**Problem Statement**

Given an array of size N, we need to determine whether it is possible to rearrange the elements in such a way that they can form a valid Binary Search Tree (BST). We are allowed to perform swaps between any two elements in the array.

Our task is to check if it is possible to rearrange the elements of the given array in such a way that they satisfy the properties of a BST.

Note: We are only allowed to perform swaps between array elements, and not any other operations like inserting or deleting nodes from a BST.

Input Format

The first line contains an integer N, representing the size of the array,

The second line contains N space-separated integers, representing the array arr.

Output Format

Return “1” if it’s possible to construct BST from the given array. Else return “0”.

Constraints

1 ≤  N ≤10^5

1 ≤  arr[i] ≤10^5

Sample Testcase 0

Testcase Input

5 1 8 27 2 3

Testcase Output

1

Explanation

All numbers are distinct, so we can create a BST by sorting the array.

Sample Testcase 1

Testcase Input

5 2 3 3 5 6

Testcase Output

0

Explanation

The number 3 is in the array 2 times, and we know that a BST doesn't contain any number multiple times.

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

class Main {

    public static void main(String[] args) {

        /\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();

        int[] arr=new int[n];

        for(int i=0;i<arr.length;i++){

            arr[i]=sc.nextInt();

        }

        Arrays.sort(arr);

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

        for(int i:arr){

            hset.add(i);

        }

        if(hset.size()==arr.length){

            System.out.println(1);

        }else{

            System.out.println(0);

        }

    }

}