/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode(int x) { val = x; }

\* }

\*/

class Solution {

Set<Integer> res = new HashSet<>();

public void burn(TreeNode root,TreeNode block,int time,List<List<Integer>> ans){

if(root == null || root == block || res.contains(root.val)) return;

if(time == ans.size()){

ans.add(new ArrayList<>());

}

ans.get(time).add(root.val);

burn(root.left,block,time+1,ans);

burn(root.right,block,time+1,ans);

}

public int fire(TreeNode root,List<List<Integer>> ans,TreeNode firenode){

if(root == null) return -1;

if(root == firenode){

if(res.contains(root.val)){

return -2;

}

burn(firenode,null,0,ans);

return 1;

}

int lf = fire(root.left,ans,firenode);

if(lf == -2) return -2;

if(lf>0){

if(res.contains(root.val)){

return -2;

}

burn(root,root.left,lf,ans);

return lf + 1;

}

int rt = fire(root.right,ans,firenode);

if(rt == -2) return -2;

if(rt>0){

if(res.contains(root.val)){

return -2;

}

burn(root,root.right,rt,ans);

return rt + 1;

}

return -1;

}

public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {

List<List<Integer>> ans = new ArrayList<>();

res.add(2);

res.add(8);

fire(root,ans,p);

// HashSet<Integer> set = new HashSet<>();

System.out.print(ans);

return p;

}

}