**Find Similarities in Lists**

3096% *of* 1336[aryan-firouzian](http://www.codewars.com/users/aryan-firouzian)

* C#
* Mono 4.2.3

Instructions

Output

* **Find Similarity between Two List of Integers**

Develop a method to get two lists of integers and a number **n**. Then it finds any sequence with the length of **n** that exist in both lists.

For example:

List<int> first = new List<int>() { 1, 2, 3, 4, 5, 6 };

List<int> second = new List<int>() { 5, 7, 8, 0, 2, 3, 4, 10 };

int n = 3;

The output will be 2,3,4

Or for:

List<int> first = new List<int>() { 1, 2, 3, 4, 5, 6 };

List<int> second = new List<int>() { 5, 7, 8, 0, 2, 3, 4, 10 };

int n = 2;

The output will be 2,3 | 3,4

, separates the item in the sequence for a similarity. (without space)

| separates the simliraties (with space before and after)

<http://www.codewars.com/kata/find-similarities-in-lists/train/csharp>

//----------BASADO EN LONGEST COMMON SUBSTRING – ROSETTACODE

<https://rosettacode.org/wiki/Longest_Common_Substring#Using_dynamic_programming>

//LA DIFERENCIA ES QUE HAY QUE BUSCAR TODOS LOS SUBSTRING COMUNES, QUE PUEDEN O NO ESTAR DENTRO DE LA SUBCADENA COMUN MAYOR. DONDE lengts es > 0 HAY UN SUBSTRING COMUN

**public** **static** **string** lcs**(string** a, **string** b**)**  
 **{**  
 **var** lengths = [**new**](http://www.google.com/search?q=new+msdn.microsoft.com) **int[**a.**Length**, b.**Length]**;  
 **int** greatestLength = **0**;  
 **string** output = "";  
 **for** **(int** i = **0**; i < a.**Length**; i++**)**  
 **{**  
 **for** **(int** j = **0**; j < b.**Length**; j++**)**  
 **{**  
 **if** **(**a**[**i**]** == b**[**j**])**  
 **{**  
 lengths**[**i, j**]** = i == **0** || j == **0** ? **1** : lengths**[**i - **1**, j - **1]** + **1**;  
 **if** **(**lengths**[**i, j**]** > greatestLength**)**  
 **{**  
 greatestLength = lengths**[**i, j**]**;  
 output = a.**Substring(**i - greatestLength + **1**, greatestLength**)**;  
 **}**  
 **}**  
 **else**  
 **{**  
 lengths**[**i, j**]** = **0**;  
 **}**  
 **}**  
 **}**  
 **return** output;  
 **}**  
 **}**

MI SOLUCION:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp91

{

class Program

{

//public static string AnySimilarity(List<int> a, List<int> b, int n)

public static string AnySimilarity(List<int> firstList, List<int> secondList, int n)

{

int[,] lengths = new int[firstList.Count, secondList.Count];

List<List<int>> allCommonStrings = new List<List<int>>();

for (int i = 0; i < firstList.Count; i++)

{

for (int j = 0; j < secondList.Count; j++)

{

if (firstList[i] == secondList[j])

{

//el indice donde termina el subarray

lengths[i, j] = i == 0 || j == 0 ? 1 : lengths[i - 1, j - 1] + 1;

}

else

{

lengths[i, j] = 0;

}

}

}

if (firstList.Count >= secondList.Count) //ESTO ESTÁ MAL

{

for (int i = 0; i < lengths.GetLength(0); i++)

{

for (int j = 0; j < lengths.GetLength(1); j++)

{

//Console.Write(lengths[i, j] + " ");

if (lengths[i, j] >= n)

{

List<int> aux = new List<int>();

for (int k = i; k < i + lengths[i, j]; k++) //o sea si usas el indice i, es para la primer lista, si usas j es para la segunda (k=i)

{

aux.Add(firstList[k - lengths[i, j] + 1]);

}

allCommonStrings.Add(aux);

}

}

//Console.WriteLine();

}

}

else //ESTO ESTÁ MAL – CON EL FOR ANTERIOR ES SUFICIENTE

{

for (int i = 0; i < lengths.GetLength(0); i++)

{

for (int j = 0; j < lengths.GetLength(1); j++)

{

//Console.Write(lengths[i, j] + " ");

if (lengths[i, j] >= n)

{

List<int> aux = new List<int>();

for (int k = j; k <j + lengths[i, j]; k++)

{

aux.Add(secondList[k - lengths[i, j] + 1]);

}

allCommonStrings.Add(aux);

}

}

//Console.WriteLine();

}

}

List<List<int>> gruposN = new List<List<int>>();

foreach (List<int> lista in allCommonStrings)

{

for(int i =0; i<lista.Count-n+1; i++)

{

List<int> aux = new List<int>();

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

{

aux.Add(lista[j]);

}

gruposN.Add(aux);

}

}

HashSet<string> hs = new HashSet<string>();

foreach(List<int> lista in gruposN )

