APL-J シンポジウム 2000/12/5

素数とＪの整数多倍長計算

－メルセンヌ数，フェルマー数からＲＳＡ暗号をめぐって－

西 川　利 男
(Toshio.Nishikawa@kiu.ne.jp)

１．はじめに

整数論は科学技術計算において微分積分，線形代数に比べてあまり重要視されず，趣味の数学などとみなされているのが現状である．現代のコンピュータ時代においては，整数論はコンピュータ・セキュリティのためのＲＳＡ公開鍵暗号への応用などその重要性はもっと認識されなくてはならない．

一般にアナログとディジタルという計測方式がよく対比される．元来，自然界の計測方式はアナログである．例えば銀河系宇宙の大きさは直径 9×10^5 光年，厚さ 5×10^4 光年であり，分子内の Ｃ-Ｃ 間原子結合の長さは 1.54 Ａ とその精度は有効数字３桁ないし４桁程度である．特別な場合を除けば科学技術の数の精度はこの程度であり，これを行う計算は科学的精度，つまり浮動小数点（Floating point）実数演算である．

これに比べるとディジタル方式は '数える' という人間の頭脳活動の産物としての人工のものである．つまり '桁' という概念をもとにいくらでも精度を上げられる．その基礎をなすのが整数論である．地球上の人間の数が５８億人，国家予算が８５兆円というのは最後の１位まで意味があるのである．人間のひとり，ふたりは誤差の範囲などと軽々しく言ってもらっては大変である．驚くなかれ，この場合の有効数字は１０桁にも及ぶ．整数演算とは実数演算に比べ，このように格段に高度な処理なのである．

このような精度の高い多桁の数値を扱うには力ずくで行えば出来るというものではない．このための武器が整数論と整数多倍長コンピュータに他ならない．

まず，小手調べとして次の問題を考えてみよう．

(1) 161 はどんな数で割り切れるだろうか？

これは 2 から順々に割っていけば，答は 7 と 23 と得られる．

(2) 16000000001 はどんな数で割り切れるか？

今度は 2 から順々に割っていくだけでは，到底答にはたどりつけないだろう．これは整数論の手法によらなくては無理である．あるいはＪの関数 q: を使って

```
   q: 16000000001
1889 847009
```

として得られる．

これまで多く整数論の教科書はその理論が主で，数値の演習問題の例はせいぜい４，５桁程度であり，実感として整数論を体得できるものではなかった．その一つの理由に多倍長精度の可能な整数計算プログラミング言語がなかったことがある．そのうちで UBASIC は日本生まれの数少ない多桁整数計算可能な言語の一つである．

Ｊの最新版では，単に数値の後ろに x を付け加えるだけ，あるいは関数 x:により拡張精度整数（extended precision integer）が可能であり，かつ p:（素数） q:（素因数分解）のような強力な関数が備えられ，多倍長整数計算が手軽に行える．

ここでは古くからよく知られたメルセンヌ数，フェルマー数から最新のＲＳＡ公開鍵暗号にいたるまでさまざまな計算をＪ言語がいかにあざやかに行えるかを示そう．それにより現代のディジタル数学としての整数論への関心を新たにしたいと思う．

２．メルセンヌ数と完全数

２．１ メルセンヌ数

メルセンヌ数とは Marin Mersenne が 1644 年に発表した次のような数である．すなわち，彼は

$$M_n = 2^n - 1$$

なる式で表わされるメルセンヌ整数は $n < 257$ の素数に対して

$$n = 2,\ 3,\ 5,\ 7,\ 13,\ 17,\ 19,\ 31,\ 67,\ 127,\ 257$$

のときには素数であり，それ以外では合成数であると予想した．そしてこの予想は必ずしも正しくなかったことが，後にいろいろな人により訂正された．それでもなお，現在最も大きな素数を与える式として知られている．

