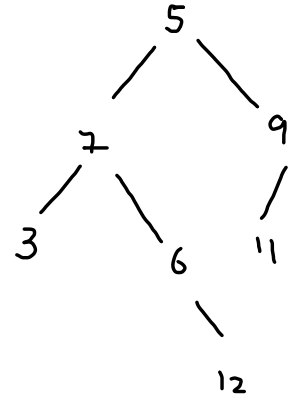
105. Construct Binary Tree from Preorder and Inorder Traversal

ps, pe, is, ie

	ps						pe
pre:	5	7	3	6	12	9	11
	0	1	2	3	4	5	6
in:	3	7	6	12	5	11	9
	is				idx		ie

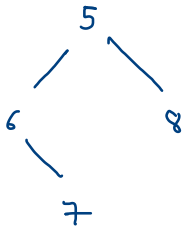
pre: N L R
in: L N R



	left	right
pre	(ps+1, ps+colse)	(ps+colse+1, pe)
in	(is, idx-1)	(idx+1, ie)

colse = idx - is

count of node's left subtree ele.



pre : 5 6 7 8

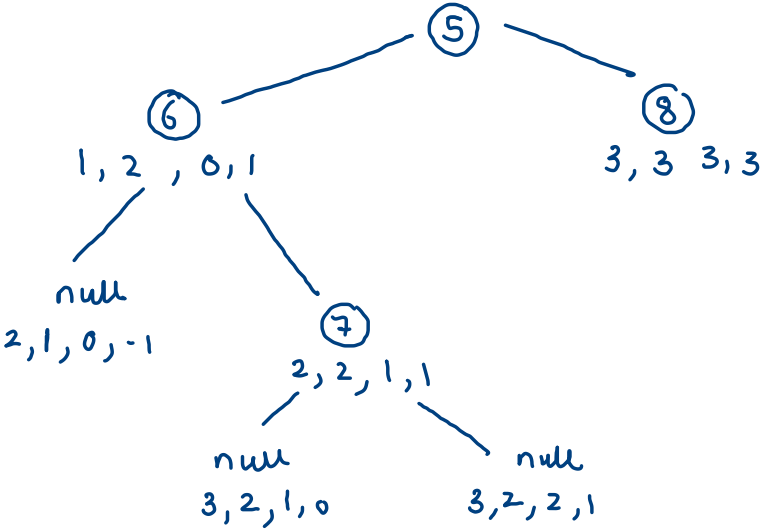
0 1 2 3

in : 6 7 5 8

	left	right
pre	(ps+1, ps+colse)	(ps+colse+1, pe)
in	(is, idx-1)	(idx+1, ie)

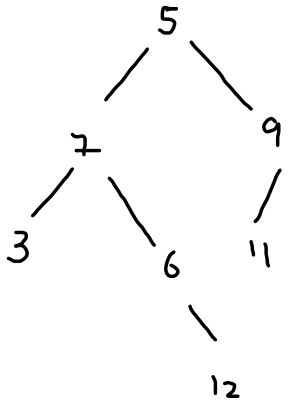
ps, pe, is, ie

0, 3, 0, 3



106. Construct Binary Tree from Inorder and Postorder Traversal

post : L R N
in : L N R



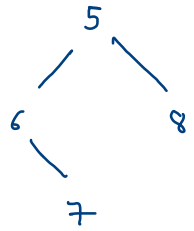
ps pe

post : 3 12 6 7 11 9 5 (5)
0 1 2 3 4 5 6

in : 3 7 6 12 5 11 9
is idx ie

colse = idx - is

	left	right
post	(ps, ps+colse-1)	(ps+colse, pe-1)
inorder	(is, idx-1)	(idx+1, ie)

