7.2 循環節の計算

循環小数の循環節を無駄なく見ることができるようになった。しかし、それでは不十分だったね。われわれが知りたいのは、$142857 \times 2 = 285714$ のように循環節が巡回をしているかどうか、である。quotient 関数は、循環節を眺めることはできても計算をすることはできない。なぜなら、quotient 関数が吐き出した数は、まとまった数値でなく桁ごとの数だからだ。

桁ごとの数をまとまった数値にするには、ひとつの変数にまとめてやればよい。それは quotient 関数をちょいといじってやれば簡単に出来そうだ。

[hs script]
```
fquotient :: [Integer] -> [Integer]
fquotient (x:xs)
    | r == 1    = drop 1 xs ++ [last xs * 10 + q]
    | otherwise = fquotient (r : xs ++ [last xs * 10 + q])
    where q = (10*x) `div` head xs
          r = (10*x) `mod` head xs
```

いじるところは商をリストに加える部分だ。商をリストに加えるたびに、あらかじめ 10 倍しておけばよいだろう。さて、試運転といこう。

(ghci env.)
```
*Main> fquotient [1,7]
[71,714,7142,71428,714285,7142857]
```

おっと、残念。先頭の 7 が残ってしまった。quotient 関数を変更するだけではうまくないようだ。そこで、リストを引数に与えると、それを集めてひとつの整数にする関数を作っておこう。回り道かもしれないけど、何事も根回し、じゃなくて下ごしらえを怠ってはいけないよ。

[hs script]
```
qsum :: [Integer] -> Integer
qsum (x:y:[]) = 10*x+y
qsum (x:y:ys) = qsum (10*x+y : ys)
```

(x:y:ys) という表現は、リストの先頭の要素を x、2 番目の要素を y として、残りをリスト ys として扱うことを意味する。qsum 関数の使い方を示そう。

(ghci env.)
```
*Main> qsum [1,2,3,4,5,6,7,9]
12345679
*Main> qsum [1,2,3,4,5,6,7,9] * 7
86419753
*Main> qsum [1,2,3,4,5,6,7,9] * 7 * 9
777777777
```

( ) でくくらなくても qsum [1,2,3,4,5,6,7,9] * 7 がどこから処理を始めるか分かっているね。*より関数 qsum の結合の方が強いんだよ。ちなみに、12345679 という数は面白い性質を持っていて、見て分かるように、数 n を掛けただけでは何の変哲もないのだが、その直後に 9 を掛けると n が並ぶのである。もちろん仕組みは単純なことで、先に 9 を掛ければ種が分かる。

(ghci env.)

```
*Main> qsum [1,2,3,4,5,6,7,9] * 9
111111111
```

ところで、(x:y:[]) は要素が 2 個あるリストを前提にしている。すると、この関数には不備があることが分かる。要素が 1 個のリストは扱えないからだ。循環節が 1 桁の数には興味がないと言うものの、エラーを起こさないようにするには、たとえば次のようにしておかなくてはならない。

[hs script]

```
qsum :: [Integer] -> Integer
qsum [] = 0
qsum (y:[]) = y
qsum (x:y:[]) = 10*x+y
qsum (x:y:ys) = qsum (10*x+y : ys)
```

つまり、リストが空だったり、要素が 1 個しかなかったりしたときに、きちんと値を返すようにしておくのだ。これで要素が 1 個のリストを qsum 関数に与えても大丈夫だ。前に書いた、$\frac{1}{n}$ の循環節をリスト表示する quotient 関数を qsum の引数に与えてみよう。

(ghci env.)

```
*Main> qsum (quotient [1,9])
1
```

ほらね。$\frac{1}{9} = 0.111\cdots$ の循環節は 1 だけなので、その値である 1 が示された。ちなみに、ここでは ( ) は必須である。( ) がないと、qsum 関数は引数にリストを期待しているのに、quotient が与えられて困惑してしまう。でも、一桁の循環節を正しく数値で見ることができても面白いことはない。