{

hs.Add(string.Join(",", lista.ToArray()));

}

return string.Join(" | ", hs.ToArray()) ;

}

static void Main(string[] args)

{

//List<int> first = new List<int>() { 1, 2, 3, 4, 5, 6 };

//List<int> second = new List<int>() { 5, 7, 8, 0, 2, 3, 4, 10 };

//int n = 3;

// LCSubStr(first, second, first.Count, second.Count);

List<int> first = new List<int>() { 1, 2, 3, 4, 5, 6 };

List<int> second = new List<int>() { 5, 7, 8, 0, 2, 3, 4, 10 };

int n = 2;//"2,3 | 3,4"

//List<int> second = new List<int>() { 1, 2, 3, 4, 5, 6 };

//List<int> first = new List<int>() { 5, 7, 8, 0, 2, 3 };

//int n = 2;//"2,3 | 3,4"

//Console.WriteLine(AnySimilarity(first, second, n));

//List<int> first = new List<int>() { 1, 2, 3, 4 };

//List<int> second = new List<int>() { 1, 2, 3, 4 };

//int n = 3;

Console.WriteLine( AnySimilarity(first, second, n));

Console.ReadLine();

}

}

}

------------MI SOLUCION ACORTADA (PROGRAMACION DINAMICA)--------------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp94

{

class Program

{

//public static string AnySimilarity(List<int> a, List<int> b, int n)

public static string AnySimilarity(List<int> firstList, List<int> secondList, int n)

{

int[,] lengths = new int[firstList.Count, secondList.Count];

//List<List<int>> allCommonStrings = new List<List<int>>();

HashSet<string> gruposN = new HashSet<string>();

for (int i = 0; i < firstList.Count; i++)

{

for (int j = 0; j < secondList.Count; j++)

{

if (firstList[i] == secondList[j])

{

//el indice donde termina el subarray es i de firstList y j de secondList

lengths[i, j] = i == 0 || j == 0 ? 1 : lengths[i - 1, j - 1] + 1;

}

else

{

lengths[i, j] = 0;

}

if (lengths[i, j] >= n)

{

List<int> aux = firstList.GetRange(i - lengths[i, j] + 1, lengths[i, j]);

for (int k = 0; k < aux.Count - n + 1; k++)

{

gruposN.Add(string.Join(",", aux.GetRange(k, n)));

}

}

}

}

return string.Join(" | ", gruposN.ToArray());

}

static void Main(string[] args)

{

//List<int> first = new List<int>() { 1, 2, 3, 4, 5, 6 };

//List<int> second = new List<int>() { 5, 7, 8, 0, 2, 3, 4, 10 };

//int n = 3;

// LCSubStr(first, second, first.Count, second.Count);

List<int> second = new List<int>() { 1, 2, 3, 4, 5, 6 };

List<int> first = new List<int>() { 5, 7, 8, 0, 2, 3, 4, 10 };

int n = 2;//"2,3 | 3,4"

//List<int> first = new List<int>() { 1,1,1 };

//List<int> second = new List<int>() { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 };

//int n = 2;//"2,3 | 3,4"

//List<int> first = new List<int>() { 1, 1, 0 };

//List<int> second = new List<int>() { 1, 1, 1};

//int n = 2;//"2,3 | 3,4"

//List<int> second = new List<int>() { 1, 2, 3, 4, 5, 6 };

//List<int> first = new List<int>() { 5, 7, 8, 0, 2, 3 };

//int n = 2;//"2,3 | 3,4"

//Console.WriteLine(AnySimilarity(first, second, n));

//List<int> first = new List<int>() { 1, 2, 3, 4 };

//List<int> second = new List<int>() { 1, 2, 3, 4 };

//int n = 3;

Console.WriteLine(AnySimilarity(first, second, n));

Console.ReadLine();

}

}

}

-----------------SOLUCIONES DE OTROS -------------------------------

[**dcieslak**](http://www.codewars.com/users/dcieslak)

public class Kata

{

public static string AnySimilarity(List<int> f, List<int> s, int n)

{

List<string> res = new List<string>();

for (int i = 0; i < f.Count() - n + 1; i++)

{

string subSeq = string.Join(",", f.Skip(i).Take(n));

if (string.Join(",", s).Contains(subSeq) && !res.Contains(subSeq)) res.Add(subSeq);

}

return string.Join(" | ", res);

}

}

[**jacky33man**](http://www.codewars.com/users/jacky33man)

public static string AnySimilarity(List<int> first, List<int> second, int n)

{

var a = Enumerable.Range(0, first.Count - n / 2).Select(i => first.Skip(i).Take(n)).Where(x => x.Count() == n).Select(x => string.Join(",", x));

var b = Enumerable.Range(0, second.Count - n / 2).Select(i => second.Skip(i).Take(n)).Where(x => x.Count() == n).Select(x => string.Join(",", x));

return string.Join(" | ", a.Intersect(b));

}