

class Solution {

public int maxDepth(TreeNode root) {

if (root == null) {

return 0;

} else {

int leftHeight = maxDepth(root.left);

int rightHeight = maxDepth(root.right);

return Math.max(leftHeight, rightHeight) + 1;

}

}

}



class Solution {

    public int mySqrt(int x) {

       int r=x;

       int l=0;

       int result=0;

       int mid = (r+l)/2;

       while(l<=r){

           mid =  l+(r-l)/2;

           if((long)mid\*mid<=x){

               result = mid;

               l = mid+1 ;

           }

           else if((long)mid\*mid>x){

               r = mid -1;

           }

       }

       return result;

    }

}



class Solution {

    public int missingNumber(int[] nums) {

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

        int n = nums.length;

        int missIndex = -1;

        for(int i = 0; i<n; i++){

            set.add(nums[i]);

        }

        for(int i=0;i<=n;i++){

            if(!set.contains(i)){

                missIndex = i;

                break;

            }

        }

        return missIndex;

    }

}







