**Fitting The Array**

Submissions: [3293](https://practice.geeksforgeeks.org/problem_submissions.php?pid=3287)  Accuracy:

44.77%

   Difficulty: [Easy](https://practice.geeksforgeeks.org/Easy/0/0/)   Marks: 2

Show Topic Tags   

Adobe is playing an array game. He is weak in the concepts of arrays. Adobe is given two arrays a[ ] and b[ ] of the same size. The array a[ ] will be said to fit in array b[ ] if by arranging the elements of both array, there exists a solution such that i\_th element of a[ ] is less than or equal to an i\_th element of b[ ]. Help Adobe find if the given arrays are fit or not.

**Input:**  
The first line of input contains an integer **T**denoting the number of test cases. For each test case, the next subsequent line contains the integer **N** i.e. size of arrays followed by array a[ ] and then array b[ ].

**Output:**  
Print "YES" if array a[ ] fit in array b[ ] otherwise print "NO".

**Constraints:**  
1<= T<= 100  
1<= N<= 100  
0<=a[ i ]<=1000  
0<= b[ i ]<=1000

**Example:**  
**Input**  
2  
4  
7 5 3 2   
5 4 8 7  
8  
7 5 3 2 5 105 45 10  
2 4 0 5 6 9 75 84  
**Output**  
YES  
NO

\*\* For More Input/Output Examples Use ['Expected Output'](https://practice.geeksforgeeks.org/problems/fitting-the-array/0#ExpectOP) option \*\*

Contributor: Siddhant Manglik  
[Author: adobe\_3128](https://auth.geeksforgeeks.org/user/adobe_3128/practice/)

<https://practice.geeksforgeeks.org/problems/fitting-the-array/0>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static int[] countsort(int[] arr)

{

int n = arr.Length;

// The output character array that

// will have sorted arr

int[] output = new int[n];

// Create a count array to store

// count of inidividul characters

// and initialize count array as 0

int[] count = new int[1001];

for (int i = 0; i < 1001; ++i)

count[i] = 0;

// store count of each character

for (int i = 0; i < n; ++i)

++count[arr[i]];

// Change count[i] so that count[i]

// now contains actual position of

// this character in output array

for (int i = 1; i <= 1000; ++i)

count[i] += count[i - 1];

// Build the output character array

// To make it stable we are operating in reverse order.

for (int i = n - 1; i >= 0; i--)

{

output[count[arr[i]] - 1] = arr[i];

--count[arr[i]];

}

// Copy the output array to arr, so

// that arr now contains sorted

// characters

for (int i = 0; i < n; ++i)

arr[i] = output[i];

return arr;

}

static string Encaja(int[] A , int[] B)

{

if (A.Length > B.Length) return "NO";

int[] sa = countsort(A);

int[] sb = countsort(B);

for(int i =0; i<A.Length; i++) if (sa[i] > sb[i]) return "NO";

return "YES";

}

static void Main()

{

int T = int.Parse(Console.ReadLine().Trim());

while (T-- > 0)

{

int n = int.Parse(Console.ReadLine().Trim());

int[] a = Array.ConvertAll(Console.ReadLine().Trim().Split(' '), e => int.Parse(e));

int[] b = Array.ConvertAll(Console.ReadLine().Trim().Split(' '), e => int.Parse(e));

Console.WriteLine(Encaja(a, b));

}

Console.ReadLine();

}

}

}