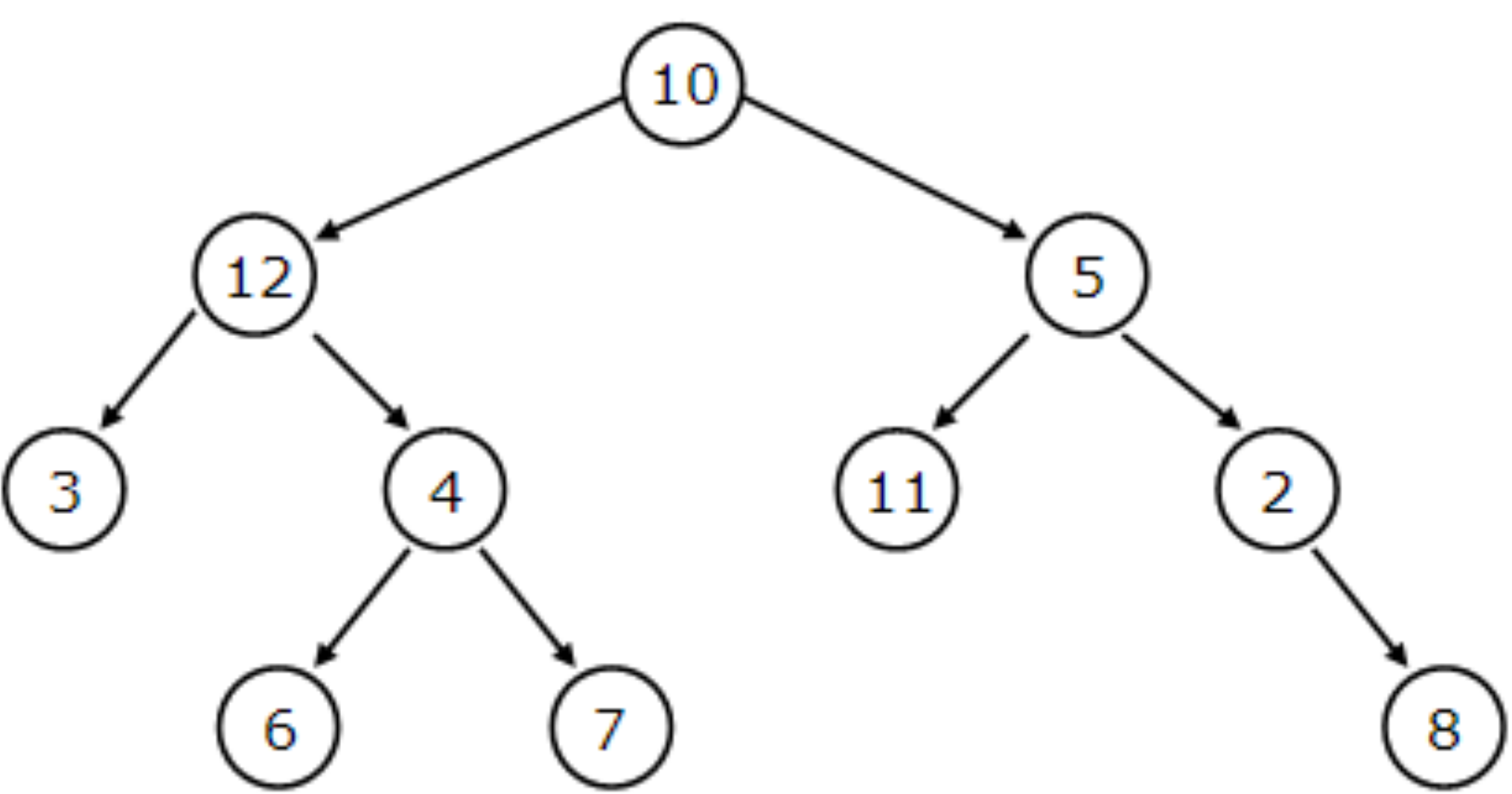
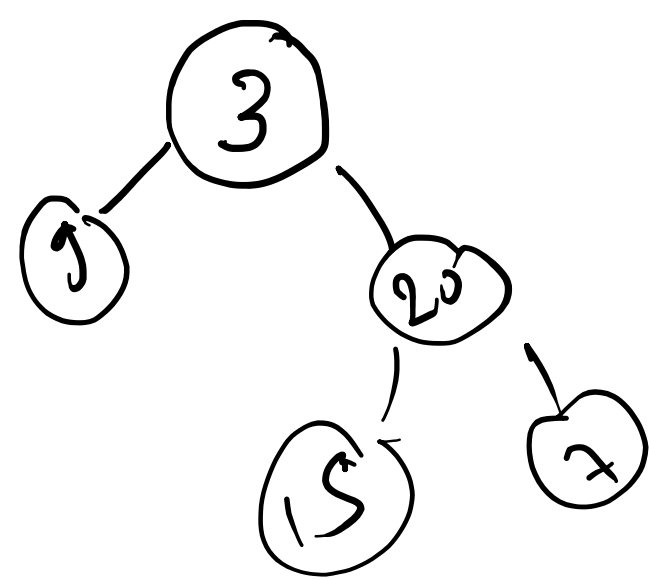
