Consider a sequence of numbers a0, a1, ..., an, in which an element is equal to the sum of squared digits of the previous element. The sequence ends once an element that has already been in the sequence appears again.

Given the first element a0, find the length of the sequence.

**Example**

* For a0 = 16, the output should be  
  squareDigitsSequence(a0) = 9.

Here's how elements of the sequence are constructed:

* + a0 = 16
  + a1 = 12 + 62 = 37
  + a2 = 32 + 72 = 58
  + a3 = 52 + 82 = 89
  + a4 = 82 + 92 = 145
  + a5 = 12 + 42 + 52 = 42
  + a6 = 42 + 22 = 20
  + a7 = 22 + 02 = 4
  + a8 = 42 = 16, which has already occurred before (a0)

Thus, there are 9 elements in the sequence.

* For a0 = 103, the output should be  
  squareDigitsSequence(a0) = 4.

The sequence goes as follows: 103 -> 10 -> 1 -> 1, 4 elements altogether.

**Input/Output**

* **[time limit] 3000ms (cs)**
* **[input] integer a0**

First element of a sequence, positive integer.

*Constraints:*  
1 ≤ a0 ≤ 650.

* **[output] integer**

<https://codefights.com/arcade/code-arcade/labyrinth-of-nested-loops/MvX84CA5HN6GKqv7R>

static int squareDigitsSequence(int a0)

{

int ans = 1;

string ns = a0.ToString();

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

while (true)

{

int sumDig = 0;

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

{

sumDig += int.Parse(ns[i].ToString()) \* int.Parse(ns[i].ToString());

}

ans++;

if (hs.Contains(sumDig))

{

return ans;

}

hs.Add(sumDig);

if (sumDig.ToString() == a0.ToString())

{

break;

}

if (sumDig == 1)

{

return ans + 1;

}

ns = sumDig.ToString();

}

return ans;

}