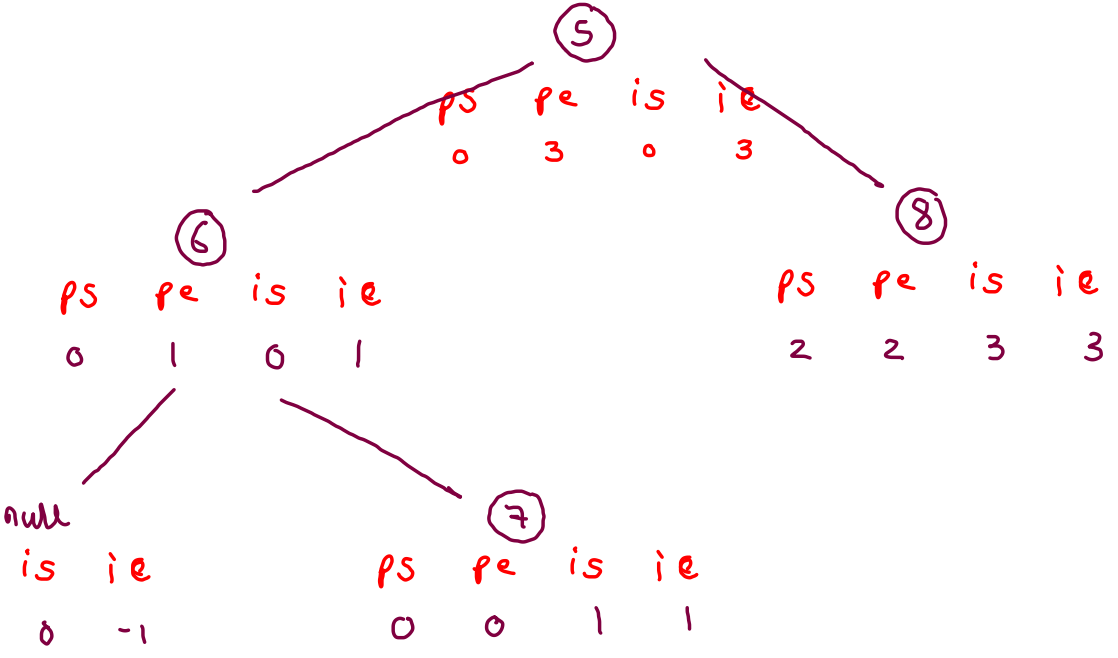


post : 7 6 8 5

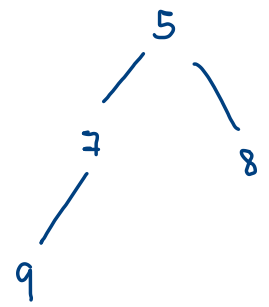
0 1 2 3

in : 6 7 5 8

```
if(ps > pe) {
    return null;
}

TreeNode node = new TreeNode(postorder[pe]);

int idx = -1;
for(int i=is; i <= ie;i++) {
    if(inorder[i] == node.val) {
        idx = i;
        break;
    }
}
int colse = idx-is;

node.left = helper(postorder,inorder,ps,ps+colse-1,is,idx-1);
node.right = helper(postorder,inorder,ps+colse,pe-1,idx+1,ie);

return node;
```

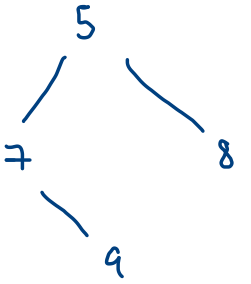


889. Construct Binary Tree from Preorder and Postorder Traversal



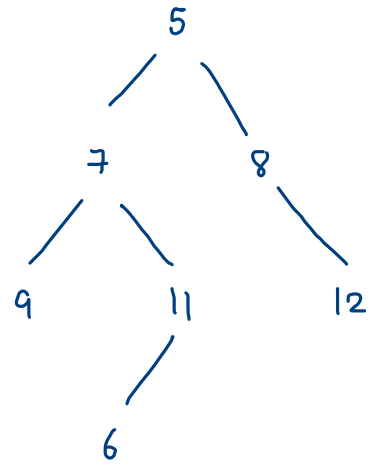
pre : 5 7 9 8

post : 9 7 8 5



pre : 5 7 9 8

post : 9 7 8 5



	prs						pre
pre:	5	7	9	11	6	8	12
	0	1	2	3	4	5	6
post:	9	6	11	7	12	8	5
	pos			idx			poe

pre: N L R

post: L R N

in: L N R

idx → index of preorder[prs+1] in postorder array.

	left	right
pre	(prs+1, prs+colse)	(prs+colse+1, pre)
post	(pos, idx)	(idx+1, poe-1)

colse = idx - pos + 1

```
if(prs > pre) {
    return null;
}
if(prs == pre) {
    return new TreeNode(preorder[prs]);
}

TreeNode node = new TreeNode(preorder[prs]);

int val = preorder[prs+1];
int idx = -1;

for(int i=pos;i <= poe;i++) {
    if(postorder[i] == val) {
        idx = i;
        break;
    }
}

int colse = idx - pos + 1;

node.left = helper(preorder,postorder,prs+1,prs + colse,pos,idx);
node.right = helper(preorder,postorder,prs + colse + 1,pre,idx+1, poe-1);

return node;
```