Given a string consisting of lowercase English letters, find the largest square which can be obtained by *reordering* its characters and *replacing* them with digits (leading zeros are not allowed) where same characters always map to the same digits and different characters always map to different digits.

If there is no solution, return -1.

**Example**

* For s = "ab", the output should be  
  constructSquare(s) = 81;
* For s = "zzz", the output should be  
  constructSquare(s) = -1.

There are no 3-digit square numbers with identical digits.

* For s = "aba", the output should be  
  constructSquare(s) = 900.

It can be obtained after reordering the initial string into "baa".

**Input/Output**

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

*Constraints:*  
2 ≤ s.length < 10.

* **[output] integer**

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

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace ConsoleApplication2

{

class Program

{

public static bool nextPermutation(char[] array)

{

// 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--;

char temp = array[i - 1];

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

array[j] = temp;

// Reverse suffix

j = array.Length - 1;

while (i < j)

{

temp = array[i];

array[i] = array[j];

array[j] = temp;

i++;

j--;

}

return true;

}

static bool isSubstitutionCipher(string string1, string string2)

{

int n = string1.Length;

Dictionary<char, char> diccio = new Dictionary<char, char>();

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

{

if (diccio.ContainsKey(string1[i]))

{

if (string2[i] != diccio[string1[i]])

return false;

}

else

{

if (diccio.ContainsValue(string2[i]))

{

return false;

}

diccio[string1[i]] = string2[i];

}

}

return true;

}

static int constructSquare(string s)

{

// int[] cuad = { 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361 };

List<long> cuad = new List<long>();

for (long i = 1; i <= 100000; i++)

{

cuad.Add(i \* i);

}

string[] cuadString = Array.ConvertAll(cuad.ToArray(), e => e.ToString());

//Array.Reverse(cuadString);

for (int i = cuadString.Length - 1; i>=0 ; i--)

{

if (cuadString[i].Length == s.Length)

{

char[] schar = s.ToCharArray();

Array.Sort(schar);

//while (nextPermutation(schar))

do

{

if (isSubstitutionCipher(new string(schar), cuadString[i]))

{

return int.Parse(cuadString[i]);

}

} while (nextPermutation(schar));

}

}

return -1;

}

static void Main(string[] args)

{

//using (StreamWriter esc = new StreamWriter("C:\\cuad.txt"))

//{

// for (int i = 1; i < 20; i++)

// {

// esc.Write((i \* i) + ", ");

// }

//}

// constructSquare("81");

//string s = "ghuias";

//char[] schar = s.ToCharArray();

//Array.Sort(schar);

//Console.WriteLine(new string(schar));

//string s = "zzz";

//string s = "aba";

//string s = "abcbbb";

//string s = "abc";

//string s = "aaaaaaaacb";

string s = "a";

Console.WriteLine(constructSquare(s));

//string s = "ab";

//char[] ch = s.ToCharArray();

////while (nextPermutation(ch))

////{

//// Console.WriteLine(new string(ch));

////}

//do

//{

// Console.WriteLine(new string(ch));

//} while (nextPermutation(ch));

Console.ReadLine();

}

}

}