完全数、友愛数とメルセンヌ数の計算問題

SHIMURA Masato
JCD02773@nifty.com
URL:http://homepage3.nifty.com/asagaya_avenue

2016 年 1 月 25 日

目次

1 ユークリッドと完全数 1

2 メルセンヌ数 5

3 オイラー 7

4 友愛数 10

付録 A 整数の検出 14

付録 B メルセンヌ数発見の最前線 14

概要

完全数はユークリッド (紀元前 300 頃）の原論に定義が述べられているが今に至るも未解決の問題も残されている。友愛数はさらに古くピタゴラスも登場する。何れも素数を求める計算は PC に高負荷がかかるが、ユークリッドやオイラーはスマートな解法を見つけている。

1 ユークリッドと完全数

完全数は以前力技法のスクリプトを作成したことがあったが、最近読んだ W. ダンハムの「オイラー入門」にユークリッドからオイラーまでの功績が簡潔に紹介されていた。

1.1　ユークリッドの完全数の定義 (1)

ユークリッドの完全数の定義を現代風に表す

定義: 完全数とはその真の約数の和がその数自身に等しいものである

整数の約数とは n を割り切る整数の総称である。($n = 0$ の場合に 0 を含めることがある)。

整数を判別する関数を持っている言語は無いので作成する。

1.2　約数を求める Script と解説

約数を求める　約数を求めるスクリプト

```
p0=:}.@i.
p1=: % p0
find_integer=:  3 : '(({:("1) 2&x: p1 y)e. 0 1) # p0 y'
pn=:  find_integer ;  +/@:find_integer
```

経過と解説　.

- i. で 0 から 5 の 6 個の数列を打ち出し先頭の 0 を落とす。(0 オリジン)

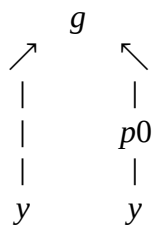
```
p0=:}.@i.

   p0 6
1 2 3 4 5
```

- 元の数を割る Fook で簡潔に表現。

```
p1=: % p0
6 % 1 2 3 4 5

   p1 6
6 3 2 1.5 1.2
```



- 整数部分を取り出す

```
 find_integer=:3 : '(({:("1) 2&x: p1 y)e. 0 1) # p0 y'
```

整数の指標の作成 e. は *Menbers of*

1 を取り出している。*e.* は `e. 1 0 _1` のようにまとめて走査できる

```
{:("1) 2&x: p1 y)e. 0 1
```

指標 `# p0 y` で *1* の部分をコピー *(#)* する

```
   find_integer 6
1 2 3
```

- 完全数と確認。 約数とその合計を表示する

```
pn=:find_integer ; +/@:find_integer
   pn 6
+-----+-+
|1 2 3|6|
+-----+-+
```

素数を求める　　素数を求める J のプリミティブ (原始関数）*Prime (p:)* がある

- 単項では指定個数の素数を打ち出す

```
   p: i.10
2 3 5 7 11 13 17 19 23 29
```

- パラメータ *3* で因数分解

```
   3 p: 220
2 2 5 11
```

- *2* では重複個数を数え上げる

```
   2 p: 220
2 5 11
2 1  1
```

- 大きな数の因数分解や約数計算は *PC* に高負荷がかかり、しばしば立ち往生する

  古代ギリシャのエラトステネスの篩のような方法も *PC* 時代に十分活用される場面が出てくる

1.3　ユークリッドの時代の完全数

古代ギリシャ時代には４個の完全数が知られていた。

```
     pn 6
+-----+-+
|1 2 3|6|
+-----+-+

   pn 28
+----------+--+
|1 2 4 7 14|28|
+----------+--+

   pn 496
+-------------------------+---+
|1 2 4 8 16 31 62 124 248|496|
+-------------------------+---+

   pn 8128
+---------------------------------------------+----+
|1 2 4 8 16 32 64 127 254 508 1016 2032 4064|8128|
+---------------------------------------------+----+
```

両項の p : を引数 3 で用いると因数分解となる。

```
     3 p: 6
2 3

   3 p: 28
2 2 7

   3 p: 496
2 2 2 2 31

   3 p: 8128
2 2 2 2 2 2 127
```

古代ギリシャの完全数に含まれる 2 以外の素数は各一個づつで 3,7,31,127 である。

1.4　ユークリッドの完全数の定義 (2)

