二と三

竹内郁雄 [1a)]

概要　諸事多忙の中，「下呂でしかできないプログラミング」に挑戦したことに関する簡単な報告である．下呂に着く直前に 2 と 3 にまつわる問題を 1 つ作成したが，それは別の参加者によって，あっという間に攻め落とされてしまった．そこで，筆者はかねてよりプログラムを書いて確かめてみたかった，やはり 2 と 3 が関係する 2 つの問題に挑戦し，時間内に 1 個だけは解けた．その問題の一番重要な部分は，平方数を除いた 2 から始まる自然数列の第 n 項を n の関数として表現することである．そこで意外な「美しい」式を発見することができた．

キーワード　2 と 3 に関係した諸問題，下呂数，$f(f(f(f(x)))) = x^2$

1. はじめに

2014 年度の冬のプロシンで，次の（2015 年度の）夏のプロシンで「ハッカソン」的なことをやるという企画が話題になったとき，それは「『プロソン』という言うべきだね」と口を滑べらしたおかげで，夏のプロシンの幹事になったものの，実際の準備を始める段になって，別の仕事がまるでバケツ雨のように降ってきてプロシンと別世界の，どっちかというと冥界，ないしは暗黒界に近いところでバタバタしてしまった．

そして，9 月 4 日のプロシン当日，新幹線で下呂に向かうものの，まだ冥界のメールが追いかけてくる．「ああ，困った，なにを喋ろう」と思いつつ，名古屋で高山本線に乗り換える．唯一の救いは 3 つのお題の 1 つが「下呂でしかできないプログラミング」という若干意味不明のもの．これだったら，TPO は万全だ．あとは中身だけ……，これが一番重い．

高山本線は特急．かなり空いていて，しかも飛騨川の眺めが美しい．8 月末の長雨が続いたあとだから，日本三大急流と呼ばれる飛騨川には勢いがある．この勢いで夏のプロソンを乗り切らねば…．特急で下呂の一つ前の駅は飛騨金山である．泥棒はもう忍び込んできている．早く縄を結わなければならない． ここで，2 と 3 にまつわる新しい問題を作ってマッチポンプでプログラムを作ろうと思い立った．それが，会場で「下呂数」と呼ばれるようになった問題である．

2. 2 と 3 から任意の数を作る

飛騨金山から，車窓の緑の谷間のシャワーを浴びながら思いついたのは，2 と 3 から任意の数を作るという問題である．これは他の参加者から詳しく解説されると思うが，簡単に紹介しておく．

0 から出発して，

(1) 2 を足す
(2) 3 を足す
(3) 2 倍する
(4) 3 倍する

という演算を適当に選んで繰り返し，与えられた 2 以上の整数を作り出せという問題である．しかし，与えられた数に対しては，暗算するだけでも何通りものやり方がある．だから，その中で最短手数のものを探せという問題に変更したのであった．

3. スライドの準備と所信表明

3.1 次元特異性？

2 泊 3 日（これも 2 と 3 だ）の夏の「プロソン」の初日は，各自の所信表明の場である．しょうがないのでその場でスライドを書き始めた．そのタイトルが「二と三」である．

私はどういうわけか数学科出身なので，数学では 2 と 3 が特異な数であることはよく聞かされていた．最小の素数の 2 個であるというだけで，2 と 3 が十分特別な数であることはよく分かるが，2 と 3 については「次元特異性」ということが言われていたような記憶がある．位相幾何学では 2 次元と 3 次元に限って特別に難しい問題が揃っているというのである．地図を 4 色で塗りわけられるという有名な「4 色問題」は 2 次元の平面地図だけが特別に難しくて，球面になったり，いろいろな意味での多次元色分け問題に拡張したら，それほど難しくなく解けるという [1]．

