5.4 究極の連分数

黄金比が

$$x = 1 + \frac{1}{x} \tag{1}$$

の解であることは述べたね。しかし、違った考えで x の解を求めることもできる。

ややこしい話だけれど、(1) の $x = 1 + 1/x$ を (1) の右辺の x に代入してみることだ。つまり

$$x = 1 + \frac{1}{1 + \frac{1}{x}}$$

ということだ。しかし、右辺にはまだ x が存在するので、再び $x = 1 + 1/x$ を右辺にある x に代入すると

$$x = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \frac{1}{x}}}$$

となるだろう。さらに、右辺にはまだ x が存在するので . . .。きりがないね。だが、これを無限に続ければ

$$x = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cdots}}} \tag{2}$$

となることが理解できるだろうか。

このように、分数の中に分数が幾重にもなる分数を**連分数**と呼んでいる。連分数は一般に

$$a = q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cfrac{1}{q_4 + \cdots}}}$$

で表すが、これでは紙面を使い過ぎるので

$$a = [q_1,\ q_2,\ q_3,\ q_4,\ \ldots]$$

といった表し方をすることが多い。連分数の各分子が常に 1 だから可能な記述なのだ。

ところで (2) は (1) の解だから $x = (1 + \sqrt{5})/2$ だ。つまり

$$\frac{1 + \sqrt{5}}{2} = [1,\ 1,\ 1,\ 1,\ \ldots]$$

なんだね。右辺の連分数には 1 しか使われていないので、連分数の中でも特別な連分数だ。

そもそも連分数が特別なんだから、黄金比はまさに究極の連分数と言えるだろう。でも、ちょっと待って。連分数って本当に特別な分数なの？　次の例を見てほしい。

$$\frac{355}{113} = 3 + \cfrac{1}{7 + \cfrac{1}{16}} \tag{3}$$

有限個の分数で終わっているけれど、これも正真正銘の連分数だ。

EX.　例が正しいことを確認せよ。

実はどんな分数も連分数で表せる。いや、無理数だって連分数になるのだ。この豆旅では深入りはしないが、有理数は有限連分数、無理数は無限連分数ということが知られているのだ。ちょっと前の豆旅で、割り切れない分数は循環小数—つまりは規則的な繰り返しをする数—であることを確認したはずだ。それに対して無理数は、不規則な小数だっただろう。でも驚いてはいけない。不規則なのは見せ掛けなのだ。

EX.　計算は少々面倒だが、$[1, 2, 2, 2, 2, 2, 2, \ldots]$ や $[1, 1, 2, 1, 2, 1, 2, 1, 2, \ldots]$ といった、規則的な連分数がどんな小数になるか計算してみよ。もちろん “. . .” の部分すべては計算できないので、“. . .” の直前で終わっている連分数として計算せよ。

おそらくどこかで目にした近似値になったのではないだろうか。もし、“. . .” をすべて計算できれば、まさにその数になる。何とも不思議なものだが、詳しくは別の機会に回そう。ここでは、有理数を有限連分数に直すプログラムに取り組むことにする。

programming list [ContFrac.java]

```
import java.util.Scanner;

public class ContFrac {

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        System.out.print("input 'a b' as a/b: ");
        int a = s.nextInt();
        int b = s.nextInt();

```

```
12:         System.out.print(a + " / " + b + " = [ " + a / b);
13:         while(a % b > 0) {
14:             int t = b;
15:             b = a % b;
16:             a = t;
17:             System.out.print(", " + a / b);
18:         }
19:         System.out.println(" ]");
20:     }
21: }
```

プログラムの説明の前に、(3) の求め方を知る必要がある。コンピュータは自ら連分数を作ってくれないのだ。あくまでも人がやる作業と同じことを、短時間でやってのけるだけなのだから。

(3) は次のようにして求められた。

$$
\begin{aligned}
\frac{355}{113} &= 3 + \frac{16}{113} \\
&= 3 + \frac{1}{\frac{113}{16}} \\
&= 3 + \frac{1}{7 + \frac{1}{16}}
\end{aligned}
$$

いちいち説明の必要はないだろう。ここでは最後に 1/16 が現れて終了となったが、かりに 3/16 にでもなろうものなら、さらに $3/16 = 1/(16/3)$ としていけばよい。最終的に分子が 1 になったところで終了だ。[ContFrac.java] では、分子が 1 であることの確認を、分数の逆数が割り切れるかどうかでしているけれど、意味するところは変わらない。

このプログラムは分子・分母の入力を要求する。そのための "御用聞き" のコードは以前目にしているね。8:–10:行目がそれだ。`nextInt()` メソッドは 2 個の引数を順番に読み込んでくれる。

12:行目の `System.out.print;` 文は、9:, 10:行目で取り込んだ最初の分数を表示するだけのものだ。最終的には、連分数を "$a/b = [q_1, q_2, q_3]$" のように表示したいので、ここで "$a/b = [q_1$" まで—すなわち a/b の整数部分まで—を表示している。

13:–18:行目にかけては、残りの分数の分子・分母を交換し、次の段階の連分数に展開している。次の段階で分子・分母を交換する必要があるかないかは、分数の逆数が割り切れるかどうかで判断するので、`while(a % b > 0)` を使っている。

14:行目からは少しややこしいので注意深く読んでほしい。

まず、14:行目で分母だった数 `b` を一時変数 `t` に保存している。これは必要なことだ。

15:行目は a を b で割った余りを b に代入しているのだが、余りは分子である a に代入するんじゃないの？　そう、たしかにそうだが、さらに次の連分数展開ではこの逆数が使われるので、余りは分母にされてしまうはずだ。だから b に代入したのだ。

したがって、いままで分母に使われていた b が新しい分子 a になるから、16:行目で分子・分母の入れ換えが完了する。プログラムはわずか 3 行で連分数展開をこなしているように見える。しかし、人が行う作業が凝集されていることを見逃してはいけない。

このとき連分数展開に現れる次の数は a/b の整数部分である。よって 17:行目の計算結果を表示することになる。

最後に 19:行目の "]" で締めくくれば、はじめの分数が連分数で表記できたことになるだろう。

TRY!　いろいろな分数の連分数表記を調べてみよ。

TRY!　[ContFrac.java] は無理数を連分数にできないが、たとえば黄金比 $(1+\sqrt{5})/2$ なら近似分数として 16180339/10000000 を使ってもよい。試しに、この近似分数を連分数にしてみよ。また、14142135/10000000 や 17320508/10000000 なども試してみるとよいだろう。