新しい定理の作り方

コンピュータを利用して強引に新しい定理を作る

白石和夫（文教大学教育学部）

コンピュータを探求の道具として利用する数学教育を推進するためには，さまざまな探求テーマを用意しておくことが欠かせない。この教材は，コンピュータを利用する探求活動の課題の一例となるものである。

1. がんばりＭＡＴＨ（仮称）の定理

1.1. 定理

筑波大院生の自主ゼミグループ「がんばり MATH（仮称）」は，彼らのホームページ上で新しい定理を発表している*。その定理は，

自然数の各桁を二乗した数の和を考える。この操作を繰り返すと、１または４になる。

特に４になる場合は次のループに入る。

４→１６→３７→５８→８９→１４５→４２→２０→４→・・・

というものである。

1.2. 定理の検証

コンピュータのプログラミング能力を仮定すれば，この定理の成立を検証することは容易である。2 桁の自然数について各桁の数の 2 乗の和を求めるとその最大値は 3 桁になるが，3 桁の数について同様のことを考えると，最大値は 3 桁にしかならない。実際，92×12＝972 であるから，12 桁以下の自然数について，各桁の数の 2 乗の和を求めるとその値は 3 桁以下になるということができる。したがって，3 桁以下のすべての自然数について定理を検証すれば，12 桁以下の自然数について定理が成立することになる。12 桁を超える自然数についても，各桁の数の 2 乗の和を求める操作を何回か繰り返せば 12 桁以下の自然数にできるから，3 桁以下のすべての自然数について定理を検証すれば，定理の証明ができたことになる。

1.3. プログラミングの技巧

定理が正しいことを前提にすれば，1 から 999 までの整数について，各桁の数の 2 乗の和をとる操作をその結果が 1 または 4 になるまで繰り返し，プログラムが終了することを確認すればよい。しかし，正しいとは断定されていない命題の検証のためのプログラミングとしては正しい方法とはいえない。命題が正しくなくて，１または 4 にならない場合には永遠に終わらないプログラムになってしまうからである。また，それでは，新たな定理を見つけ出す手法になりえない。

繰り返しが起こることが命題の要点であるから，各桁の数の 2 乗の和をとる操作を繰り返し行い，同じ数が出て来たら止まるようにプログラムするのが正しい方針である。3 桁以下の自然数について各桁の数の 2 乗の和をとる操作の結果は必ず 3 桁以下になるのであるから，この操作で無限に新しい数が出続けることはない。つまり，必ず，いつかは同じ数が出現することは保証される。

---

* http://www.first.tsukuba.ac.jp/~subaru/gambarim/index.html

既に出現した数を記憶させるのに配列を用いるのが一応自然な発想であろう。次に示すプログラムでは，添字付き変数 c(n)を単なるフラグとして用い，通過したら印をつけ，2 度目にその数に到達した場合にはそれ以降の計算は無意味なのでそこで停止するようにしている。

```
100 DIM c(243)
110 MAT c=ZER
120 FOR i=1 TO 243
130     LET n=i
140     PRINT n;
150     DO
160         LET c(n)=1
170         LET p0=MOD(n,10)
180         LET p1=MOD((n-p0)/10,10)
190         LET p2=INT(n/100)
200         LET n=p0^2 + p1^2 + p2^2
210         PRINT n;
220         IF c(n)>0 THEN EXIT DO
230     LOOP
240     PRINT
250 NEXT i
260 END
```

2. 新しい定理

2.1. 新しい問題

がんばり MATH の定理で各桁の数を２乗するところを３乗に変えてみたらどうなるだろうか。つまり，各桁の数の３乗の和をとる操作を繰り返したらどうなるかということである。

４桁の整数について１回上述の操作を行うと，その結果は 2916 以下になるから，2916 以下の自然数について調べてみればいい。

2.2. プログラムの技巧

まず，1 から 2916 までの添字が使える配列 a を用意して，a(*n*)に *n* の各桁の数の 3 乗の和をあらかじめ計算して代入しておく。こうすることで，無駄な計算が省ける。

もっとも面倒なのがループの検出である。2 乗の和の問題の場合には，2 系統のループが存在した。3 乗の和の場合には，おそらく，それ以上の数のループが存在するに違いない。ループを効率よく見つけ出す工夫が必要だ。

1 から 2916 までの添字が使える配列 c を用意し，ある数 n から始めて次々に次の数を取る操作を行い，出現した数 k について c(k)に n を代入することにする。そうすれば，代入する前に c(k)が n であることを調べるようにすれば，ループになったかどうかが判定できる。

ループが検出されたら，そのうちの最小数を検出しておこう。そうすれば，その数をループを識別する目的に使える。ループ中の最小数を見つけるのは簡単で，出発点を記憶した上でループをもう一度回ってくればよい。