[1] 東京大学名誉教授
[a)] nue@nue.org

いまだに未解決のポアンカレ予想は 3 次元球面に関するものである．2 次元でも 4 次元以上でももう証明は終わっている．なぜ，3 次元だけがこんなに難しいのか？

これについては，人間が 2 次元と 3 次元の空間で主に世界を認知しているからという説がある．その次元で見つかる問題は，そこでは難しいが，多次元に拡張すると途端に簡単になるというわけだ．もし，7 次元空間に生活している高等生物がいたら，7 次元固有の特異性に悩んでいるかもしれない．

3.2 Collatz の問題と謎の算術生命体ナミーバ

私が学生のころ，故角谷静夫教授が特別講義で，いわゆる $3n+1$ 問題（一般にはいま Collatz 予想と呼ばれるが，角谷の問題と呼ばれることもある）を紹介してくれた．2 以上の任意の数 n から出発して，

(1) n が偶数なら，$n \leftarrow n/2$
(2) n が奇数なら，$n \leftarrow 3n+1$

を繰り返す．するといつかは $n=1$ となるという予想である．この講義を聞いていた故米田信夫教授は，ずーっと（夜どおし）それを計算機で検証するという作業をしておられた．当時の計算機の能力ではたいしたところまではいかなかったが，現在は 5×2^{60} までは反例がないことが確かめられているそうだ．いまどきの計算機は 64 ビットだから簡単に行けそうだが，途中の結果が簡単に 2^{64} を超えるからちゃんとその備えをしておかないといけない．

こんな小学生にも分かるような問題がどうして世界中の数学者によって解けないのか，と疑問に思われるかもしれないが，事情通の方の解説によれば，これを解いても「数学的には面白いことがない」そうなのである．フェルマーの最終定理は，$n>2$ だと，

$$x^n + y^n = z^n$$

となる $x,\ y,\ z$ の整数解はないという定理であるが，1994 年に Andrew Wiles 教授によって証明された．定理の発表から 360 年も経って証明されたことになるが，その過程でいろいろな新しい数学的な発見があったという．しかし，$3n+1$ 問題は，それが解けても，数学の発展に寄与するような何かが得られる見込みがないというのである．よく分からないが，逆にそれが 2 と 3 がもたらす特異性なのかもしれない．

特別講義では角谷先生が「$(3/2)^n \bmod 1\ (n>0)$ は区間 $[0,1]$ で稠密になる」というお話もされたと記憶しているが，あまり自信はない．別に 3/2 でなくてもいいような気もするが，こちらもあまり自信はない．

あるとき，2 と 3 以外に $3n+1$ 問題みたいな面白いことが起こらないかと考え，5 とか 7 とか 11 とかいろいろ素数を動員して行き当たりばったりに試したことがあったが，どれもうまくいかない．やっぱり，2 と 3 は偉大なのである．

正確な時期は忘れたが，私が電通大にいたころ，ICPC のお手伝いをしたことがある．なにか問題を作れというので，いろいろ考えていたが，馬耳東風の教授会はやはり創造力を発揮できる環境のようだ．$3n+1$ 問題をタネに ICPC 向けの問題を作れないかなぁと思い立って紙にいろいろ書いていて，ほほう，これは面白そうと思いついたのが謎の算術生命体ナミーバ（numoeba）である．これについては当時 NTT 研究所の佐藤進也君（現在日本工業大学教授）がプロシンで発表をしている [2,3]．分かりやすい説明が

`http://www.shin-ya.org/numoeba`

にある．ICPC では語のビット長の制限があるので不自然な制約をつけてしまったが，本来これはなくてよい．いずれにせよ，2 と 3 でなにやら出生・成長・波乱を経て（必ず？）死滅に至る生命体風のものが生成できたらしい．佐藤君はいろいろなデータを取ってなにか面白い法則が発見できないか苦労したようだが，そうは問屋は卸ろさなかったようだ．

3.3 竹内関数と Conway のライフゲーム

竹内関数の元の命名はタライ回し関数である．これは整数引数 $x,\ y,\ z$ に対して以下のように定義される関数である．

$$\begin{aligned} &t(x,y,z) = \\ &\quad \textbf{if } x \le y \textbf{ then } y \\ &\qquad \textbf{else } t(t(x-1,y,z), \\ &\qquad\qquad t(y-1,z,x), \\ &\qquad\qquad t(z-1,x,y)) \end{aligned}$$