ユークリッドは次のように書き記している。

> もし、1 から始まり、続けて 2 倍したものを並べていき、この和が素数になるまでこれを続け, もしこの和に最後のものをかけるとしたら、その積は完全であろう。

ユークリッドは $1+2+4+\cdots+2^{k-1}$ が素数であることを仮定したが...

```
   <\  3 p: 496
+-+---+-----+-------+----------+
|2|2 2|2 2 2|2 2 2 2|2 2 2 2 31|
+-+---+-----+-------+----------+
```

```
     2^>:i.10
2 4 8 16 32 64 128 256 512 1024
```

```
 */L:0  <\  3 p: 496
+-+-+-+--+---+
|2|4|8|16|496|
+-+-+-+--+---+
```

$1+2+4+\cdots+2^{k-1}$ の和は 2^k-1 となる。

ユークリッドの記述を現代風に表すと

定義: もし 2^k-1 が素数で $N=2^{k-1}(2^k-1)$ のとき N は完全数である

この定理はある数が完全数であるための十分条件を与えている。

$N=28$ を例に確認してみよう。

$$N=2^{k-1}(2^k-1)$$

$$28=2^{3-1}(2^3-1)=2^2(2^3-1)=4\cdot 7$$

ユークリッドのこの定理は完全数の探索を $p=2^k-1$ 型の素数を探す問題に置き換え効率化を図ったものである。

2　メルセンヌ数

この n^k-1 がもたらす素数はメルセンヌ数と呼ばれている。メルセンヌはデカルトやフェルマーを結ぶ数学者のネットワークの中心にいたフランス人のイエスズ会の牧師で数学者であった。

次はコンウエイ・ガイの「数の本に」よる。

メルセンヌはメルセンヌ数を次のように主張・予想した

k=2 3 5 7 13 17 19 31 67 127 257

- $2^{127}-1$ がメルセンヌ数と証明されたのは 1876 年であった。

- 31 までは素数になるが 67 と 257 は素数にならない。

```
   3 p: <: 2^2 3 5 7 13 17 19 31x
         3
         7
        31
       127
      8191
    131071
    524287
2147483647
```

```
   3 p: <: 2^67 127x
                                        193707721 761838257287
170141183460469231731687303715884105727            0
```

3 p: <: 2^257 では J の同時計算ではが挙動不審になり（2.4Ghz　Memory 4GByte)PC 苛めにもなる。ループをぐるぐる回してひとつづつ計算した方がよい。

メルセンヌは 61,89,107 を見逃した。

今日、メルセンヌ数は数百万台の PC をつないだ分散コンピューティングで、数年かけて探査が進められ、40 数個が発見されている。量子計算の時代には新しい世界が見られるだろう。

メルセンヌ御製のメルセンヌ数から完全数を求めてみよう。どれかの数に 127x のように x を付けると浮動小数点でなく、どこまでも演算する。

```
k=: 2 3 5 7 13 17 19 31 61 89 107 127x
k,. (2^<:k) * <:2^k
  2                                                                  6
  3                                                                 28
  5                                                                496
  7                                                               8128
 13                                                           33550336
 17                                                         8589869056
 19                                                       137438691328
 31                                                   2305843008139952128
 61                                 2658455991569831744654692615953842176
 89                   191561942608236107294793378084303638130997321548169216
107          13164036458569648337239753460458722910223472318386943117783728128
127 14474011154664524427946373126085988481573677491474835889066354349131199152128
```

3　オイラー

$2^{31} - 1$ が完全数であることを示したのはオイラーである。
オイラーは $\int n$ と表記したが、ダンハムは $\sigma(n)$ を用いている。
オイラーはの議論は約数に加えその数自体を含めている。

$$\sigma(6) = 1 + 2 + 3 + 6 = 12$$

オイラーは真の約数全体の和ではなく、すべての約数の和に焦点を当てた。

3.1　Script

```
pe0 =:find_integer , ]
pe=: pe0 ; +/@pe0
```

3.2　$\sigma(n)$ を用いたオイラーの定理

(1)　p が素数であることと、$\sigma(p) = p+1$ が成り立つことは同値である。

```
   pe 11
+----+--+
|1  11|12|
+----+--+
```

(2)　N が完全数であることと、$\sigma(N) = N + N = 2N$ が成り立つことは同値である。

```
   pe 28
+--------------+--+
|1  2  4  7  14  28|56|
+--------------+--+
```

(3)	p が素数の時、$\sigma(p^r) = \dfrac{p^{r+1}-1}{p-1}$ が成り立つ	`(<:11^3+1)%<:11x`
		`1464`
		
