OJ 10527

Persistent Numbers

Prof. Edson Alves - UnB/FGA

OJ 10527 – Persistent Numbers

The multiplicative persistence of a number is defined by Neil Sloane (Neil J.A. Sloane in 'The Persistence of a Number' published in Journal of Recreational Mathematics 6, 1973, pp. 97-98., 1973) as the number of steps to reach a one-digit number when repeatedly multiplying the digits. Example:

$$679 \to 378 \to 168 \to 48 \to 32 \to 6.$$

That is, the persistence of 679 is 5. The persistence of a single digit number is 0. At the time of this writing it is known that there are numbers with the persistence of 11. It is not known whether there are numbers with the persistence of 12 but it is known that if they exists then the smallest of them would have more than 3000 digits.

The problem that you are to solve here is: what is the smallest number such that the first step of computing its persistence results in the given number?

Entrada e saída

Input

For each test case there is a single line of input containing a decimal number with up to 1000 digits. A line containing –1 follows the last test case.

Output

For each test case you are to output one line containing one integer number satisfying the condition stated above or a statement saying that there is no such number in the format shown below.

Exemplo de entrada e saída

Entrada	Saída
0	10
1	11
4	14
7	17
18	29
49	77
51	There is no such number.
768	2688
-1	

Solução $O(\log n)$

- Se n é o produto dos dígitos de x, então a fatoração prima de n só pode conter primos cuja representação decimal só tem um dígito, a saber: 2, 3, 5 e 7
- Assim, se a fatoração de n tem qualquer outro primo o problema não terá solução
- Nos demais casos, a fatoração prima de n seria uma solução, embora nem sempre seja a mínima
- Para minimizar a solução, é preciso agrupar os fatores primos em dígitos compostos
- Antes de fazer esta redução, tratemos primeiro de um caso especial

Solução $O(\log n)$

- Mesmo não estando explícito no texto do problema, espera-se que x tenha, no mínimo, dois dígitos, conforme se observa nos exemplos de entrada e saída
- Assim, se n tiver um único digo, a solução mínima seria o número $10 + n$
- Nos demais casos, para minimizar x agruparemos os fatores primos nos compostos 9, 8, 6 e 4, nesta ordem, de forma gulosa
- Feito este agrupamento, x é formado por estes agrupamentos, ordenados do menor para o maior

Solução $O(\log n)$

```
5  def factorization_2357(x):
6
7      fs = [0]*10
8
9      for p in [2, 3, 5, 7]:
10         while x % p == 0:
11             fs[p] += 1
12             x //= p
13
14     if x > 1:
15         fs = []
16
17     return fs
```

Solução $O(\log n)$

```
20 def solve(n):
21
22     if len(n) == 1:
23         return '1' + n
24
25     fs = factorization_2357(int(n))
26
27     if not fs:
28         return 'There is no such number.'
29
30     fs[9] = fs[3] // 2
31     fs[3] %= 2
32
33     fs[8] = fs[2] // 3
34     fs[2] %= 3
35
```

Solução $O(\log n)$

```
36      if fs[3] > 0 and fs[2] > 0:
37          fs[6] = 1
38          fs[2] -= 1
39          fs[3] -= 1
40
41      fs[4] = fs[2] // 2
42      fs[2] %= 2
43
44      x = 0
45
46      for d in range(2, 10):
47          while fs[d] > 0:
48              x = 10*x + d
49              fs[d] -= 1
50
51      return str(x)
```

Solução $O(\log n)$

```
54 if __name__ == '__main__':
55
56     ns = [x.strip() for x in sys.stdin.readlines()]
57     ns = takewhile(lambda x: x != '-1', ns)
58     xs = map(solve, ns)
59
60     print('\n'.join(xs))
```