$t(n,0,n+1)$ をまともに計算すると $O(n^n)$ の手間がかかることは Donald E. Knuth 教授が証明してくれたが，実はこの関数は以下の超簡単な関数と同値である [4]．

$$\begin{aligned} &t_0(x,y,z) = \\ &\quad \textbf{if } x \le y \textbf{ then } y \textbf{ else if } y \le z \textbf{ then } z \textbf{ else } x \end{aligned}$$

思うに，これは 2 重再帰と 3 引数の組合せなので，やはり 2 と 3 に絡んでいる．この関数の 2 と 3 の呪縛を逃れるべく，いろいろな変形・拡張が試みられたようだが，成功したという話は聞いていない．

ついでながら，John Horton Conway 教授の有名なライフゲームでも 2 と 3 が重要な役割を果たしている．いろいろな拡張が試みられているが，2 次元セルオートマトンで 2 と 3 というのが一番面白いようだ [5]．ちなみ

に私が電通大にいたころ，セルオートマトンではなく，平面空間で点と点の距離を実数で測る「リアル」ライフゲームを卒論で挑戦した学生がいたが，本当に目をしょぼしょぼさせながらパラメータ調整していたことを思い出す．

3.4 $f(f(f(f(x)))) = x^2$ 問題

これは数学セミナー誌に出題するために，エジプトに行っている間に思いついた問題である．有理数から有理数への関数 f で

$$f(f(f(f(x)))) = x^2$$

を満たし，かつ

$$x \to 0 \text{ のとき，} f(x) \to \infty$$

となるものがあるかどうか？あるならその関数を具体的に構成せよ，という問題である．

f が実数から実数への関数であれば，

$$f(x) = 1/|x|^{\sqrt[4]{2}} \ (x \neq 0)$$
$$f(x) = 0 \ (x = 0)$$

と至極簡単なのだが…．

実はこの問題に用意した私の解答は，やはり 2 と 3 が絡む，整数の 2 乗数と 3 乗数が活躍する．正解と言える解答の中でも実際にプログラムを組むと私の解答が一番「効率がいい」と思っていた．それを下呂でプログラムを書く時間があるときに確かめてみたいと発表した．

3.5 2 と 3 の演算で整数を作る問題

プレゼンの最後に下呂に到着する直前に思いついた 2 と 3 の足し算，掛け算で 2 以上の任意の整数を生成する問題（前章参照）を紹介して，これを下呂でプログラムしてみたいとも発表した．

4. 蓋を開けてみると

実は私が発表した直後から 2〜3 名の参加者が，2 と 3 の演算で整数を作る問題に挑戦し始めた．参加者の発表が終わったころには，ある程度の結果が出てしまっていた．しかもいきなり「下呂数」，え，それ何？という言葉が飛び出してしまい，もう私の出る幕はなくなってしまった．

しょうがないので，2 日目の朝から夕方までひたすらプログラミングというセッションでは，「下呂数」には手をつけず，3.4 で述べた $f(f(f(f(x)))) = x^2$ を満たす有理数から有理数への関数を実際にプログラムで書いてみることにした．ところが，私の（効率のいいはずの）解答では，整数 p を平方数でも立方数でもない q を使って $q^{2^m 3^n}$ $(m \geq 0,\ n \geq 0)$ と表わすという分解をする必要がある．しかし，書こうとすると意外と面倒ではないか．

そこで，私があまり効率的な方法がないと思っていた，数学セミナーの常連解答者 ζ 氏の方法でプログラムを書いてみることにした．ζ 氏の解答は以下の通りである．

2 以上の整数に関しては，非平方数の列 2, 3, 5, 6, 7, 8, 10, 11, ... の先頭から 2 つずつ数をとってそれぞれ数列を作る．例えば，最初の 2 個の数 2 と 3 からは以下の数列を作る．

$$2 \mapsto 1/2 \mapsto 3 \mapsto 1/3 \mapsto 2^2 \mapsto (1/2)^2 \mapsto 3^2 \mapsto (1/3)^2 \mapsto 2^4 \mapsto \ldots$$