		`pe 11x^3`
		`+-----+----+`
		`|1 11 121 1331|1464|`
		`+-----+----+`

(4)	もし p と q が相異なるなら、$\sigma(pq) = \sigma(p)\sigma(q)$ が成り立つ。	$\sigma(21) = 1+3+7+21 = 4\times 8 = \sigma(3)\times\sigma(7) = (1+3)\times(1+7)$
(5)	もし a と b が互いに素なら、$\sigma(ab) = \sigma(a)\sigma(b)$ が成り立つ。	$\sigma(4800) = \sigma(2^6)\times 3\times 5^2) = \sigma(2^6)\times \sigma(3)\times\sigma(5^2) = 127\times 4\times 31 = 15748$

(5) を検証してみよう

- 4800 を素数に分解する。　個数を数える　`2&p:` を用いる

```
   2 p: 4800
2 3 5
6 1 2
```

- 分解した素数に適用する。$2^6, 3, 5^2$.
  オイラーによる約数の分解は計算の見通しが良くなる強力なツールである。

```
   > pe L:0 (2^6);3;5^2
+----------------+---+
|1 2 4 8 16 32 64|127|  NB. 2^6=64
+----------------+---+
|1 3             |4  |  NB. 3
+----------------+---+
|1 5 25          |31 |  NB. 5^2=25
+----------------+---+

  */ 127 4 31
15748
```

正直に $\sigma(4800)$ を計算してみる（最後に 15478 が現れている）

```
 ; pe 4800
1 2 3 4 5 6 8 10 12 15 16 20 24 25
30 32 40 48 50 60 64 75 80 96 100
120 150 160 192 200 240 300 320 400
480 600 800 960 1200 1600 2400 4800 15748
```

定理：N が偶数の完全数ならば $N = 2^{k-1}(2^k - 1)$ と書ける。ここで $2^k - 1$ は素数である。

オイラーによるユークリッドのレシピの仕上げである。しかし、メルセンヌ数の無限性や奇数の完全数の存在はいまだ未解決である。

3.3 その他の完全数の特性

- 完全数の $\sigma(n)$ の逆数の和は 2 になる

```
   ,. ;(L:1)  pn (L:0) 6; 28;496;8128
+-------------------------------------------------+
|1 2 3 6                                          |
+-------------------------------------------------+
|1 2 4 7 14 28                                    |
+-------------------------------------------------+
|1 2 4 8 16 31 62 124 248 496                     |
+-------------------------------------------------+
|1 2 4 8 16 32 64 127 254 508 1016 2032 4064 8128|
+-------------------------------------------------+
```

```
      +/@:%@ ;(L:1)  pn (L:0) 6; 28;496;8128
    +-+-+-+-+
    |2|2|2|2|
    +-+-+-+-+
```

4　友愛数

先のオイラーの (4) を考えてみよう。

もし、p, q が相異なる素数なら、$\sigma(pq) = \sigma(p)\sigma(q)$

```
  pe 21
+--------+--+
|1 3 7 21|32|
+--------+--+
```

p、q の素数は $1, p, q, pq$ の４つのみである。

$\sigma(pq) = 1 + p + q + pq = (1 + p) + (1 + q) = (1 + p)(1 + q) = \sigma(p)\sigma(q)$

$32 = 1 + 3 + 7 + 21 = \sigma(3) + \sigma(7) = 4 \times 8$

$\sigma(m) = m + n = \sigma(n)$ と美しい対称性を持つものが友愛数の組である。

4.1　サビート・イブン=クッラ

友愛数はピタゴラス、ユークリッドから始まり、アラブで研究され、フェルマーからオイラーに伝わった。

アラブの中心は９ C のサビート・イブン＝クッラで次のアルゴリズムを示した。子のアルゴリズムは強力であるが、すべての友愛数を探査するものではなく、またこのアルゴリズムより強力なものもない。

そして、友愛数も完全数と同様の未解決問題を抱えている。

$$
\begin{aligned}
p &= 3\times \quad 2^{n-1} - 1 \\
q &= 3\times \quad 2^{n} - 1 \\
r &= 9\times \quad 2^{2n-1} - 1
\end{aligned}
$$

n と p, q, r が素数のとき $2^n pq$ と $2^n r$ は友愛数となる

1. 因数分解や約数を求めるには負荷が大きいので Loop で考える
2. 最初に友愛数のベースになるマトリクスを作成する

```
   ynmat 1 10
1  2   3    4    5     6     7      8       9      10 NB. n
2  4   8   16   32    64   128    256     512    1024 NB. rn=2^n
6 12  24   48   96   192   384    768    1536    3072 NB. sn=3 * rn
5 11  23   47   95   191   383    767    1535    3071 NB. pn=sn-1
0 71 287 1151 4607 18431 73727 294911 1179647 4718591 NB. qn=(sn-1)*sn -1 // 
```