上述の操作を 1 から 2916 までのすべての整数について実行するのは無駄が多い。次の数をたどる操作をしていって，既に調べた数に達したらこの操作は打ち切ってしまってよい。そのとき，その判断のために添字付き変数 c(k)の値が利用できる。n から操作を始めたとき，c(k)の値で 0 でもなく，n でもなければ既に調査済みのルートに達したということである。

2.3. 実際のプログラム

実際に作成したプログラムは次のとおりである。注釈に日本語を用いていること以外は，日本工業規格 Full BASIC に準拠している。日本語の注釈を削除すれば，JIS に従う Full BASIC 処理系であればどれを用いても動作し，同じ結果が得られることが保証される。

```
100 DIM a(32805),b(32805),c(32805)
110 REM n の各桁の数の 3 乗の和を a(n)に代入
120 FOR n=1 TO 32805
130     LET t=n
140     LET p1=MOD(t,10)
150     LET t=(t-p1)/10
160     LET p2=MOD(t,10)
170     LET t=(t-p2)/10
180     LET p3=MOD(t,10)
190     LET t=(t-p3)/10
200     LET p4=MOD(t,10)
210     LET t=(t-p4)/10
220     LET p5=MOD(t,10)
230     LET t=p1^4 + p2^4 + p3^4 + p4^4 + p5^4
240     LET a(n)=t
250 NEXT n
260 MAT b=ZER
270 MAT c=ZER
280 REM ループ確定
290 FOR n=1 TO 32805
300     LET t=n
310     REM ループ検出
320     DO WHILE c(t)=0
330        LET c(t)=n
340        LET  t=a(t)
350     LOOP
360     IF c(t)=n THEN
370        REM ループ内最小数検出
380        LET tini=t
390        LET tmin=t
400        DO
410           LET t=a(t)
420           IF t<tmin THEN LET tmin=t
430        LOOP UNTIL t=tini
440        LET b(tmin)=1
450     END IF
460 NEXT n
470 REM 各ループ内最小数の表示
480 FOR n=1 TO 32805
490     IF b(n)=1 THEN PRINT n;
500 NEXT n
510 PRINT
520 PRINT
530 REM 全変移の出力
```

```
540 FOR n=1 TO 32805
550     PRINT n;
560     LET t=n
570     DO UNTIL b(t)=1
580         LET t=a(t)
590         PRINT t;
600     LOOP
610     PRINT
620 NEXT n
630 END
```

このプログラムは，480 行～500 行の FOR～NEXT を実行すると，

1　55　136　153　160　370　371　407　919

を表示する。この結果から，自然数の各桁の数の 3 乗の和を取る操作を繰り返すと，ここに示された 9 個の数のいずれかになるという定理が成立することがわかる。

540 行以降は，1 から 32805 までの各自然数について，どんなルートをたどって上記のいずれかになるかを実際に出力している。計算という観点からすると，ほとんど何もしていないに等しいが，(仮称)十進 BASIC の Windows 版で実行するとこの部分の実行で最も多くの時間を消費する。十進 BASIC はテキストの表示に Windows の標準機能を利用しているためである。

2.4. さらに新しい定理

2 乗の和，3 乗の和を考えるかわりに 4 乗の和を考えたらどうなるだろうか。$9^4=6561$ であり，$6561\times5=32805$ であるから，5 桁の自然数について各桁の数の和の 4 乗の和を取る操作を行うとその結果は 32805 以下になる。5 桁を超える数は，各桁の数の和の 4 乗の和を取る操作を何回か行えば 5 桁以下の数になるから，32805 以下の数について調べればよいだろう。

上に示したプログラムを修正して調べてみると，この場合には，1，1138，1634，2178，8208，9474 のいずれかになることがわかる。

2.5. 発展的探求課題

5 乗の和，6 乗の和，・・・と考えていくことは自然な発想として出てくるであろう。その場合でも必ず何らかの結果が出ることは保証されている。また，桁の数を考えるということは記数法に十進法を用いることを暗黙に仮定している。記数法における基底を変更すれば，また，問題のバリエーションを作ることができる。

このような単純な発展ばかりでなく，より探求的な課題を見つけることもできる。たとえば，3 乗の和の問題の場合，370 を含むループと 371 を含むループは，実は長さが 1 である。つまり，$3^3+7^3=370$ となっている。ループの長さも研究の対象となりうる。

3. この課題の教育的価値

ここで扱った課題は数学的には必ず結果が得られることが保証された問題である。数学者には興味のない問題であろう。しかし，ちょっとコンピュータを動かしてみれば，もしかしたら，まだ，誰も論文に発表していないかも知れない結果が得られるのである。学生・生徒のための探求課題としては非常に価値の高いものだといえるだろう。