 75, 87

- The last element in the postorder traversal is the root of the tree.
- So, here 50 will be the root of the tree.
- We will find the index of 50 in the inorder traversal. The index found is 5. Let this index be denoted by 'pos'.
- All the elements to the left of this index (from 0 to 4 index) will be in the left subtree of 50.
- And all the elements to the right of this index (from 6 to 10) will be in the right subtree of 50.

Now the structure of the tree is:



Now, we will divide the postorder and inorder array into two parts. One is for the left subtree and the other is for the right subtree.

Let **psi**: starting index for the preorder array

pei: ending index for the preorder array

isi: starting index of the inorder array

iei: ending index of the preorder array

clc: Number of elements in the left subtree

Clearly, clc = pos - isi;

For left subtree:

Postorder array: from index psi, psi + clc - 1

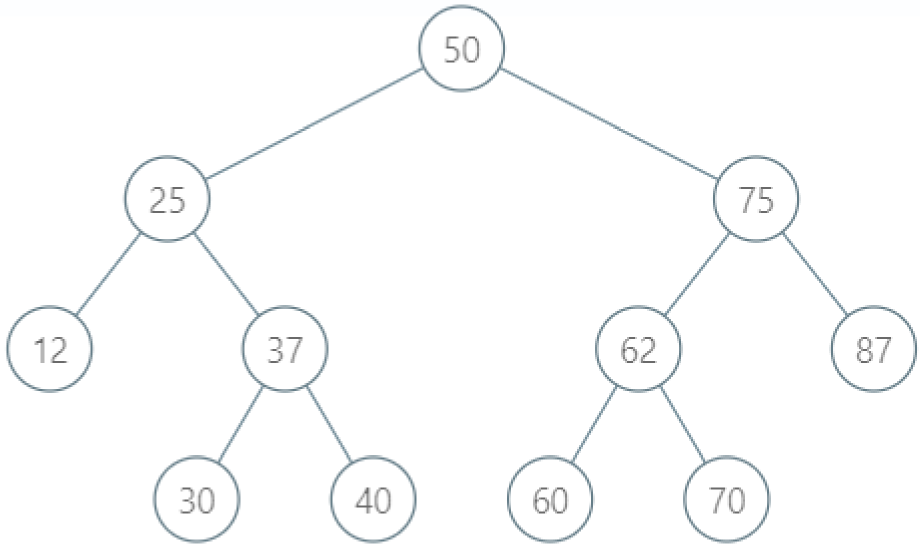
Inorder array: from index isi, isi + clc -1

For right subtree:

Postorder array: from index psi+clc, pei - 1

Inorder array: from index isi + clc + 1, iei

Using the above arrays, all the steps are recursively repeated. **The following binary tree is obtained:**


50
25
75
12
37
62
87
30
40
60
70