さあ、これで $\frac{1}{7}$ の循環節がどのように循環するかが計算できる。分母は 7 だから 1 から 6 までの掛け算を見ればよいだろう。

(ghci env.)

```
*Main> map (* qsum (quotient [1,7])) [1..6]
[142857,285714,428571,571428,714285,857142]
```

いいね。でも、まだ課題がある。$\frac{1}{7}$ 程度ならリストを眺めて巡回の様子がつかめるが、循環節がもっと長いときは大変である。リストは、ほとんど数字の羅列にしか見えないはずだ。実際、

(ghci env.)
```
map (* qsum (quotient [1,113])) [1..112]
```

なんかを試してみればよい。蟻の巣を突つくことになるけど。

また、このようなリストを見比べるために何度か実行することになったら、quotient の引数とマッピングさせる範囲を変えて実行するだろう。マッピングの範囲は quotient の引数より 1 小さいので

(ghci env.)
```
*Main> let qsumlist n = map (* qsum (quotient [1,n])) [1..n-1]
```

のように定義しておけば、かなりお手軽なスクリプトになる。

(ghci env.)
```
*Main> qsumlist 7
[142857,285714,428571,571428,714285,857142]
*Main> qsumlist 13
[76923,153846,230769,307692,384615,461538,538461,615384,692307,769230,8461
53,923076]
```

さあ、これでいろいろな分数の循環節の巡回の様子を調べることが簡単になった。でも、リスト表示は少々見づらい。もう少しなんとかできないだろうか。

しかし、表示の改善の前にちょっと気になる点がある。$\frac{1}{13}$ の循環節の巡回の仕方は $\frac{1}{7}$ の循環節の巡回の仕方とちょっと違うんじゃない？　$\frac{1}{13} = 0.\dot{0}76923\dot{}$ だから循環節は 076923 が繰り返しているものの、なぜか別の循環節 153846 も巡回している。これは何だろう？

少しだけ秘密を教えよう。たとえば $\frac{1}{7}$ の最初の循環節 142857 は、$\frac{1}{7} = 0.\dot{1}42857\dot{}$ の繰り返し部分だけを切り取ったと見てよい。一方で最後の循環節 857142 は、$\frac{6}{7} = 0.\dot{8}57142\dot{}$ の繰り返し部分だけを切り取っている。この 2 つの循環節の和は 999999 であるが、これは $0.\dot{9}99999\dot{}$ であり $\frac{1}{7} + \frac{6}{7}$ になっている。

このことは重要である。$\frac{1}{7}$ の循環節で言えば、同じことが 2 番目と 5 番目で成り立つ。$\frac{2}{7} + \frac{5}{7}$ だ。もちろん 3 番目と 4 番目は $\frac{3}{7} + \frac{4}{7}$ である。

この視点で $\frac{1}{13}$ の循環節を見てほしい。最初と最後の循環節の和から始まって、2 番目と 11 番目、...、6 番目と 7 番目までの循環節の和はすべて 999999 だ。要するに $\frac{1}{n}$ の分数は、$\frac{n-1}{n}$ の分数と組み合わせることで必ず小数点以下の和が $999\cdots9$ になる。$\frac{1}{n} + \frac{n-1}{n} = 1$ なんだから当たり前だよね。

ただしここまでの話では、なぜ循環節が巡回するかの説明には、まったくなっていない。そもそ

も $\frac{1}{7}$ は循環節が 6 であるのに、$\frac{1}{13}$ も循環節は 6 である理由が不明である。他にもいろいろな分数を調べてみなくちゃいけないみたいだ。そのためにも、循環節の巡回を見やすくする必要性は高まった。

ところで話を戻すが、リストを `(x:y:ys)` としてリストの 1, 2 番目の要素を指定できるなら、`(x:xs)` の 2 番目の要素を `head xs` で指定する必要はない。何の話かというと、第 5 週でフィボナッチ数列を扱ったとき、リストの 1, 2 番目の要素を指定するのにかなり無様な書き方をしただろう。最初から `(x:y:ys)` と書けばよかったのだ。是非とも、2 週前にいじり回したところを掘り返して整えてみよう。