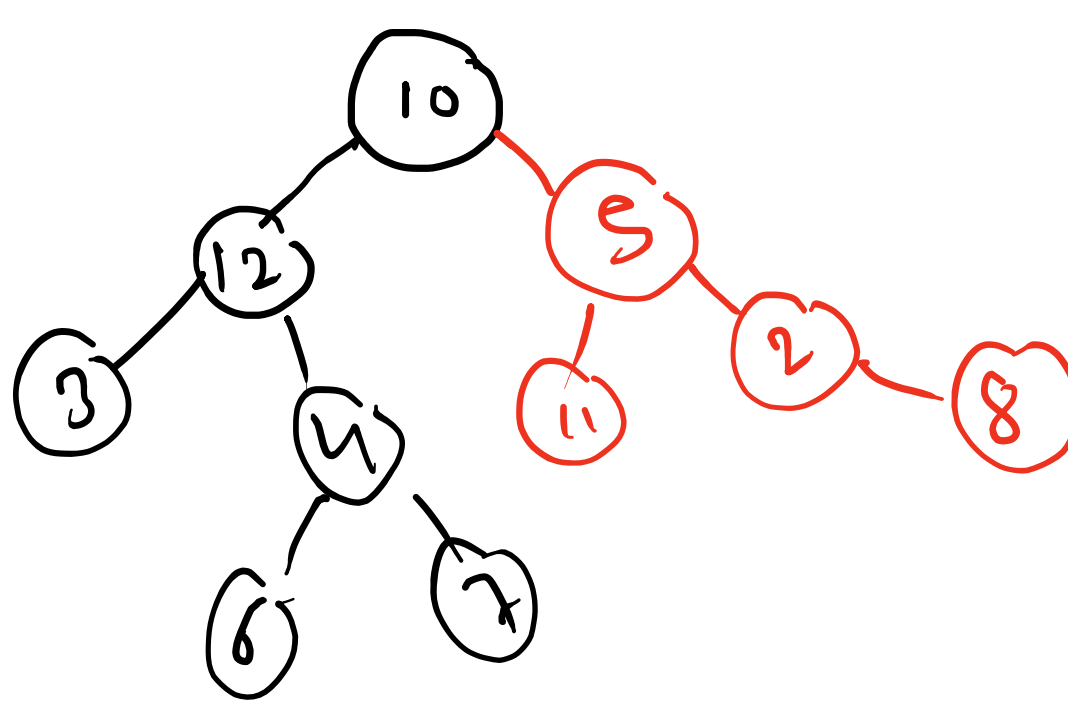