２．２ Ｊによる計算例

このようすをＪによって見てみよう．

```
NB. Mersenne number
mers =: 3 : '(2x^y.) - 1x'

   mers 2
3
   q: mers 2
3
```

```
   mers 3
7
   q: mers 3
7
   mers 5
31
   q: mers 5
31
   mers 7
127
   q: mers 7
127
   mers 11
2047
   q: mers 11
23 89
   mers 13
8191
   q: mers 13
8191
   mers 17
131071
   q: mers 17
131071
   mers 19
524287
   q: mers 19
524287
   mers 23
8388607
   q: mers 23
47 178481
   mers 29
536870911
   q: mers 29
233 1103 2089
```

```
   mers 31
2147483647
   q: mers 31
2147483647
   mers 37
137438953471
   q: mers 37
223 616318177
   mers 41
2199023255551
   q: mers 41
13367 164511353
   mers 43
8796093022207
   q: mers 43
431 9719 2099863
   mers 47
140737488355327
   q: mers 47
2351 4513 13264529
   mers 53
9007199254740991
   q: mers 53
6361 69431 20394401
   mers 59
576460752303423487
   q: mers 59
 break
|        q:mers 59
```

Jではここまででであった．しかし，このような計算をコンピュータがない時代にどうやってやったかまさに驚異である．

２．２ メルセンヌ数と完全数

また，メルセンヌ数は完全数との関係においても有名である．完全数とはその約数の和が元の数に等しくなる数であり，ピタゴラスの時代より神秘な数として知られている．

ユークリッドにより与えられた完全数の以下の式はメルセンヌ数 M_n と次のように関係付けられている．

$(2^{n-1})(2^n - 1) = (M_n + 1)M_n / 2$

２．２ Ｊによる計算例

これはＪでは次のように示される．

```
NB. generate Perfect number from Mersenne number
mperf =: 3 : ' -: (y. + 1x)*y.'

NB. test Perfect number
s =: ~.@q:                              NB. 素数だけの取出し
t =: >:@(+/@(] =/ ~.)@q:)               NB. 素因数分解のべきの数＋1
tau =: */@t                             NB. 約数の個数
sigma =: */@( (<:@(s^t)) % (<:@s) )     NB. 約数の和
tperf =: 2&* = sigma

   mers 2
3
   mperf mers 2
6
   tperf mperf mers 2
1
   mers 3
7
   mperf mers 3
28
   tperf mperf mers 3
1
   mers 5
31
   mperf mers 5
496
```

```
    tperf mperf mers 5
1
    mers 7
127
    mperf mers 7
8128
    tperf mperf mers 7
1
    mers 11
2047
    mperf mers 11
2096128
    tperf mperf mers 11
0
    mers 13
8191
    mperf mers 13
33550336
    tperf mperf mers 13
1
    mers 17
131071
    mperf mers 17
8589869056
    tperf mperf mers 17
1
    mers 19
524287
    mperf mers 19
137438691328
    tperf mperf mers 19
1
    mers 23
8388607
    mperf mers 23
35184367894528
```

```
   tperf mperf mers 23
0
   mers 29
536870911
   mperf mers 29
144115187807420416
   tperf mperf mers 29
0
   mers 31
2147483647
   mperf mers 31
2305843008139952128
   tperf mperf mers 31
1
```

ここでも $n = 11, 23, 29$ の場合に上の式から計算される数は完全数ではない．Jではメモリと時間の制約があり，ここまでにしておこう．

３．フェルマー数とオイラーの快挙

３．１ フェルマー数とは

Pierre de Fermat(1601-1665)は

・フェルマーの小定理

p が 素数のとき，*p* と互いに素の *x* とに関して

$$x^{p-1} \equiv 1 \mod p$$

・フェルマーの最終定理

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

は $n > 2$ では整数解を持たない

など整数論に数々の名を残した偉大なアマチュア数学者である．

フェルマー数ももちろん彼にちなむもので，次の式で表わされる数である．

$$F_n = 2^{2^n} + 1$$

最初の方を上げてみると

F_0 = 3

F_1 = 5

F_2 = 17

F_3 = 257

F_4 = 65537

F_5 = 4294967297

....

