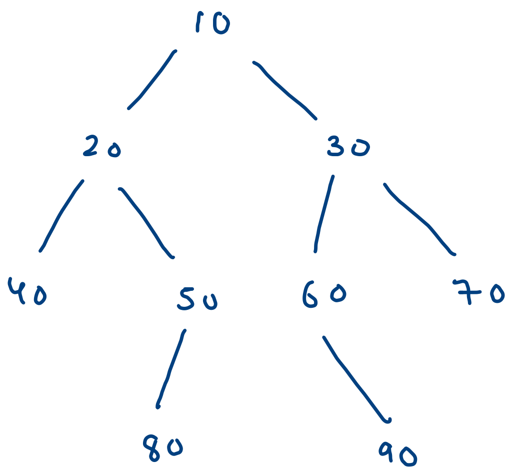
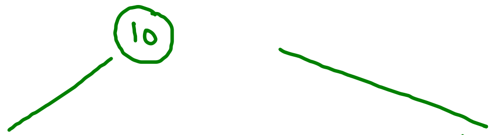
Construct Binary Tree From Inorder And Levelorder Traversal



level :	10	20	30	40	50	60	70	80	90
	0	1	2	3	4	5	6	7	8
in :	40	20	80	50	10	60	90	30	70
	is				idx				ie



