**Simple Fun #30: Strings Construction**

6096% of 2715 of69[myjinxin2015](http://www.codewars.com/users/myjinxin2015)

C#

* [TRAIN AGAIN](http://www.codewars.com/kata/simple-fun-number-30-strings-construction/train/csharp)
* [NEXT KATA](http://www.codewars.com/trainer/csharp)

Details

[Solutions](http://www.codewars.com/kata/simple-fun-number-30-strings-construction/solutions/csharp)

[Discourse (2)](http://www.codewars.com/kata/simple-fun-number-30-strings-construction/discuss/csharp)

* Add to Collection
* |
* Share this kata:

**Task**

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 2.

We can construct 2 strings A with letters from B.

**Input/Output**

* [input] string A

String to construct, A contains only lowercase English letters.

Constraints: 3 ≤ A.length ≤ 9.

* [input] string B

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

Constraints: 3 ≤ B.length ≤ 50.

* [output] an integer

<http://www.codewars.com/kata/simple-fun-number-30-strings-construction/csharp>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication2

{

class Program

{

public static int StringsConstruction(string A, string B)

{

//coding and coding..

Dictionary<char, int> frecA = A.ToCharArray() .GroupBy(x => x)

.ToDictionary(x => x.Key, x => x.Count());

Dictionary<char, int> frecB = B.ToCharArray().GroupBy(x => x)

.ToDictionary(x => x.Key, x => x.Count());

int min = int.MaxValue;

foreach (KeyValuePair<char, int> kvp in frecA)

{

if (!frecB.ContainsKey(kvp.Key))

{

return 0;

}

else

{

if (frecB[kvp.Key] >= kvp.Value)

{

min = Math.Min(min, frecB[kvp.Key] / kvp.Value);

}

else

{

return 0;

}

}

}

return min;

}

static void Main(string[] args)

{

string A = "abc" ;

string B = "abccba";

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

Console.ReadLine();

}

}

}

------otras soluciones---------------

namespace myjinxin {

public class Kata {

public int StringsConstruction(string A, string B)

{

int diff = B.Length;

foreach (var c in A)

{

int a = A.Length - A.Replace(c.ToString(), "").Length;

int b = B.Length - B.Replace(c.ToString(), "").Length;

if (diff > b / a)

diff = b / a;

}

return diff;

}

}

}

Best Practices0Clever1

0ForkCompare with your solutionLink

AlienHack

namespace myjinxin

{

using System;

public class Kata

{

public int StringsConstruction(string A, string B){

//coding and coding..

int count=0;

int index=0;

bool found = true;

while(B.Length>0) {

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

var regex = new System.Text.RegularExpressions.Regex(System.Text.RegularExpressions.Regex.Escape(""+A[i]));

if(B.Contains(""+A[i]))

B = regex.Replace(B, "", 1);

else {

found = false;

break;

}

}

if(found)

count++;

else

break;

}

return count;

}

}

}

Best Practices0Clever0

0ForkCompare with your solutionLink

mtork07

namespace myjinxin

{

using System;

public class Kata

{

public int StringsConstruction(string A, string B){

bool keepGoing = true;

int matches = 0;

while(keepGoing)

{

int inner = 0;

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

{

if(B.IndexOf(A[i]) != -1)

{

inner ++;

B = B.Remove(B.IndexOf(A[i]), 1);

}

else

{

keepGoing = false;

}

}

if(inner == A.Length)

{

matches ++;

}

}

return matches;

}

}

}

Best Practices0Clever0

0ForkCompare with your solutionLink

mtork07

namespace myjinxin

{

using System;

public class Kata

{

public int StringsConstruction(string A, string B){

//coding and coding..

char [] x = A.ToCharArray();

bool keepGoing = true;

int matches = 0;

while(keepGoing)

{

int inner = 0;

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

{

if(B.IndexOf(A[i]) != -1)

{

inner ++;

B = B.Remove(B.IndexOf(A[i]), 1);

}

else

{

keepGoing = false;

}

}

if(inner == x.Length)

{

matches ++;

}

}

return matches;

}

}

}

Best Practices0Clever0

0ForkCompare with your solutionLink

nachoMonllor

namespace myjinxin

{

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

public class Kata

{

public int StringsConstruction(string A, string B){

//coding and coding..

Dictionary<char, int> frecA = A.ToCharArray() .GroupBy(x => x)

.ToDictionary(x => x.Key, x => x.Count());

Dictionary<char, int> frecB = B.ToCharArray().GroupBy(x => x)

.ToDictionary(x => x.Key, x => x.Count());

int min = int.MaxValue;

foreach (KeyValuePair<char, int> kvp in frecA)

{

if (!frecB.ContainsKey(kvp.Key))

{

return 0;

}

else

{

if (frecB[kvp.Key] >= kvp.Value)

{

min = Math.Min(min, frecB[kvp.Key] / kvp.Value);

}

else

{

return 0;

}

}

}

return min;

}

}

}

Best Practices0Clever0

0ForkCompare with your solutionLink

yonifra

using System;

using System.Collections.Generic;

namespace myjinxin

{

public class Kata

{

public int StringsConstruction(string A, string B)

{

//coding and coding..

Dictionary<char, int> wordToBuild = new Dictionary<char, int>();

Dictionary<char, int> wordsAvailable = new Dictionary<char, int>();

foreach (var letter in A) {

if (wordToBuild.ContainsKey(letter)) {

wordToBuild[letter]++;

}

else

{

wordToBuild.Add(letter, 1);

}

}

foreach (var letter in B) {

if (wordsAvailable.ContainsKey(letter)) {

wordsAvailable[letter]++;

}

else

{

wordsAvailable.Add(letter, 1);

}

}

int count = int.MaxValue;

foreach (var item in wordToBuild)

{

int local = 0;

if (!wordsAvailable.ContainsKey(item.Key)) {

return 0;

}

while (wordsAvailable[item.Key] > 0) {

wordsAvailable[item.Key] -= item.Value;

if (wordsAvailable[item.Key] >= 0) {

local++;

}

}

count = Math.Min(count, local);

}

return count;

}

}

}

Best Practices0Clever0

0ForkCompare with your solutionLink

mrfu2

namespace myjinxin

{

using System;

using System.Collections.Generic;

public class Kata

{

public int StringsConstruction(string A, string B)

{

var charList = new List<char>();

var intList = new List<int>();

// Collect each unique char in string A

foreach( char c in A )

{

if( !charList.Contains( c ) )

{

charList.Add( c );

intList.Add( CountChar( c, A, 0 ) );

}

}

var count = 0;

// Count how many times each char appears in B

foreach( char c in charList )

{

intList[count] = CountChar( c, B, intList[count] );

count++;

}

var intMax = Int32.MaxValue;

// Return the lowest integer, easy peasy.

foreach( int i in intList )

{

if( i < intMax )

{

intMax = i;

}

}

return intMax;

}

public static int CountChar(char c, string str, int baseVal )

{

var count = 0;

foreach(char x in str)

{

if( x == c )

{

count++;

}

}

if(baseVal==0)

{

return count;

} else {

return count/baseVal;

}

}

}

}