105 Construct Binary Tree From Preorder and Inorder traversals

algorithm :

(i) select preorder[0] as root. Erase preorder[0].

(ii) find preorder[0] in inorder. The left and right elements in inorder are the left and right subtree.

(iii) Recursively follow the procedure.

(iv) Base case occurs when inorder.size() == 0.

→ In the solution below, I have used find function to search for preorder[0] in inorder. A better way would be to use a hashmap to store indices of elements in inorder beforehand.

→ I have implemented another function called build because I need to pass inorder without reference.

Time complexity → $O(n^2)$ with this method.
→ $O(n)$ with hash map

space complexity → $O(n^2)$ very bad. for this solution. This solutin is infact very bad.

CODE

```
TreeNode* buildTree (vector <int> &preorder , vector <int> &
                                                inorder ) {

        TreeNode* root ;
        root  = build ( preorder, inorder ) ;
        return root;
    }

TreeNode*  build ( vector <int> &preorder,  vector <int> inorder ){

        TreeNode* root = new TreeNode ();
        if ( inorder . size () == 0 ) {
                root = NULL;
                return root;
            }
        auto it = find ( inorder. begin (), inorder. end (),
                                            preorder [0] ) ;
        int pos = it - inorder. begin () ;

        root → val = preorder [0] ;
        preorder . erase ( preorder . begin ()) ;

        vector <int> in_left;
        in_left. assign ( inorder . begin (), inorder. begin () + pos );
        vector <int> in_right;
        in_right . assign ( inorder. begin () + pos + 1, inorder. end ());

        root → left = build (preorder , in_left );
        root → right = build ( preorder, in_right );
        return root;
    }
```