Define an integer's *roundness* as the number of trailing zeros in it. Sometimes it is possible to increase a number's *roundness* by swapping two of its digits.

Given an integer n, find the minimum number of swaps required to maximize n's roundness.

Example

* For n = 902200100, the output should be  
  maximizeNumberRoundness(n) = 1.

It's enough to swap the leftmost 0 with 1.

* For n = 11000, the output should be  
  maximizeNumberRoundness(n) = 0.

n already has the maximum roundness possible.

Input/Output

* **[execution time limit] 3 seconds (cs)**
* **[input] integer n**

A positive integer.

*Guaranteed constraints:*  
104 ≤ n ≤ 109.

* **[output] integer**
  + The minimum number of swaps required to maximize n's roundness.

<https://app.codesignal.com/challenge/h5rpQQTrM3cCCevmq>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication2

{

class Program

{

static int maximizeNumberRoundness(int n)

{

string s = n.ToString();

int cont = 0;

char[] ch = s.ToCharArray();

int izq =0, der = s.Length-1;

int i = izq, j = der;

while (izq < der)

{

// int i = 0;

for ( ; izq < der ; izq++)

{

if (s[izq] == '0')

{

i = izq;

break;

}

}

// int j = s.Length - 1;

for (; der > izq; der--)

{

if (s[der] != '0')

{

j = der;

break;

}

}

if (i < j && s[izq] == '0' && s[der] != '0')

{

char temp = ch[i];

ch[i] = ch[j];

ch[j] = temp;

izq++;

der--;

cont++;

}

}

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

return cont;

}

static void Main(string[] args)

{

Console.WriteLine( maximizeNumberRoundness(902200100));

Console.ReadLine();

}

}

}