OJ 763

Fibinary Numbers

Prof. Edson Alves - UnB/FGA

OJ 763 – Fibinary Numbers

The standard interpretation of the binary number 1010 *is* $8 + 2 = 10$*. An alternate way to view the sequence "`1010`" is to use Fibonacci numbers as bases instead of powers of two. For this problem, the terms of the Fibonacci sequence are:*

$$1, 2, 3, 5, 8, 13, 21, \ldots$$

Where each term is the sum of the two preceding terms (note that there is only one 1 *in the sequence as defined here). Using this scheme, the sequence "`1010`" could be interpreted as* $1 \times 5 + 0 \times 3 + 1 \times 2 + 0 \times 1 = 7$*. This representation is called a Fibinary number.*

Note that there is not always a unique Fibinary representation of every number. For example the number 10 could be represented as either $8 + 2$ (10010) *or as* $5 + 3 + 2$ (1110)*. To make the Fibinary representations unique, larger Fibonacci terms must always be used whenever possible (i.e. disallow* 2 *adjacent* 1*'s). Applying this rule to the number* 10*, means that 10 would be represented as* $8 + 2$ (10010)*.*

Write a program that takes two valid Fibinary numbers and prints the sum in Fibinary form.

Entrada e saída

Input

The input file contains several test cases with a blank line between two consecutive.

Each test case consists in two lines with Fibinary numbers. These numbers will have at most 100 *digits.*

Output

For each test case, print the sum of the two input numbers in Fibinary form.

It must be a blank line between two consecutive outputs.

Exemplo de entrada e saída

Entrada	Saída
`10010` `1`	`10100`
`10000` `1000`	`100000`
`10000` `10000`	`100100`

Solução em $O(N)$

- Primeiramente é preciso observar que não é possível somar diretamente os números em base de Fibonacci
- Por exemplo, em base de Fibonacci o número 5 é representado por `1000` e a soma $5 + 5 = 10$ teria representação `10010`
- Veja que ao somar os dois dígitos `1` correspondentes, o dígito que o ocupa a segunda posição da representação, o qual já teria sido processado, foi modificado
- Assim, a solução consiste em converter a e b para a base decimal, obter a soma $c = a + b$ e converter c para a base de Fibonacci
- Se $N = \max\{|a|, |b|\}$, então esta solução tem complexidade $O(N)$

Solução em $O(N)$

```python
import sys


fibs = [1, 2]

while len(fibs) <= 101:
    fibs.append(fibs[-1] + fibs[-2])


def to_decimal(n):

    ys = list(map(lambda x: int(x), n))[::-1]

    return sum(map(lambda x, y: x*y, fibs, ys))
```

Solução em $O(N)$

```python
def to_fibinary(n):

    if n == 0:
        return '0\n'

    res = []

    for fib in fibs[::-1]:
        if fib <= n:
            n -= fib
            res.append('1')
        else:
            res.append('0')

    return ''.join(res).lstrip('0') + '\n'


```

Solução em $O(N)$

```python
def fibinary_sum(p):

    a, b = p
    n = to_decimal(a) + to_decimal(b)
    return to_fibinary(n)


def solve(xs):

    return map(lambda p: fibinary_sum(p), xs)


if __name__ == '__main__':

    xs = [x.strip() for x in sys.stdin.readlines() if x.strip()]
    xs = list(zip(xs[::2], xs[1::2]))
    print('\n'.join(solve(xs)), end='')
```