ピタゴラス数(2つの平方和で表す)-その2
Jでシルバーマンの数論を読む

西川 利男

先の例会で、Jで(5,4,3)や(13,12,5)などピタゴラス数を列挙するプログラムを報告した[1]。しかし、どんな数ならピタゴラス数になるか、つまり2つの平方和に分解できるか？という問題は整数論のいろいろな理解の末に解決できるテーマである。

このようなときに、前回あげた参考書のうち、「シルバーマンの数論」[2]はこの問題を非常にていねいに解説しており、このテーマが整数論を学び、楽しむにも、まさにうってつけの課題であることが分かった。

さらに、われわれにとっての武器であるJの環境は、この目的には極めて強力であり、まさに最適である。これから上のようなタイトルにした次第である。

１．シルバーマンの攻略法

シルバーマンは最初の章で、次のような整数論の問題解決の攻略法を強調している。まず、何を求めたいかという問題の目標をはっきりとかかげた上で

・具体的に数値を計算して、データを集める。
・そのデータから値のパターンや関係の予想を立てる。
・別の新しい値に対して、その予想が正しいことを確かめる。
・最後にその予想が正しい根拠を示す。これが証明である。

これまで多くの整数論の教科書では、最初の計算する、データを集めるなどのステップを踏むことなく、最後の証明だけを書いている本がほとんどである。

２．ピタゴラス数をどう攻略するか？

いまの目標と攻略法を次のように定める。また、今回は斜辺が与えられたときのピタゴラス数を問題にする。直角をはさむ一辺のときは違う展開になるが、別の機会にゆずる。

目標：どんな数が２つの平方和で表わされるか、そしてどう表わされるか

ステップ１：1, 2, … と具体的な数で平方和の可否を調べ、データを集める。

ステップ２：平方和への予想のパターンは次のようになるだろう。

(1) 素数と合成数とでようすが違う。
(2) 素数の場合は
  (2a) 4を法として1に合同(4n+1型)な数は２つの平方和で表わせる。
  (2b) 4を法として3に合同(4n+3型)な数は平方和では表わせない。
(3) 合成数の場合は何ともいえない。
  (3a) しかし、どんな合成数でも素数の積で表わせる。
  (3b) したがって、それぞれの素数が(4n+1型)素数のときは
    (2a)によって平方和で表わせる。
  (3c) ところで、一般に次の恒等式が成り立つ。

$$(u^2+v^2)(A^2+B^2)=(uA+vB)^2+(vA-uB)^2$$

すなわち、この式では２組の平方和の積は１組の平方和に変換できる。
今回のテーマでは、さまざまところで活躍する重要な式である。

(3d) これを使って、平方和を２組ずつまとめて変換し１組の平方和にする。
これをくりかえせば、結局最後は１組の平方和になる。

ステップ３：

(1) 奇数の素数 p が２つの平方の和ならば（4n+1 型)の数、すなわち
p は 4 を法として 1 に合同である。
式で表すと　p ≡ 1 (mod 4)　となる。

(2) (4n+1 型)の素数を２つの平方和に分解するには
無限降下法（後述）により計算で求められる。

３．どんな数がピタゴラス数になるか？

まず、１から 50 までの整数につきピタゴラス数、つまり平方和で表わされるかどうか、チェックしてみる。これには J は極めて強力な助っ人になる。

例えば、$10 = 3^2 + 1^2$ は OK だが、11 は NO である。これをチェックするには

$11 - 1^2 = 10$、$11 - 2^2 = 7$、 $11 - 3^2 = 2$

で得られる数が２乗数になるかどうか見ればよい。

Ｊでやってみよう。まず、テストするには平方根までの整数でよいから、これを得るには sqx で行う。

```
   sqx =: 3 :'%: y. - *: >: i. < %: y.'
```

11 の場合には

```
   sqx 11 => 1 2 3
```

これを使って、平方和の可否を 1 (=OK)、0 (=NO)のフラグとして求める関数を作る。

```
   sqsum =: 3 : 0
   SY =. sqx y.
   ST =. =/"(1) SY ,. <. SY
   SS =. ST # <. SY
   0 < #SS
   )
```

