How many strings equal to A can be constructed using letters from the string B? Each letter can be used only once and in one string only.

**Example**

For A = "abc" and B = "abccba", the output should be  
stringsConstruction(A, B) = 2.

We can construct 2 strings A with letters from B.

**Input/Output**

* **[time limit] 3000ms (cs)**
* **[input] string A**

String to construct, A contains only lowercase English letters.

*Constraints:*  
3 ≤ A.length ≤ 10.

* **[input] string B**

String containing needed letters, B contains only lowercase English letters.

*Constraints:*  
3 ≤ B.length ≤ 50.

* **[output] integer**

<https://codefights.com/arcade/code-arcade/mirror-lake/chW9F8bCgxYJBcgj3>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

//------ESTA FUNCION NO PASA TODOS LOS TEST ------------

//static int stringsConstruction(string A, string B)

//{

// Dictionary<char, int> fa = new Dictionary<char, int>();

// for (int i = 0; i < A.Length; i++)

// {

// if (fa.ContainsKey(A[i]))

// {

// fa[A[i]]++;

// }

// else

// {

// fa[A[i]] = 1;

// }

// }

// bool estaEnA = false;

// Dictionary<char, int> fb = new Dictionary<char, int>();

// for (int i = 0; i < B.Length; i++)

// {

// if (fa.ContainsKey(B[i]))

// {

// estaEnA = true;

// if (fb.ContainsKey(B[i]))

// {

// fb[B[i]]++;

// }

// else

// {

// fb[B[i]] = 1;

// }

// }

// }

// if (!estaEnA) return 0;

// int min = int.MaxValue;

// foreach (KeyValuePair<char, int> kvp in fa)

// {

// if (fb.ContainsKey(kvp.Key))

// {

// int div = fb[kvp.Key] / kvp.Value;

// min = Math.Min(div, min);

// }

// }

// return min;

//}

static int stringsConstruction(string A, string B)

{

/\*Si el tamaño de B es menor que el de A,

es porque B no contiene todos los elementos de A\*/

if (B.Length < A.Length)

return 0;

/\*

recorro A y me fijo que cada elemento este en B,

de lo contrario B no tiene todos los elementos de A

\*/

for (int i = 0; i < A.Length; i++)

if (!B.Contains(A[i]))

return 0;

/\*

Una vez que me aseguro de que B tiene todos los elementos de A

recorro B, y me fijo si el caracter actual es parte de A

de ser asi, cuento cuantas veces esta en B y lo divido por las

veces que esta en A

La division minima es el resultado del problema ya que

el minimo es el que tiene la menor frecuencia, porque por ejemplo

\* un caracter de A esta en B 500 veces ej 'j' (que puede estar varias veces en A

pero tomo de a grupos) y otro caracter ej 'b' esta 3 veces, entonces

el resultado es B, porque eso significa que 'j'

\* tambien esta AL MENOS 3 veces

\*/

int min = int.MaxValue;

foreach (char ch in B)

{

if (A.Contains(ch))

{

int div = B.Count(f => f == ch) / A.Count(f => f == ch);

min = Math.Min(min, div);

}

}

return min;

}

static void Main(string[] args)

{

//string A = "abc";

//string B = "abccba";

string A = "zzz";

string B = "zzzzzzzzzzz";

//string A = "abc";

//string B = "def";

Console.WriteLine(stringsConstruction(A, B));

//string a = "aaaabb";

//int cont = a.Count(f => f =='a');

//Console.WriteLine(cont);

Console.ReadLine();

}

}

}