1.3 コラッツの問題

有名な数列はフィボナッチ数列だけではない。フィボナッチ数列は不思議な数列だが、他にも未解決の数列がある。それは次の規則で作られる。

はじめに勝手な自然数を用意する。次の項は「前の項が偶数なら 2 で割り、奇数なら 3 倍して 1 を足す」というものだ。たとえば 50 から始めると『50, 25, 76, 38, 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, ...』のような数列が出来上がる。数列はどこまでも続くように思える。しかし、そうではないのだ。繰り返しを増やして調べよう。すると『50, 25, 76, 38, 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, ...』になった。これを出力したコードは『

```
\begin{luacode*}
    n = 50  tex.print(n .. ', ')
    for i = 2, 30 do
        if n % 2 == 0 then
            n = n // 2
        else
            n = 3 * n + 1
        end
        tex.print(n .. ', ')
    end
    tex.print('$\\dots$')
\end{luacode*}
```

』である。

とくに新しい命令は登場していないが、偶数 `n` を `2` で割る際 `//` を用いたのは、`/` では小数点が表示されるためではなく確実に割り切れることが分かっているからだ。もう一つ、`for` 文の繰り返しを `i = 2, 30` としたことを見てほしい。これで繰り返しは 29 回となるが、ここでの `i` は繰り返しの回数というより、第 2 項から第 30 項までの処理と読む方がよいだろう。なぜなら、**初項**を `for` 文の前に出力しているからだ。よって次の処理は 2 項目からという意味で、30 項までが出力される。細かい気遣いというものである。

さて 50 から始めた数列だが、最後の方は 4, 2, 1, ..., が繰り返すようである。50 以外の数ならどうなるだろうか。結論を言ってしまえば、どんな数から始めても必ず 4, 2, 1 で終わることが知られている。知られてはいるが**証明**されたわけではない。つまり未解決問題ということだ。これはコ

ラッツ[1)]の問題などと呼ばれている。

証明されていなくてもコラッツ数が必ず 1 で終わるのなら、スクリプトで項数を決めるのはよくない。数によっては、指定した項数までに 1 が現れないかもしれないからだ。そこで、項が 1 になったところでスクリプトが終了するように書き換えよう。結果は『50, 25, 76, 38, 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1,』となる。このスクリプトは『

```
\begin{luacode*}
  n = 50  tex.print(n .. ', ')
  while n > 1 do
      if n % 2 == 0 then
          n = n // 2
      else
          n = 3 * n + 1
      end
      tex.print(n .. ', ')
  end
\end{luacode*}
```

』である。新しい命令が登場したようだ。

と言っても、`for i = 2, 30` を `while n > 1` に差し替えたにすぎない。`while` 構文は、**条件式**が**真**である限り `do` ループを繰り返す命令である。この場合は `n > 1` が真である—すなわち `n` が `1` より大きな数である—限りループを繰り返す。それで `n` が `1` になったところでループから抜けるのである。最後の `end` が `while do` と対（つい）になっている。

コラッツ数は 27 から始めるとなかなか楽しめる。`n = 27` とすると『27, 82, 41, 124, 62, 31, 94, 47, 142, 71, 214, 107, 322, 161, 484, 242, 121, 364, 182, 91, 274, 137, 412, 206, 103, 310, 155, 466, 233, 700, 350, 175, 526, 263, 790, 395, 1186, 593, 1780, 890, 445, 1336, 668, 334, 167, 502, 251, 754, 377, 1132, 566, 283, 850, 425, 1276, 638, 319, 958, 479, 1438, 719, 2158, 1079, 3238, 1619, 4858, 2429, 7288, 3644, 1822, 911, 2734, 1367, 4102, 2051, 6154, 3077, 9232, 4616, 2308, 1154, 577, 1732, 866, 433, 1300, 650, 325, 976, 488, 244, 122, 61, 184, 92, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1,』が出力される。これは、先のスクリプトを `n = 27` に変えただけだ。最後の , が恥ずかしければ、`tex.print` 文を工夫しよう。

また、コラッツの問題はちょっとした拡張ができる。前の項が偶数なら 2 で割り、奇数なら 3 倍

1) ローター・コラッツ（1910–1990）：ドイツの数学者。

して 1 を足すという規則を、たとえば奇数なら 5 倍して 3 を引く、みたいにしてみることだ。これは `n = 3 * n + 1` を `n = 5 * n - 3` に変えるだけなので、すぐに実行に移せるだろう。もちろん他の規則にしてもよいし、偶数の規則を変えたって構わない。ただし、`while n > 1` を条件にするのはやめよう。終了しない危険がある。

では、いくつか試してみることを勧める。ところが実際にやってみると、コラッツの問題ほどうまくハマらないことが分かるだろう。すぐに循環したり、いつまでたっても収束の気配を見せなかったりとね。なぜ、$3n+1$ だとうまく運ぶのだろうか。このことは未だ謎なのだ。もっともらしい理由はあることはあるのだが、決定的な説明ではない。興味をそそられたら取り組んでみるとよいだろう。