これによれば

```
   sqsum 10 => 1、sqsum 11 => 0
```

1 から 50 までの整数 N50 =: >: i.50 に対して、調べてみよう。

平方和で表せる数は

```
   (sqsum"(0) # ]) N50
1 2 4 5 8 9 10 13 16 17 18 20 25 26 29 32 34 36 37 40 41 45 49 50
```

平方和で表せない数は

```
   N50 -. (sqsum"(0) # ]) N50
3 6 7 11 12 14 15 19 21 22 23 24 27 28 30 31 33 35 38 39 42 43 44 46 47 48
```

しかし、これからはパターンは見出せない。そこで 1 から 50 までの素数についてやってみよう。素数を得るには J の有名なイディオムを用いる。

```
   primes =: i.&.(p:^:_1)
   P50 =: primes 50
   P50
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
   ]Q50 =: (sqsum"(0) # ]) P50
2 5 13 17 29 37 41
   ]R50 =:  P50 -. (sqsum"(0) # ]) P50
```

3 7 11 19 23 31 43 47

この結果は、整数論の重要な定理を示している。

2平方和の定理(素数の場合)

素数pが2つの平方和で表せるのは

4で割って1余る数(4n+1型)の数、式で表すと　$p \equiv 1 \pmod 4$

または

$p = 2$

に限る。

証明: 証明はいろいろな教科書にのっているが、ここでは最も易しいものを示す。

$$p = a^2 + b^2$$

pは奇数であることはわかっているので、aは奇数、bは偶数としても一般性を失わない。

$$a = 2n + 1, \quad b = 2m$$

とすると

$$p = (2n + 1)^2 + (2m)^2$$
$$= 4n^2 + 4n + 1 + 4m \equiv 1 \pmod 4$$

(証明終)

Jで計算するため、整数論でよく使うガウスの合同式に合わせて次のようにした。

```
mod =: |~
```

しかし、数学の記法では、例えば

$$13 \equiv 1 \pmod 4$$

となるところ、Jでは次のように微妙に違うので注意を要する。

```
13 mod 4 => 1
```

この平方和の定理によれば、先のsqx、sqsumのようにぼう大な計算を行うことなくピタゴラス数の可否を決めることができる。

```
NB. test prime?
qpr =: 1: = (#@q:)
NB. test pythagorean?
qpt =: 3 : '(qpr y.) *. ((1 = y. mod 4) +. (2 = y.))'
```

こんなのも一瞬である。

```
   (qpt # ]) primes 300
2 5 13 17 29 37 41 53 61 73 89 97 101 109 113 137 149 157 173 181 193 197 229
233 241 257 269 277 281 293
```

４．素数をどう平方和に分解するか？

４．１　無限降下法による平方和への分解

いよいよここで、フェルマーの無限降下法が登場する。考え方の流れは、次のようになる。くわしくはシルバーマンの本(p.165-168)に説明してある。

素数pに適当な数Mを掛けた数Mpを２つの平方和で表した方が容易なことがある。次に平方剰余の相互法則を利用し、２通りの平方和で表す。これに法Mとする合同式の演算を行い、続いて平方和の積から平方和への変換式を利用し、最後により小さい乗数を求める。この操作を乗数が1になるまで繰り返せば最終の平方和の分解となる。

一般的なアルゴリズムは込み入っているので、p = 881とした数値例で示す。

$A^2 + B^2 = M \cdot 881$

となるような数Mとして170をみつける。それには、合同式

x^2 ≡ -1 (mod 881)
を解くことが必要になる。これを簡単に解くには
b ≡ a^(p-1)/4
を試行錯誤でaを選び、当てはまるものをさがす。
(3^(881-1)%4) mod 881 => 387
つまり、A ＝ 387 で B＝1 とする。
すなわち、乗数170を掛けることで２つの平方和で表せる。
387^2 + 1^2 ＝ 170・881
この式を最初の式として降下法の操作を始める。
まず、合同式の演算より
387 ≡ 47 (mod 170)
1 ≡ 1 (mod 170)
となるので、合同式の性質より
387^2 + 1^2 ≡ 47^2 + 1^2 (mod 170)
≡ 0 (mod 170)
上の合同式は、次のようにも表せる。
47^2 + 1 ＝ 170 ・13
また最初の式は、次のようである。
387^2 + 1 ＝ 170 ・881
これら２つの式を辺々同士掛け合わせる。
(47^2 + 1^2)(387^2 + 1^2) ＝ 170^2 ・13・881
ここで、左辺は平方和の積から平方和への式で変換できる。
(47・387 + 1・1)^2 + (1・387 - 47・1)^2 ＝ 170^2 ・13・881
両辺を170^2で割ることができる。
(18190/170)^2 + (340/170)^2 ＝ 13・881
これを簡単にすると
107^2 + 2^2 ＝ 13・881
ここにおいて、２つの平方和で表す、より小さい乗数13が得られた。
続いて、上と同じ操作を
107^2 + 2^2 ＝ 13・881
を初期値として繰り返せば
25^2 + 16^2 ＝ 1・881
乗数は1となり、目的とする２つの平方和への分解がなされた。