Input: preorder = [3, 9, 20, 15, 7], inorder = [9, 3, 15, 20, 7]



Levelorder tree traversal
10, 12, 5, 3, 4, 11, 2, 6, 7, 8
Inorder tree traversal
3, 12, 6, 4, 7, 10, 11, 5, 2, 8
Preorder tree traversal
10, 12, 3, 4, 6, 7, 5, 11, 2, 8
Postorder tree traversal
3, 6, 7, 4, 12, 11, 8, 2, 5, 10

in [3, 12, 6, 4, 7]
pre [12, 3, 4, 6, 7]

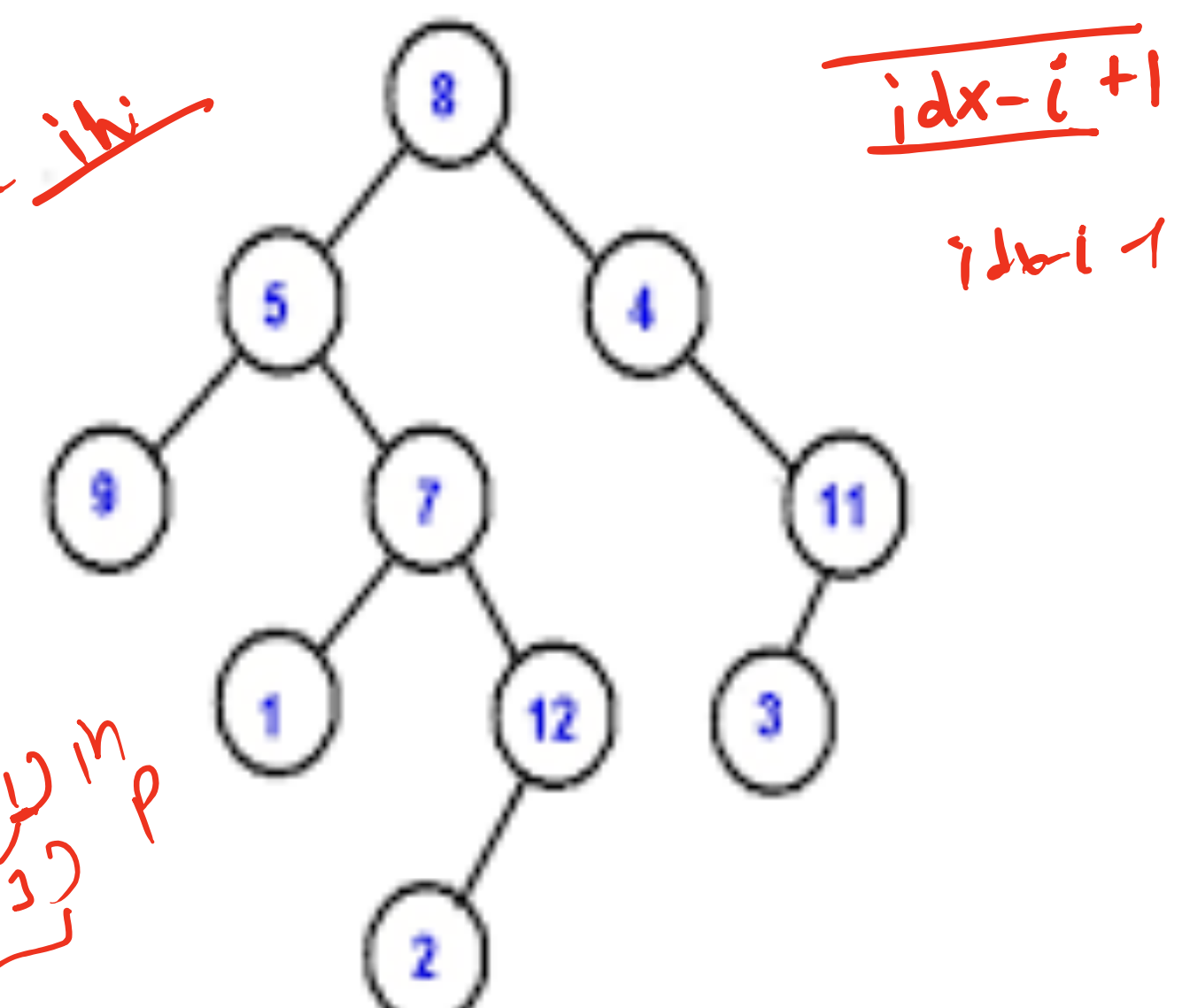
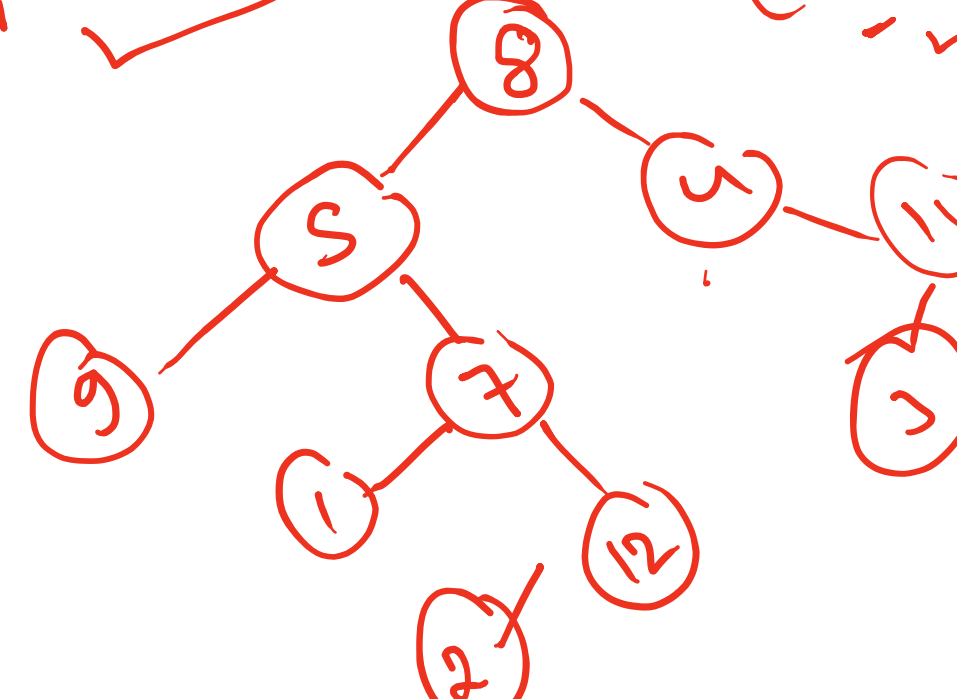


