第5週の庭いじり

5.1 黄金比

さて、庭いじりも 5 週目に突入だ。地面から思わぬものが出てくるかもしれないよ。

始めにフィボナッチ数列に再登場してもらおう。

$$1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ 89,\ 144,\ 233,\ 377,\ \ldots$$

そして、フィボナッチ数列の 2 項の比

$$\frac{1}{1},\ \frac{2}{1},\ \frac{3}{2},\ \frac{5}{3},\ \frac{8}{5},\ \frac{13}{8},\ \frac{21}{13},\ \frac{34}{21},\ \cdots$$

を計算してみよう。これは **Haskell** がすぐにやってくれる。以前と同じように、対話モードで関数を定義して計算させれば十分だ。

(ghci env.)

```
Prelude> let f n = if n > 1 then f (n-1) + f (n-2) else 1
Prelude> [f (n-1) / f (n-2) | n <- [1..20]]
[1.0,1.0,2.0,1.5,1.6666666666666667,1.6,1.625,1.6153846153846154,1.6190476
19047619,1.6176470588235294,1.6181818181818182,1.6179775280898876,1.618055
5555555556,1.6180257510729614,1.6180371352785146,1.618032786885246,1.61803
4447821682,1.6180338134001253,1.618034055727554,1.6180339631667064]
```

なんだか乱雑な数が並んでしまったけど、先頭から数個を除けば、どれも `1.6` ほどの値だ。おっと、その前に `f (n-1) / f (n-2)` の計算は 2 項間における $\frac{(\text{後の項})}{(\text{前の項})}$ のである。しかし、比なんだから別に $\frac{(\text{前の項})}{(\text{後の項})}$ で求めたっていいだろうに。$\frac{(\text{前の項})}{(\text{後の項})}$ として計算をやり直してもらおう。

(ghci env.)

```
Prelude> [f (n-2) / f (n-1) | n <- [1..20]]
[1.0,1.0,0.5,0.6666666666666666,0.6,0.625,0.6153846153846154,0.61904761904
76191,0.6176470588235294,0.6181818181818182,0.6179775280898876,0.618055555
5555556,0.6180257510729614,0.6180371352785146,0.6180327868852459,0.6180344
478216818,0.6180338134001252,0.6180340557275542,0.6180339631667066,0.61803
39985218034]
```

状況は似ているが、今度は先頭から数個を除けば、どれも 0.6 ほどの値だ。よーく見ると、各値は末尾に多少の違いがあるようだが、最初に計算した比の値と較べて大体 1 の差があるね。この誤差は **Haskell** の計算誤差なのか、それとも本当に微妙に差があるのだろうか。実は正しい比の値の差は 1 で、まったく誤差はないのだ。この面白い性質を持つ比の値は**黄金比**と呼ばれている。一般には、$1.618033\cdots$ の方を黄金比と呼ぶことが多い。

黄金比とは一体どういう数なのだろうか。計算は簡単にできるので確かめてみよう。

まず、$1.618033\cdots$ に収束する真の値を x としよう。あとから求めた比 $0.618033\cdots$ は、x の逆比 $\dfrac{1}{x}$ のことである。それが x より 1 小さい値に等しいのだから

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

が成り立つ。等式は簡単な 2 次方程式となる。両辺に x を掛けて整理すると $x^2 - x - 1 = 0$ だから、解の公式を使えば一発で解ける。フィボナッチ数の比は正の値なので、$x > 0$ の解を求めると

$$x = \frac{1 + \sqrt{5}}{2}$$

であることがわかる。黄金比は**無理数** $\sqrt{5}$ を含む数なのだ。庭いじりも 5 週目に入ったので、それにふさわしい数の登場だ。解は正の数だけに限っているものの、方程式からひとつの解しか得られないということは、この性質を満たす数は黄金比だけということである。

Haskell でシミュレートしてみよう。ここでは $\dfrac{1}{x} = x - 1$ を移項して

$$x = \frac{1}{x} + 1 \tag{5.1}$$

で x を探ることにする。そのほうが x の値を直接計算できるからだ。ところで、探るという言葉が気になるね。なぜそのような言葉遣いをしたかはスクリプトを見てもらいたい。

(ghci env.)

```
Prelude> let g n = if n > 1 then 1/g (n-1) + 1 else 1
Prelude> g 20
1.6180339985218035
Prelude> g 100
1.618033988749895
```

スクリプトは対話モードで $x = \dfrac{1}{x} + 1$ に対応する式を、再帰を用いてちょいと書いただけだ。`else` 節が `1` であることから、初期値として $x = 1$ が与えられている。`g 20` で 20 回程度計算しても、`g 100` で 100 回程度計算しても大差ないようだ。どうやら黄金比に収束するように見える。

さて、スクリプトを見て、あれ？　**Haskell** が 2 次方程式を解くんじゃないんだ、と感じただろう。そう、コンピュータは計算をする機械であって、問題を解く機械ではない。問題を解く手順は

人間が与えるのだ。では、この手順は何をしているのだろう。どう見ても $\frac{1}{x} = x - 1$ と同等な—つまり両辺に x を掛けた—方程式

$$1 = x^2 - x$$

を解いているのではないね。でも、解である黄金比が求められている。

それなら $1 = x^2 - x$ を移項した式、$x = x^2 - 1$ を使っても同じことだろう。計算式の定義を `g n = g (n-1) * g (n-1) - 1` に変えてみる。`*` が何と何の掛け算なのかは分かってるね。演算子より関数の結合の方が優先されるはずだったから、この式は `g n = (g (n-1)) * (g (n-1)) - 1` と同等だよ。さあ、準備は整った。関数を走らせよう。

(ghci env.)

```
Prelude> let g n = if n > 1 then g (n-1) * g (n-1) - 1 else 1
Prelude> g 20
0
Prelude> g 21
-1
```

あれ？　0？-1？　そりゃ、そうだ。$x^2 - 1$ に 1 を代入して 0、その 0 を代入して -1、その -1 を代入して 0、...だからね。`else` 節がだめなんだ。`else 2` ぐらいに直して仕切り直しじゃ。

(ghci env.)