$$
\begin{aligned}
&n\\
&rn = 2^n\\
&s_n = 3 \times 2^n\\
&p_n = 3 \times 2^n - 1\\
&q_n = 9 \times 2^{2n-1} - 1
\end{aligned}
$$

```
ynmat=: 3 : 0
NB. Usage: ynmat 1 10  // start and n
NB. mat 5 raw is N rn Sn pn qn
'Start N0'=. y
N=. Start + i. N0 * 1x  NB. extend with 1x
rn=.2^N
tmp1=.rn,: sn=. 3*rn
tmp2=. tmp1,pn=. <:sn
N,qn=.tmp2,  0, ><:@:  */(L:0) 2<\ sn
)
```

3. p, q 双方が素数の時に友愛数になる。
4. Loop 型の Script
   - 小さな幅で $p.q$ 双方が素数の場合を検出する

```
   yn0 1 10
+-------+---------+-----------+
|2 11 71|4 47 1151|7 383 73727|
```

```
+-------+---------+-----------+
 n pn qn
```

- 友愛数を計算する

$$M = p_n P_{n-1} r_n = 2^n p_{n-1} P_n$$

$$N = r_n qn = 2^n q_n$$

```
   yn 1 10
220 17296 9363584
284 18416 9437056
```

$$220 = 5 \times 11 \times 2^2$$

$$284 = 71 \times 2^2$$

- Script

```
NB. for Loop
pprime0=: 3 : 0
ind=.1 = # 3 p: tmp=: 3{ y
if. 1=ind do. ANS=. tmp else. ANS=. 0 end.
)
qprime0=: 3 : 0
NB. 3 p: 0  is error
 tmp=: 4{ y
if. 0=  tmp do. ANS=. 0   goto_end. end.
ind=.1 = # 3 p: tmp
if. 1=ind do. ANS=. tmp else. ANS=. 0 end.
label_end.
)

yn0=: 3 : 0
NB. using Loop because of avoid hungup
mat=. |: ynmat y
```

```
num=. #  mat
ANS=. <''
for_ctr. i.  num do.
tmpraw=. ctr{mat
tmp=.>(pprime0  ;  qprime0) L:0 tmpraw
  if. 0 e. tmp  do. goto_skip. end.
ANS=. ANS,<({. tmpraw),tmp
label_skip.
end.
 }.ANS
)

yn=: 3 : 0
mat=: |: ynmat y
ANS=: >yn0 y
tmp0=. (2-~{."1 ANS){mat
px=.(2^ {."1 ANS )* */"1 (3{"1 tmp0) ,. 1{"1 ANS
qx=. (2^{."1 ANS)* {:"1 ANS
px,:qx
```

5. この中世アラブ時代の簡易アルゴリズムは (5020,5564) を見落としている。友愛数を求めるスマートなアルゴリズムはまだ発見されていないようだ。

付録 A　整数の検出

整数を検出するエレガントなスクリプトを西川敏男氏から教えていただいた

```
integer=: (<. = ]) # ]


   integer 1 2 1.7 3.14 4 0 5.25
1 2 4 0
```

Floor(<.) で切り捨てたものと元の値を比較し、同じもののみコピーする

`find_integer0=:  3 : '(({:("1) 2&x: p1 y)e. 1) # p0 y'`

`find_integer` は次のように簡約できる

```
p0=:}.@i.
p1=: % p0
find_integer=: 3 : '((<. = ]) p1 y) # p0 y'
pn=:  find_integer ;  +/@:find_integer
NB. Usage: pn 6
```

付録 B　メルセンヌ数発見の最前線

2016 年 1 月 24 日の朝日新聞は、U.S. セントラルミズリー大の計算機科学のカーチス・クーパー教授が 2237 万桁のメルセンヌ数を発見したと伝える。800 万台の PC をつなぐ GIMPS を用いて、自己の持つ記録を大幅に更新した。1 億桁を超えるメルセンヌ数を発見した場合、賞金 5 万ドルが GIMPS から提供されるようだ。

References

J.H. コンウエイ ·R.K. ガイ　根上生也訳「数の本」シュプリンガー東京　2001

W. ダンハム · 黒川、若山、百々谷訳「オイラー入門」シュプリンガー東京　2004