Plat, plo, phi
plo, phi

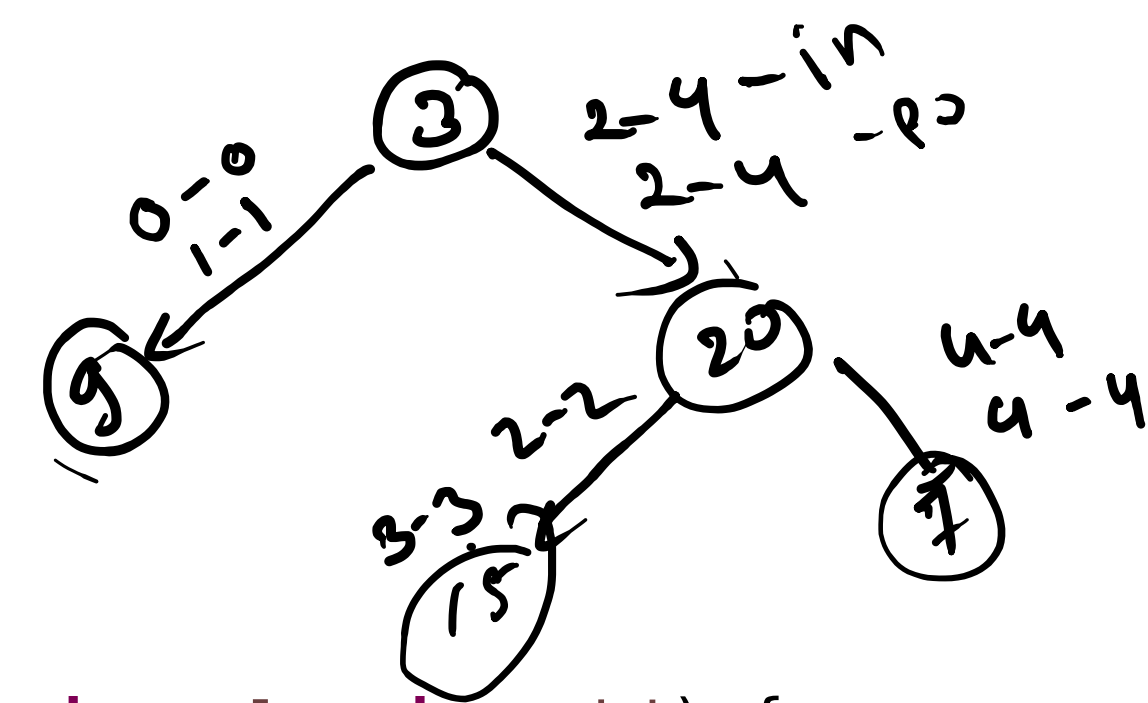
[11, 5, 2, 8]
[5, 11, 2, 8]

PreOrder - 8, 5, 9, 7, 1, 11, 2, 4, 11, 3
InOrder - 9, 5, 1, 7, 2, 12, 8, 4, 3, 11
PostOrder - 9, 1, 2, 12, 7, 5, 3, 11, 4, 8
LevelOrder - 8, 5, 4, 9, 7, 11, 1, 12, 3, 2

in [3, 5, 1, 7, 2, 12]
pre [5, 9, 7, 11, 2, 8]



preorder = [3, 9, 20, 15, 7],
inorder = [9, 3, 15, 20, 7]



```
public TreeNode build(int[] pre, int[] in, int ilo, int ihi, int plo, int phi) {  
    if (ilo > ihi || plo > phi) {  
        return null;  
    }  
    TreeNode node = new TreeNode(pre[plo]);  
    int idx = search(in, ilo, ihi, pre[plo]);  
    int c = idx - ilo; 1-0=1  
    node.left = build(pre, in, ilo, idx - 1, plo + 1, plo + c);  
    node.right = build(pre, in, idx + 1, ihi, plo + c + 1, phi);  
    return node;  
}
```

