Given an array of equal-length strings, check if it is possible to rearrange the strings in such a way that after the rearrangement the strings at consecutive positions would differ by exactly one character.

**Example**

* For inputArray = ["aba", "bbb", "bab"], the output should be  
  stringsRearrangement(inputArray) = false;
* For inputArray = ["ab", "bb", "aa"], the output should be  
  stringsRearrangement(inputArray) = true.

**Input/Output**

* **[time limit] 6000ms (cs)**
* **[input] array.string inputArray**

A non-empty array of strings of lowercase letters.

*Constraints:*  
2 ≤ inputArray.length ≤ 10,  
1 ≤ inputArray[i].length ≤ 15.

* **[output] boolean**

<https://codefights.com/arcade/intro/level-7/PTWhv2oWqd6p4AHB9/description>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication2

{

class Program

{

public static bool nextPermutation(int[] array, string[] s)

{

// Find non-increasing suffix

int i = array.Length - 1;

while (i > 0 && array[i - 1] >= array[i])

i--;

if (i <= 0)

return false;

// Find successor to pivot

int j = array.Length - 1;

while (array[j] <= array[i - 1])

j--;

int temp = array[i - 1];

array[i - 1] = array[j];

array[j] = temp;

string ts = s[i - 1];

s[i - 1] = s[j];

s[j] = ts;

// Reverse suffix

j = array.Length - 1;

while (i < j)

{

temp = array[i];

array[i] = array[j];

array[j] = temp;

ts = s[i];

s[i] = s[j];

s[j] = ts;

i++;

j--;

}

return true;

}

static bool difiereEn1(string s1, string s2)

{

int n = s1.Length;

int count = 0; // Count of edits

int i = 0, j = 0;

while (i < n && j < n)

{

// If current characters don't match

if (s1[i] != s2[j])

{

if (count == 1)

return false;

i++;

j++;

count++;

}

else // If current characters match

{

i++;

j++;

}

}

// If last character is extra in any string

if (i < n || j < n)

count++;

return count == 1;

}

static bool stringsRearrangement(string[] inputArray)

{

int[] indices = new int[inputArray.Length];

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

{

indices[i] = i;

}

do

{

int i = 0;

for (i = 0; i + 1 < inputArray.Length; i++)

{

// Console.Write(inputArray[i] + " ");

if (!difiereEn1(inputArray[i], inputArray[i + 1]))

{

break;

}

}

if (i == inputArray.Length-1) return true;

// Console.WriteLine();

} while (nextPermutation(indices, inputArray));

return false;

}

static void Main(string[] args)

{

//string[] inputArray = { "aba", "bbb", "bab" };

//string[] inputArray = { "a", "b", "c" };

string[] inputArray = { "aa", "ab", "bb" };

//string [] inputArray= {"aba",

// "bbb",

// "bab"};//false

Console.WriteLine(stringsRearrangement(inputArray));

Console.ReadLine();

}

}

}