これで 2 以上の整数とその逆数を全部尽くせる．それ以外の既約有理数 q/p については，

$$f(q/p) = p/q \quad (0 < q/p < 1)$$
$$f(q/p) = p/q^2 \quad (p^{2^n} < q < p^{2^{n+1}},\ n(\geq 0) \text{ は偶数})$$
$$f(q/p) = p^2/q \quad (p^{2^n} < q < p^{2^{n+1}},\ n(\geq 0) \text{ は奇数})$$

と定義する．なお，$f(0) = 0$，$f(1) = 1$，負の数 x については $f(x) = f(-x)$ と定義する．

この解の説明で，私はこう書いていた．「ζ 氏が意識しておられたように，これのいいところは，整数とその逆数については，対となる整数がなんであるかを調べるのにちょっとだけ面倒な計算が要るものの，計算の手間があまりかからないということです．」

整数平方根，整数立方根の計算をあきらめた私は，ζ 氏の解答が本当に面倒なのかどうか確かめることにした．まさに「下呂温泉に飛び込む」心境であった．

5. 平方数を除いた数列に関する計算

もちろん，昔取った杵柄の TAO（Lisp 方言）を使う．このプログラムを書くときに一番時間がかかったのは，1 オリジンに修正した形で表現すると，

$$2, 3, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 37, 38, ...$$

という 4, 9, 16, 25, 36 といった平方数を取り除いた数列の n 番目 $(n \geq 1)$ の数を探索的なことをせずに求める関数の作成である．

ほぼ $\sqrt{n}$ 個の平方数が抜かれているので，直感は

$$n + \sqrt{n}$$

である．しかし，答えは整数でないといけないので切り下げ関数を使わないといけない．数を扱うプログラム

なぞ久しく書いたことがないので，さてどうしたものかと思って，TAO 処理系の中をいろいろ探索していたら，TAO には，Common Lisp にならって平方根以下の最大の整数平方根を求める `isqrt` という関数があることが分かった．しかも任意多倍長．これは便利．なお，TAO に有理数のデータ型があり，例えば `12/36` と入力すると，内部では `1/3` に約分された既約形で保存される．

以下，$\sqrt{n}$ は n の整数平方根を表わすとする．残念ながら，直感の $n+\sqrt{n}$ はすぐ馬脚が出る．あまり理論的に考えるようなガッツもなかったので，いろいろ試行錯誤すること約 20〜30 分．とうとう見つかったのが

$$n+\sqrt{n+\sqrt{n}}$$

である．実はこれではだめで，整数平方根の入れ子を無限にしないといけないのかと危惧したのだが，これで大丈夫のようだ．

それにしても美しい式が見つかったものだと思う．これは下呂温泉効果であると，固く信じたい．証明はそれほど容易ではない．

プログラム全体は付録の図 1 に示した通りである．代入を `setq` ではなく `!` で書く Lisp だが，読解可能だと思う．`make-ratio` は 2 つの整数 p，q から有理数 p/q を作る関数である．なお，このプログラムは数列を 0 オリジンで表現しているので，上記の式とは微妙に違う関数になっていることに注意されたい．図 2 に実行例を示しておいた．

6. 午後の挑戦

朝 9 時過ぎから始めて，昼前には $f(f(f(f(x))))=x^2$ 問題が解決してしまった．しかも，気持のよい関数が 1 つ見つかって，ルンルン状態である．午後は，これも数学セミナーに出題していて，いつかはプログラムを書いてみたいと思っていた問題に挑戦することにした．「3 人の賢者」という問題で，3 人の賢者が自分と自分の左隣，つまり 2 人に関する情報を知っていて，賢者としての推論をするという設定なので，これも微妙に 2 と 3 に絡む問題である．

「3 人の賢者」の賢者の推論をシミュレーションしようとしたが，午後 4 時すぎになっても完全なプログラムを書くことができなかった．後日，一応完成させたつもりだが，つもりはつもりであり，本当に完全かどうかはまだ確信できていない．