４．２　Ｊによるプログラムの実行

Ｊでは、前半の操作はfind、後半の操作はdescとし、これを使って全体操作はdescendとしてプログラミングした。
それぞれのプログラムのコーディングの詳細は最後のプログラム・リストを参照されたい。

最初にピタゴラス数のチェックを行う。

```
   qpt 881
1

   P50 =: primes 50
```

```
   R50 =: P50 -. (qpt # ]) P50
   R50
3 7 11 19 23 31 43 47
```
まず、findにより最初の乗数Mをさがす。
```
   FIND =: 881 find"(0) R50
   R50 ,. FIND
 3 881        170 387 1
 7 881        277 494 1
11 881 774401r881 880 1
19 881 774401r881 880 1
23 881        170 387 1
31 881        277 494 1
43 881 774401r881 880 1
47 881        277 494 1
```
前節に説明した各回ごとの繰り返しの部分はdescを使って次になる。
```
   desc 881, 170, 387, 1
881 13 107 2
   desc 881, 13, 107, 2
881 1 25 16
```

以上を連続して繰り返し解を得る全体プログラムdescendでは次のようになる。
```
   3 descend 881
0 11453
1 881
decomp to sum squared:
881 = 25^2 + 16^2
*** end ***
```

左引数7でも解は得られた。
```
   7 descend 881
0 149770
1 11453
2 881
decomp to sum squared:
881 = 25^2 + 16^2
*** end ***
```

左引数11では解は得られない。これは19でも同様である。(省略)
```
   11 descend 881
0 2
1 1
2 0
3 0
decomp to sum squared:
881 = 0^2 + 0^2
```

*** end ***

また、左引数23でも解は得られた。

```
 23 descend 881
0 11453
1 881
decomp to sum squared:
881 = 25^2 + 16^2
*** end ***
```

５．一般の整数（合成数も含めて）をどう平方和に分解するか？

５．１　平方和・積から平方和の変換式の利用

整数には素数と合成数とがある。そして合成数は素数の積に分解できる。また、前節で(4n+1)型の素数は平方和に分解できることを示した。

これから、たとえ合成数であっても、素数の積に分解して、それぞれを平方和に分解し、これらを先の平方和・積から平方和の変換式を利用すれば、最終的には一般の整数でも平方和に分解できることになる。

具体的な数1105 でやってみる。

```
m ＝ 1105 ＝ 5・ 13・ 17
          ＝ (2^2 + 1^2)・(3^2 + 2^2)・(4^2 + 1^2)
          ＝ ((2・3 + 1・2)^2 + (1・3 - 2・2)^2 )・(4^2 + 1^2)
          ＝ ( 8^2 + 1^2 ) ・(4^2 + 1^2)
          ＝ (32 + 1)^2 + (4 - 8)^2
          ＝ 33^2 + 4^2
```

素数分解で偶数の巾乗が含まれる場合には、その部分を括り出して行う処理が必要になる。そのような例として、25798500 をやってみる。

```
m ＝25798500 ＝ 2^2・3^4・5^3・7^2・13
             ＝ 5・13・(2・3^2・5・7)^2
             ＝ (8^2 + 1^2)・(2・3^2・5・7)^2
             ＝ (8^2 + 1^2)・(630)^2
             ＝ (8・630)^2 + (1・630)^2
             ＝ 5040^2 + 630^2
```

５．２　Ｊによるプログラムの実行

Ｊのプログラム compo と prod2sum とは最後に挙げてある。途中経過の素数分解はMP、平方和分解はMC として示した。さらに、偶数の巾乗の括り出し処理も含めた最終プログラム compox の実行例を示す。

```
 compox 1105
33 4
 MP
5 13 17
 MC
2 1
```

```
3 2
4 1
   compox 25798500
5040 630
```