のようになる．はじめの方はすぐ分かるように素数である．このようにしてフェルマーはこの式で得られる数はすべて素数である，と予想した．しかし彼は証明したわけではなかった．

1729 年，オイラー Leonhard Euler(1707-1783)は

$$F_5 = 2^{2^5} + 1$$

については素数ではなく合成数であることを示した．いうまでもなく約数が１つでも見つかれば素数ではなく合成数である．オイラーはもちろん手当たりしだいに順々に割って調べたわけではない．以下オイラーの数式扱いのマエストロとしての手腕を見ていこう．

３．２ オイラーのフェルマー数（F_5）攻略

オイラーの攻略は次の準備からはじまる．

（定理Ａ）a を偶数とし，p は aの約数ではないが，$a+1$ を割り切る素数とする．

そのときは k を任意の整数として

$$p = 2k + 1$$

となる．

（証明）

a が偶数なので $a+1$ は奇数となる．したがって奇数の約数は奇数であり

$$p = 2k + 1$$

なんとなれば （奇数）×（奇数）＝（奇数）

（奇数）×（偶数）＝（偶数）

（偶数）×（偶数）＝（偶数）

（定理Ｂ）a を偶数とし，p は aの約数ではないが，a^2+1 を割り切る素数とする．

そのときは k を任意の整数として

$$p = 4k + 1$$

となる．

（証明）

a が偶数なので a^2 も偶数であり，a^2+1 は奇数である．したがって p は奇数である．

ここですべての奇数は次のように表わせる．

$$p = 4k + 1$$

または $p = 4k + 3$

しかし，この場合は $p = 4k + 3$ とはなり得ない．以下それを示す．

もし $p = 4k + 3$ と仮定してみる．

まず，フェルマーの小定理により

$$a^{p-1} - 1 = a^{(4k+3)-1} - 1 = a^{4k+2} - 1 \equiv 0 \mod p$$

上の式は pで割り切れる．

一方，次の積を考えると

最初の因数 a^2+1 からこの積全体も p で割り切れる．一見，複雑なこの積も掛け合わせ

$$(a^2+1)(a^{4k} - a^{4k-2} + a^{4k-4} - \Lambda + a^4 - a^2 + 1)$$

ると，$a^{4k+2}+1$ になる．

結局，p は $a^{4k+2}-1$ と $a^{4k+2}+1$ のいずれも割り切ることができる．したがって p はその差

$$(a^{4k+2}+1) - (a^{4k+2}-1) = 2$$

の約数になるはずである．しかしながら奇数の素数である pは 2 を割ることは出来ない．

これは矛盾である．つまり $p = 4k + 3$ とする仮定は成り立たず

$$p = 4k + 1$$

でなければならない，と結論できる．

（定理C）a を偶数とし，p は a の約数ではないが，$a^4 + 1$ を割り切る素数とする．そのときは k を任意の整数として

$$p = 8k + 1$$

となる．

ここでは証明を省略するが，オイラーはさらに進めて次々と定理を得た．

（定理D）もし p が $a^8 + 1$ を割り切るなら $p = 16k + 1$ となる．

（定理E）もし p が $a^{16} + 1$ を割り切るなら $p = 32k + 1$ となる．

（定理F）もし p が $a^{32} + 1$ を割り切るなら $p = 64k + 1$ となる．

以上の準備のもとに，オイラーは $a = 2$ とした

$$2^{32} + 1$$

の約数を調べた．つまり p が約数となるならば

$$p = 64k + 1$$

という形になる．

ここで例えば

$$k = 1, 2, 3, \ldots, 10$$

に対して

$$p = 65, 129, 193, 257, 321, 385, 449, 513, 577, 641$$

となるが，素数以外は候補から取り除くと

$$p = 193, 257, 449, 577, 641$$

について調べればよい．オイラーはこれらの候補に対して実際に割り算を実行して，幸い５番目にして 641 という約数を得たのである．順々に割り算を行う愚より，ずっと早いことはいうまでもない．これこそ数学探偵のマエストロのあざやかな技である．