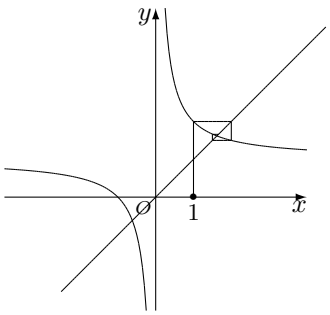
```
Prelude> let g n = if n > 1 then g (n-1) * g (n-1) - 1 else 2
Prelude> g 20
99041466760369954044427685932862489508781487688562513893320343819689510782
53796496171092896204287749283238857037808951820815674642678881510362463739
15636953973312352766372789017153205542513361903279906043122219476004512654
60442883907418839004677997414550293560155508846528321744640901134913334498
28742920504296364941188676495189175254433827715708241914001579564159781655
13207539676495418845280123755859633124041303938610450928325140375461485009
1140543940116958..............
```

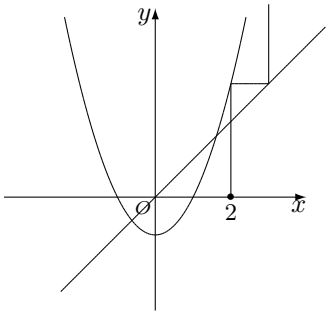
うわー、やばっ！　だけど落ち着いて。巨大数が表示されても ctrl-c で止まるはずだから。でも `g 20` 程度なら、おろおろしてる間に最後まで表示してしまうかもしれない。そんなわけで、間違っても `g 100` なんて試さないように。

どうして？　同じ関係の方程式を使ったはずなのに。その秘密はグラフを描けば見えてくる。はじめのスクリプトに使ったのは $x = \frac{1}{x} + 1$ で、次のスクリプトに使ったのは $x = x^2 - 1$ だから、それぞれ

- $y = x$　と　$y = \frac{1}{x} + 1$
- $y = x$　と　$y = x^2 - 1$

のグラフの交点を求めていることになる。ただし、スクリプトは一気に交点を求めているのではない。適当な x から始めて、いわば 2 つのグラフを渡り歩くような処理がスクリプトのしていることだ。そういうことなら実際にグラフを描いてみるのがよい。





x 軸上の点から始めて、2 つのグラフを縦横縦横... 交互に進んでみよう。$y = x$ と $y = \dfrac{1}{x} + 1$ のグラフでは、$x = 1$、つまり $(1,\ 0)$ で始まった値が $\dfrac{1}{x} + 1$ によって 2、すなわち双曲線上の点 $(1,\ 2)$ になる。次は $\dfrac{1}{x} + 1$ に $x = 2$ を代入することになるのだが、その値は $x = 2$ の位置の双曲線上の点を探せばよく、図形的には $y = x$ 上の $x = 2$ に対応する双曲線上の点である。このようなたどり方は、縦横の移動によって $y = \dfrac{1}{x} + 1$ と $y = x$ を渡り歩くことに相当する。すると、その軌跡は $y = \dfrac{1}{x} + 1$ と $y = x$ の交点に向かって進むことになるだろう。だから、収束している。

一方、$y = x$ と $y = x^2 - 1$ のグラフで同じことをすると、$(2,\ 0)$ で始まった値は、$y = x^2 - 1$ によって 3、すなわち放物線上の点 $(2,\ 3)$ になる。次は直線上の $(3,\ 3)$ を経て放物線上の $(3,\ 8)$ へ行く。こうなると、もう誰にも進行を止められない。値は

$$2^2 - 1 = 3 \quad \to \quad 3^2 - 1 = 8 \quad \to \quad 8^2 - 1 = 63 \quad \to \quad 63^2 - 1 = 3968 \quad \to \quad \cdots$$

のように爆発的に大きくなる。これが 20 回の計算のあとで表示された巨大数の正体である。

でも、4 回の計算で 3968 なんだから、あと 16 回の計算であそこまで大きな数になるはずがないと思うかもしれない。$3968 \approx 4 \times 10^3$ として、この先の計算を考えてみよう。あと 16 回というのは

$$(\cdots(((4 \times 10^3 \overbrace{)^2)^2)^2 \cdots)^2}^{16\text{ 個}}$$

ということだ。これは

$$(4 \times 10^3)^{2^{16}} = (4 \times 10^3)^{65536} = 4^{65536} \times 10^{196608}$$

だから 10^{196608} の部分だけでも 20 万桁近い。20 万桁の数ってのは、たとえば 80 桁 25 行の画面で見たら 100 ページ分だよ。ひどいことになったのも当然だ。