６．ピタゴラス数への分解

このテーマの始めに戻って、ピタゴラス数とは３つの組(a, b, c)で

```
a^2 + b^2  = c^2
```

を満たすものである。そして既約のピタゴラス数、gcd(a, b) ＝ 1であれば、互いに素な奇数 s > t > 1 をとって

```
a  = st, b = (s^2 - t^2)/2, c  = (s^2 + t^2)/2
```

として、得られる。

先の例 1105 では

```
2c  = 2 ・1105
    = (1^2 + 1^2)・(33^2 + 4^2)
    = 37^2 + 29^2
```

これから、

```
s  = 37, t = 29
```

とすれば

```
a  = st = 1073
b = (s^2 - t^2)/2  = 264
```

つまり、 ピタゴラス数 (1073, 264, 1105) となる。

Jによるプログラムの実行は以下のとおりである。

```
   pytha 1105
1073 264 1105
```

他のいくつかの例をやってみよう。

```
   pytha 1189
989 660 1189
   +/ *: 989 660
1413721
   *: 1189
1413721
   pytha 1885
1643 924 1885
   +/ *: 1643 924
3553225
   *: 1885
3553225
   pytha 3185
3087 784 3185
   +/ *: 3087 784
10144225
   *: 3185
10144225
```

文献

[1] 西川利男「ピタゴラス数－J701 でやってみる」JAPLA 研究会資料 2011-3-26

[2] J.H.シルヴァーマン、鈴木治郎訳「はじめての数論」発見と証明の大航海

ピタゴラスの定理から楕円曲線まで (株)ピアソン・エデュケーション(2003).

Jのコーディング・リスト

```
NB. シルバーマン「はじめての数論」p.165-168
NB. 連続降下法  2011/4/1
mod =: |~

qpr =: 1: = (#@q:)
qpt =: 3 : '(qpr y.) *. ((1 = y. mod 4) +. (2 = y.))'

NB. シルバーマン「はじめての数論」p.139, 140
NB. Non Quadratic Residue <=>
NB.     primes, cogruents of which (p mod 4) equal 3
NB.   NR 50 => 3 7 11 19 23 31 43 47

NR =: 3 : 0
(3 = (primes y.) mod 4)#(primes y.)
)

find =: 3 : 0
:
p =: x.
a =. y.
q =: (p-1)%4
A =: ((x:a)^(x:q)) mod (x:p)
M =: (1 + *: A) % p
B =: 1
p, M, A, B
)

desc =: 3 : 0
'p M A B' =. y.
u =. A mod M
v =. B mod M
C =. ((u*A)+(v*B))%M
D =. ((v*A)-(u*B))%M
R =. (*: C) + (*: D)
r =. (R) % p
p, r, C, D
)

wr =: 1!:2&2
descend =: 3 : 0
'p st' =. y.
t =. p find st
i =. 0
```

```
while. i < 4 do.
  t =. desc t
  wr i, CD2 =. +/ *: ((2, 3){t)
  if. CD2 = p do. goto_fin. end.
  rd 1
  i =. i + 1
end.
label_fin.
'C D' =. ((2, 3){t)
wr 'decomp to sum squared:'
wr (": p), ' = ', (": C), '^2 + ', (": D), '^2'
'*** end ***'
)

NB. compo 1105 => 33 4   p.173
compo =: 3 : 0
if. 2 = y. do. 1, 1 return. end.
MP =: q: y.
if. 0 = */ qpt"(0) MP do. 0, 0 return. end.
MC =: }:"(1) desc_prime"(0) MP
prod2sum / MC
)

NB. compox 25798500 => 5040 630 p.174
M =. q: y.
N =. +/"(1) (~. M) =/ M
'B A' =. |: 10 2 #: N
OD =: A # ~. M
EV =: B # ~. M
if. 0 = +/ (compo */OD) do.  'Not decomposed!' return. end.
Res =. (*/EV) * (compo */OD)
)

NB. Pythagorean Number === 2011/4/17 ===============
NB.   pytha 1105 => 1073 264 1105
pytha =: 3 : 0
C =. compox y.
'S T' =. (1, 1) prod2sum C
A =. S*T
B =. ((*:S)-(*:T)) % 2
A, B, (+/ *:C)
)
```