inorder : 40 20 80 50

lo : 20 40 50 80

inorder : 60 90 30 70

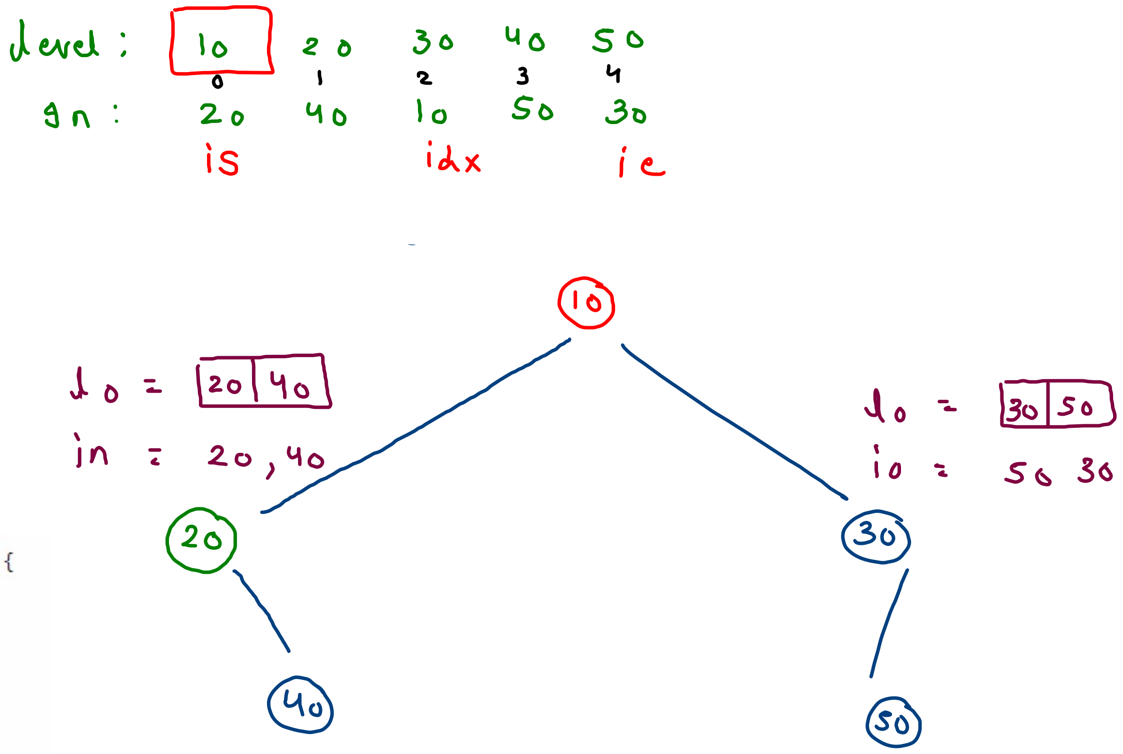
lo : 30 60 70 90

inorder : virtually shorten

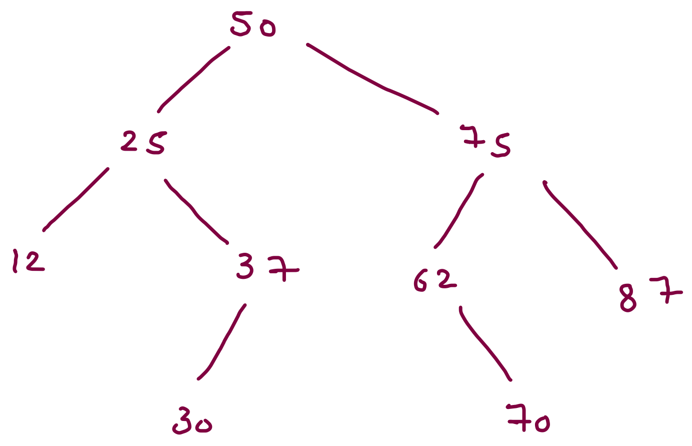
levelorder: reality shorten

(left and right sub-tree element are not in continous)

```java
Node helper(int[]inorder,int[]level,int is,int ie) {
    if(is > ie) {
        return null;
    }

    Node node = new Node(level[0]);
    int idx = map.get(node.data);
    int colse = idx - is;
    int corse = ie - idx;

    int[]llo = new int[colse];
    int[]rlo = new int[corse];

    segregateLevelOrder(llo,rlo,level,idx);

    node.left = helper(inorder,llo,is,idx-1);
    node.right = helper(inorder,rlo,idx+1,ie);

    return node;
}

void segregateLevelOrder(int[]llo,int[]rlo,int[]level,int idx) {
    int j = 0;
    int k = 0;

    for(int i=1; i < level.length;i++) {
        if(map.get(level[i]) < idx) {
            //belongs to left-substree
            llo[j++] = level[i];
        }
        else {
            //belongs to right-subtree
            rlo[k++] = level[i];
        }
    }
}
```