３．３ Ｊによる計算例

次のようにいくつかの関数を定義する.

```
NB. ダンハム「数学の知性」p.280-287
NB. オイラーによるフェルマーの予想の否定

NB. make Fermat number:  Fn <-> fm n
fm =: 3 : '1x + 2x^2x^y.'
NB. make p value:  e.g. p =. n pk k
pk =: 4 : '1x + y.*(2x^>: x.)'
NB. multi-precision integer divide
NB.  returns (quotient, residue)
idiv =: 3 : 0
:
r =. y. | x.
((x. - r)%y.) , r
)
```

実行はつぎのようになる.

```
   F5 =. fm 5
   F5
4294967297

   p =. 5 pk i.15
   p
1 65 129 193 257 321 385 449 513 577 641 705 769 833 897

   p ,. F5 idiv"(0) p
  1 4294967297   0
 65   66076419  62
129   33294320  17
193   22253716 109
257   16711935   2
321   13379960 137
385   11155759  82
```

```
449     9565628 325
513     8372255 482
577     7443617 288
641     6700417   0
705     6092152 137
769     5585133  20
833     5156023 138
897     4788146 335

   F5 idiv 641
6700417 0
   q: F5
641 6700417
```

フェルマー数 F5 を計算し，p = 1，..，641，.. 897 までについて 関数 idiv により整数多倍長の割り算を行い，実際 p = 641 のときに割り切れた．なお，Jの q: では一瞬で素因数分解が得られる．

```
  F6 についてもやってみよう．
NB. Randley (1880)
   F6 =. fm 6
   F6
18446744073709551617
   F6 idiv 274177
67280421310721 0
```

フェルマー数 F6 では値を求めるところまでは出来たが，約数を探すのは無理のようである．ランドリーによる約数を使って，合成数であることの確認は出来た．また，Jの q: では時間がかかり途中でとりやめた．

４．暗号と整数論

４．１　暗号(cipher)とは

英語の 'cipher' という語は現在では '0', '1', '2' といった数字を意味するが，本来はシーザー暗号（Caesar cipher）というように暗号を意味し，シーザーの昔から軍事で重要な技術だった．シーザーの初めての暗号指令

平文　ATTACK AT DAWN

暗文　BUUBDLABUAEBXO

は次の文字に置き換えるだけだが，その後いろいろな複雑な方式が考えだされ，使われてきた．現在でも暗号技術は軍事に限らず，商業通信，外交文書など重要性は変わらない．特にコンピュータのセキュリティに関連して暗号はますます重要であり，その最新技術がＲＳＡ公開鍵方式暗号である．

４．２　Ｊによる公開鍵暗号の例

ＲＳＡ（R.Rivest, A.Shamir & L.Adleman）の公開鍵方式の暗号のしくみは長大数の素因数分解は極めて困難でほとんど不可能に近いという整数の性質に基づいている．

Ｊにより例をあげてそのしくみを見ていこう．例えば２つの素数 47 と 79 より

$N = 47 \times 79 = 3713$

を得て，これだけの文字種類が表わせるとする．このときは

$L = (47-1) \times (79-1) = 3588$

とこれと互いに素である数

$p = 37$

を公開鍵とする．つまり Nと pとは公開されて，メッセージ文字列に対応した数値を Xとするとき

$Y = X^p \bmod N$　　つまり $Y = X^{37} \bmod 3713$

なる暗号文をつくり送る．

解読するには

$p\,s \equiv 1 \bmod L$　　つまり $37 \times 97 \equiv 1 \mod 3588$

なる s より

$Z = Y^s \bmod N$　　つまり $Z = Y^{97} \bmod 3713$

として元のメッセージに戻せる．

しかしながら，たとえ N と p がわかっても，N の素因数分解がわからなければ，L は容易に求められず，従って s も得られず，解読されず秘密が保たれる．

つぎのようなＪのプログラムを作った．ここで，文字の数値化では'A' → 1, 'B' → 2, 空白 → 0 で英文字２文字ずつを１つの数値になるようコード化する．

```
NB. R. Sedgewick, "Algorithms - Chapter 23 Cryptology", p.339

NB. Encode 'A' -> 1, 'Z' -> 26, SP(' ') -> 0
encod =: -&64@(a.&i.)
encode =: encod`0: @.(' '&=)

NB. Decode 1 -> 'A', 26 -> 'Z', 0 -> SP(' ')
decod =: encod ^:_1
decode =: decod`(' '"_) @. (0&=)

NB. Make code
xc =: 3 : '100#.((-:#y.), 2) $ y.'
makecode =: 3 : 'xc encode"0 y.'

NB. Solve code
yc =: 3 : '100 100#:y.'
solvcode =: 3 : ',decode"0 yc y.'
```

実行はつぎのようになる.