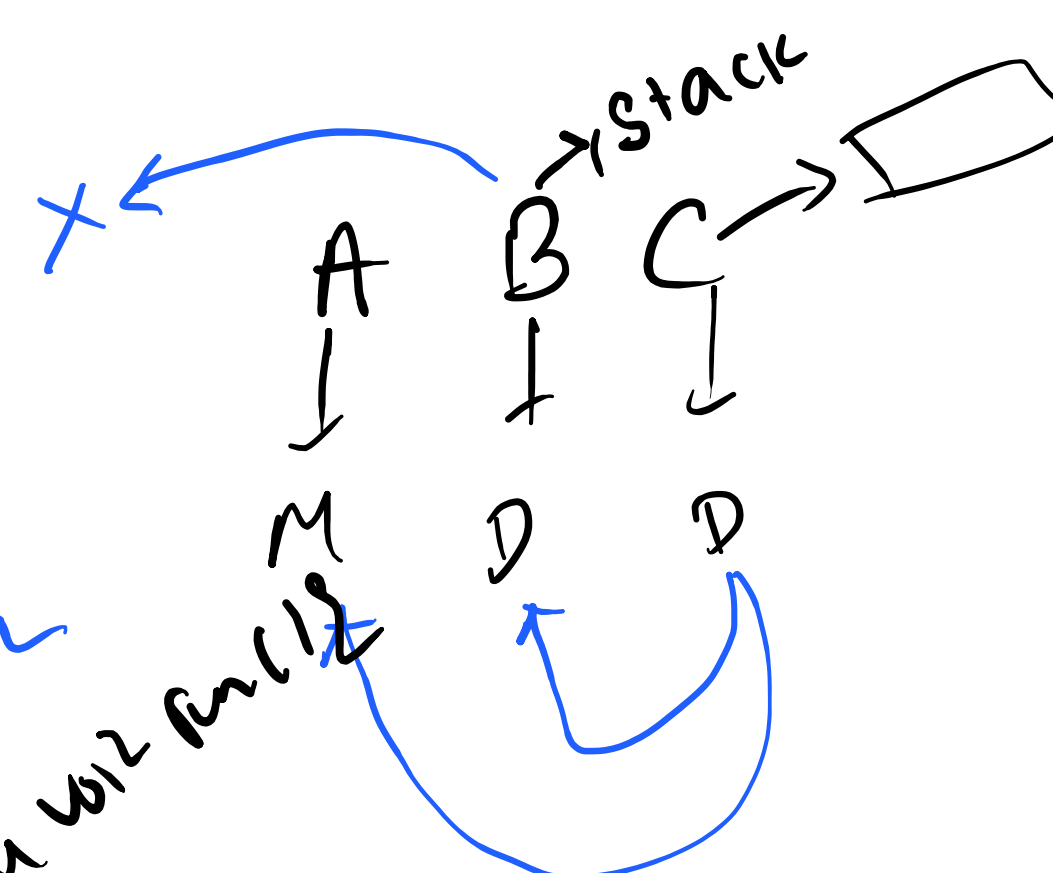
2-2=0

Subtree method
ans tree class

Variable

longest

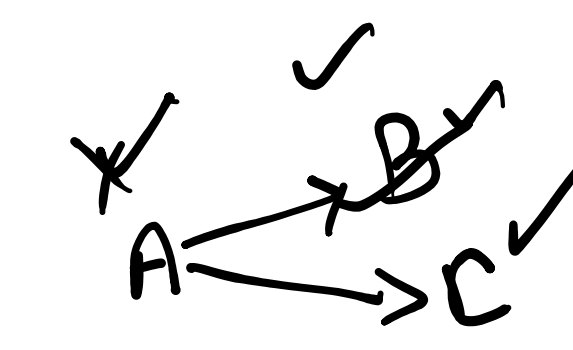
class Solution {
 public int minPathSum(TreeNode root) {
 // ...
 }
}



void push(int item) {

void pop() {

int peek() {



New AC
obj. func()