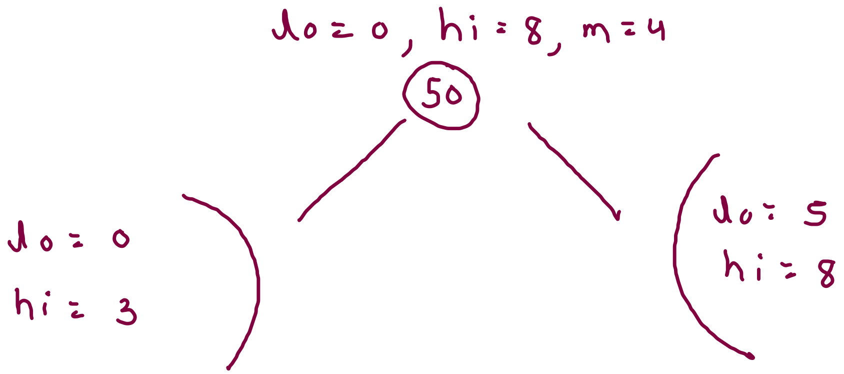


Construct Bst From Inorder Traversal

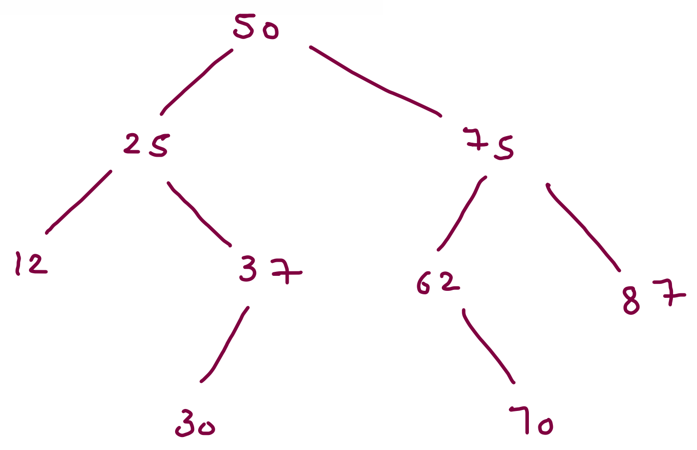


inorder : 12 25 30 37 50 62 70 75 87

12_0	25_1	30_2	37_3	50_4	62_5	70_6	75_7	87_8
lo				mid				hi



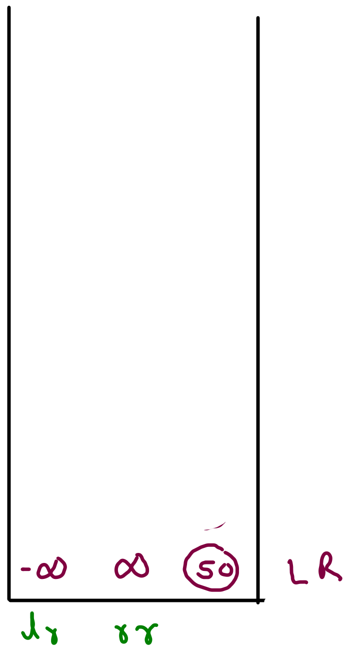
1008. Construct Binary Search Tree from Preorder Traversal

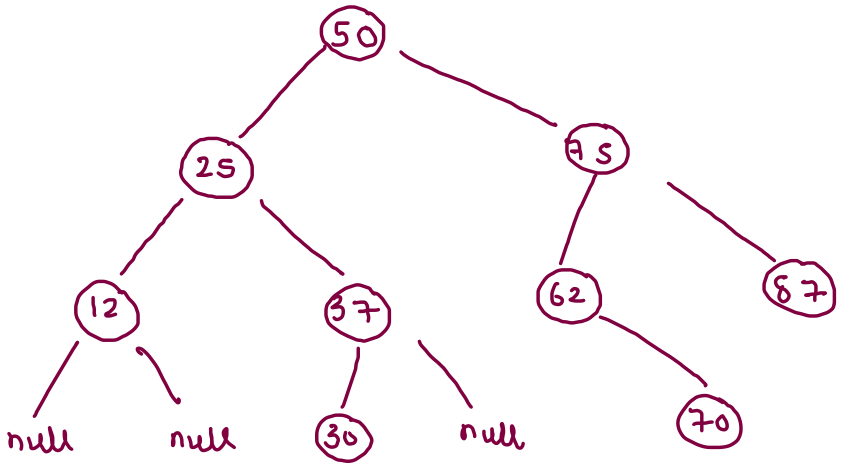


pre : 50 25 12 37 30 75 62 70 87

i

pre : 50 25 12 37 30 75 62 70 87

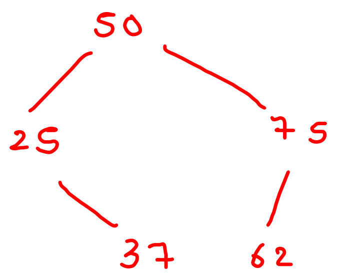




$$\begin{tikzcd}
lr,\ pre[i] & & pre[i],\ rr \\
& lr,\ rr \arrow[ul, dash, "left"'] \arrow[ur, dash, "right"] &
\end{tikzcd}$$

$lr < pre[i] < rr$

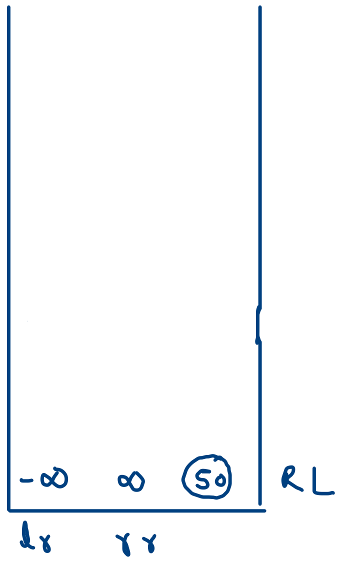
Construct BST from Postorder

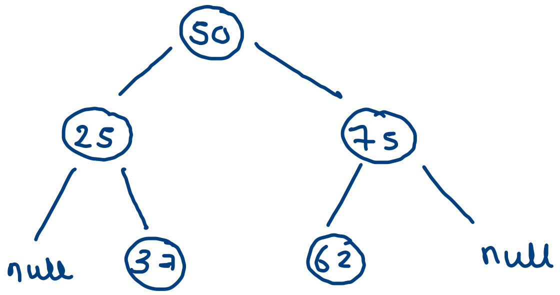


right call
left call

i

post : 37 25 62 75 50




50
25
75
null
37
62
null