メッセージ文字列を数値化する.

```
   X =. makecode 'ATTACK AT DAWN'
   X
120 2001 311 1 2000 401 2314
   solvcode X
ATTACK AT DAWN
```

公開鍵を使って暗号化して送る.

```
   Y =. 3713 |(x: X)^37
   Y
1404 2932 3536 1 3284 2280 2235
```

秘密鍵を使って解読する.

```
   Z =. 3713 |(x: Y)^97
   Z
120 2001 311 1 2000 401 2314
```

```
   solvcode Z
ATTACK AT DAWN
```

プログラム encode，decode は次のようにはたらく．

```
   encode"(0) 'AB CZ'
1 2 0 3 26
   decode"(0) 1 2 0 3 26
AB CZ
```

計算の途中経過の数値をみて見ると，次のように大変な長大数である．

```
   XX =. (x: X)^37
   Y =. 3713 |XX
   0{XX
85056224998211021445761316841148299345920000000000000000000000000000000000000
   0{Y
1404
   1{XX
14000459175376804467912949231282420945491641181562842586399055661487207855713284
3332297059802411089216125760782162664074001
   1{Y
2932
   2{XX
17066104240455169288000053202976125042985163928805452800241922967047169214906626
3120153234071
   2{Y
3536
   3{XX
1
   3{Y
1
   4{XX
13743895347200000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000
   4{Y
3284
```

```
   5{XX
20717711778840901230914616878324551141972902378233289755935770489506692083044953
05340529386574801
   5{Y
2280
   6{XX
30300104114041206444959812463157292753259163827839927107928268238310763589277588
50012038446264473569096063180945508532643430 4
   6{Y
2235
```

５．おわりに

メルセンス数，フェルマー数さらには暗号理論を通じて整数論の現代における重要性を喚起するとともに，Ｊ言語の多倍長演算機能がこれらの計算にいかに手軽で，かつ強力であるかを示した．

参考文献

・ Albert H. Beiler, "Recreations of the Theory of Numbers", Dover (1964).
・ 木田祐司，牧野潔夫，「UBASIC によるコンピュータ整数論」，評論社 (1994).
・ W. Dunham，中村由子，「数学の知性」，現代数学社 (1990).
・ Robert Sedgewick, "Algorithms", Addison-Wesley (1988).

付録 長大整数の約数を求める

```
NB. factoring large numbers
NB.  A. Beiler, "Recreations in the Theory of Numbers", p. 237
resol =: 3 : 0
t =. 6!:1 ''
N =. x: y.
s =.  -: <: N
a =. <. %: s
r =. s  -*: a
n =. 0
while. n <: 10000
do.
   A =. a + n
   B =. a  - n
   C =. (>: +: r) + (+: *: n)
NB.   wr n, A, B, C
NB.   wr (A +. C), (B +. C)
   if. 1 ~: GCD =. A +. C
      do. break. end.
   if. 1 ~: GCD =. B +. C
      do. break. end.
   n =. >: n
end.
wr 'divisor = ', (":GCD), ', at n = ', (":n)
(": (6!:1 '') -t), ' sec'
)

   resol 1600001
divisor = 13, at n = 3
0 sec
   q: 1600001
13 123077
   resol 16000001
divisor = 109, at n = 6
0.05 sec
```

```
   q: 16000001
109 229 641
   resol 160000001
divisor = 7, at n = 2
0 sec
   q: 160000001
7 1453 15731
   resol 1600000001
divisor = 1889, at n = 51
0.38 sec
   q: 1600000001
1889 847009
   resol 16000000001x
divisor = 13219, at n = 3091
9.78 sec
   q: 16000000001x
13219 1210379
```