ProjectEuler_40

June 3, 2021

1 Champernowne's constant

1.1 Problem 40

An irrational decimal fraction is created by concatenating the positive integers:

0.12345678910111213141516171819202l…

It can be seen that the 12th digit of the fractional part is 1.

If d_n represents the nth digit of the fractional part, find the value of the following expression.

d_1 × d_10 × d_100 × d_1000 × d_10000 × d_100000 × d_1000000

Soluzione pragmatica e compatta anche se non veloce: comporre la sequenza di cifre, selezionare le cifre da moltiplicare e … moltiplicarle.

Fino a quale numero occorre concatenare?
Per avere una stima ragionevolmente corretta si può partire dalla considerazione che **le prime 9** cifre della parte frazionaria sono le cifre **da 1 a 9**.
Le successive **2 x 90 cifre** sono costitute dalla sequenza di interi da **10 a 99**.
Poi seguono le **3 x 900 cifre da 100 a 999**, e così via.

In realtà si può tagliare la testa al toro e sommare i primi 1000000 di numeri, producendo sicuramente molte più cifre di quante ne servano, ma al costo di qualche secondo di calcolo e un po' di memoria.

[32]:
```
s = "".join(str(i) for i in range(1, 1000000))
print ("costruita la sequenza delle prime", len(s), "cifre della parte␣
 ↪frazionaria")
p = 1
for i in range(7):
    p *= int(s[10**i - 1])
print ("il prodotto cercato è", p)
```

```
costruita la sequenza delle prime 5888889 cifre della parte frazionaria
il prodotto cercato è 210
```

Una soluzione più veloce, anche se con un codice meno compatto, si ha individuando l'intero della sequenza in cui si trova ciascuna delle cifre **d_n** necessarie, e poi prelevando la cifra giusta.

[33]:
```
def cifra(x):
    intero = 1
    up_limit = 9
    potenza = 1
    trovato = False
    while trovato is False:
        if x > potenza * up_limit:
            x -= potenza * up_limit
            intero += up_limit
            up_limit *= 10
            potenza += 1
        else:
            intero += (x-1) // potenza
            posizione_cifra = potenza - (x-1) % potenza -1
            trovato = True
    return (intero//10**posizione_cifra) % 10

print (cifra(1) * cifra(10) * cifra(100) * cifra(1000) * cifra(10000) *␣
 ↪cifra(100000) * cifra(1000000))
```

```
210
```