これについては，半年後の冬のプロシンで発表することにした．

7. むすびに代えて

冥界から下呂に行き，久々にプログラムを書いた．美しい関数が見つかったとはいえ， 実用性とはまったくほど遠いプログラムである．任意多倍長数や有理数が最初からデータ型に揃っていたので，飛び道具満載だったが，やはりプログラミングは楽しいということを実感させられたプロソンであった．

[参考文献]

[1] 一松信: 四色問題—その誕生から解決まで（ブルーバックス 351）新書，1978.

[2] 佐藤進也, 天海良治, 竹内郁雄: ある新種の算術生命体 — ナミーバ, 情報処理学会第 42 回プログラミングシンポジウム報告集, 2001.

[3] 佐藤進也, 竹内郁雄: 制約より生じる調和 — 謎の算術生命体ナミーバの組織化, 情報処理学会夏のプログラミングシンポジウム報告集, 2002.

[4] Donald E. Knuth: Textbook Examples of Recursion, in Artificial Intelligence and Mathematical Theory of Computation, Papers in Honor of John McCarthy, (ed. Vladimir Lifischitz), pp.207-230, Academic Press, 1991.

[5] Andrew Ilachinski: Cellular Automata: a Discrete Universe, World Scientific Pub Co Inc, 2001,

[付録]

```
(set-case-sensitive)

(defun F (x &aux p q)
    (if (< x 0) (!!neg !x)) ; 負なら正へ変換
    (if (= x 0) (exit 0)) ; if x = 1, then (make-ratio 1 1) → 1
    (if (integerp x) (exit (make-ratio 1 x))) ; 整数なら単純に 1/x
    (!q (numerator x)) ; p/q に分解
    (!p (denominator x))
    (if (= q 1) (exit (pairF p))) ; 1/p なら対の数（の巾）
    (cond ((< x 1) (/ 1 x)) ; ζ 氏の方法に準拠
          ((evenp (p^<2^<n<q<p^2^n+1 p q)) (make-ratio p (* q q)))
          (t (make-ratio (* p p) q)) ))

; F で対のほうへ移る
(defun pairF (x &aux sr srth (power 1))
    (!sr (isqrt x))
    (if (= x (* sr sr)) ; 平方数だったら，祖先を探す
        (multiple-value-setq (srth power) (get-root sr 2))
        (!srth (- x sr 1)) ) ; 0 オリジン
    (if (evenp srth)
        ; 対は 1 大きい数
        (expt (get-nth-root (1+ srth)) power)
        ; 対は 1 小さい数
        (expt (get-nth-root (1- srth)) power) ))

; 平方数を祖先まで辿る（答えは祖先，巾 2^n の n の 2 値）
(defun get-root (x n &aux sr)
    (!sr (isqrt x))
    (if (= x (* sr sr)) (get-root sr (* 2 n)) (values (- x sr 1) n)) )

; 平方数を除いた n 番目の祖先，isqrt が二重！
(defun get-nth-root (n &aux x)
    (inc n)
    (+ n (isqrt (+ n (isqrt n)))) )

; p^2^n < q < p^2^(n+1) となる n を求める
(defun p^2^n<q<p^2^n+1 (p q &aux (pp p) (n 0))
    (loop (:while (< pp q) (1- n))
          (inc n)
          (!pp (* pp pp)) ))
```

図 1. $f(f(f(f(x)))) = x^2$ **となる** f **のプログラム**

```
◆ (!qwe 2387874732108473289219873473822810 5/
        3738472910238472828473281)
2387874732108473289219873473822810 5/
3738472910238472828473281

◆ (F qwe)
3738472910238472828473281/
5701945736242113076861894175231283595135747660808379134188750118910 25

◆ (F (F (F (F qwe))))
5701945736242113076861894175231283595135747660808379134188750118910 25/
1397617970058691651809374464455546690124133090496 1
```

図 2. f **の実行例**

見やすくするために，/ のところで折り返してある．