Solving Euler 53 With Scheme

Sutthichok Wongthavorn

May 24, 2012

You know what, Ive just managed to solve a *Project Euler* problem using Scheme for the first time! First code outside SICP!! Exciting.

Problem 57 says a square root of two can be expressed as an infinite continued fraction. It wants me to find out how many fractions within the first 1,000 expansions that numerator has more digits than the denominator.

$$\sqrt{2} = 1 + 1/(2 + 1/(2 + 1/(2 + ...))) = 1.414212...$$

First 4 expansions turn out to be something like this.

(1st) `1 + 1/2 = 3/2 = 1.5`

(2nd) `1 + 1/(2 + 1/2) = 7/5 = 1.4`

(3rd) `1 + 1/(2 + 1/(2 + 1/2)) = 17/12 = 1.41666...`

(4th) `1 + 1/(2 + 1/(2 + 1/(2 + 1/2))) = 41/19 = 1.41379...`

Not until the 8th expansion, `1393/985`, that the numerator has more digits than denominator.

First I thought about attacking this problem with the usual tool, Ruby. But after looking closely at the equation, I see recursion. So I think, I could try this one with Scheme. After shuffling around the parenthesis for several hours (since last night and continued this afternoon), I came up with this.

```
;; Find nth expansion of square root x
(define (sqrte x n)
  (define (iter expansion)
    (if (= expansion 0)
        0
        (/ 1 (+ x (iter (- expansion 1))))))
  (+ 1 (iter n)))

(define (euler57 n max)
  (define (iter times count)
    (define sq 0)
    (if (> times max)
        count
        (begin
          (set! sq (sqrte n times))
          (if (>
               (string-length (number->string (numerator sq)))
               (string-length (number->string (denominator sq))))
              (set! count (+ 1 count)))
          (iter (+ 1 times) count))))
  (iter 0 0))

(display (euler57 2 1000))
```

Listing 1: Project Euler 057 in Scheme

The code doesnt look too messy. So Im quite happy with it.