（ＪＡＰＬＡ 2005/11/26）

素数と合成数等に関する話題

帝京平成大学　鈴木義一郎

１．全ての自然数は３つのグループに分けられる

「素数」とは、１と自分自身以外に約数をもたない数のことである。これに対して、１および自分自身のほかに、真の約数をもっている数は、「合成数」と呼ばれている。また、“１”という数だけは、ただ１つの約数しかもたない特殊な数値である。

したがって、全ての自然数は、次の３つのグループに分けられる。

1）第１のグループは、ただ１つの約数しかもたない唯一の数（１）。

2）第２のグループは、２つの約数(１と自分自身)をもっている数（素数）。

3）第３のグループは、３つ以上の約数をもっている数（合成数）。

２．エラトステネスのふるい

エラトステネスという数学者は、自然数列の各数を全て書き連ねて、２の倍数を全て取り除き、次に３の倍数、さらに５の倍数といった具合に、次々に素数以外の数をふるい落とした（取り除いた）残りの数を素数であるとした（「エラトステネスのふるい」）。

例えば、１と３５の間にあるすべての素数を求める場合を考えてみよう。まず、$\sqrt{35} < 6$ であるから、２，３，５の倍数を取り除いていけばよい。

まず２以外の２の倍数を取り除けば

2,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35

といった数が残る。さらに３以外の３の倍数を取り除いて

2,3,5,7,11,13,17,19,23,25,29,31,35

となる。そして最後に、５以外の５の倍数を取り除けば

2,3,5,7,11,13,17,19,23,29,31

という結果が得られ、これが求める素数の全体ということになる。

er_filter=:4 :'x.,(0~:x.|y.)#y.'
]p=.2 er_filter p=.}.1+i.35
2 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35
	]p=.3 er_filter p
	3 2 5 7 11 13 17 19 23 25 29 31 35
	]p=.5 er_filter p
	5 3 2 7 11 13 17 19 23 29 31

prime=:3 :0	prime1=:3 :0

```
r=.x=.2
while. x<y.
   do. r=.r,(1=#q:x)#x=.x+1
end.
)
    prime 35
2 3 5 7 11 13 17 19 23 29 31
```

```
r=.p:i.>:x [ x=.<:<.y.%^.y.
while.y.>p:x
   do. r=.r,(y.>:t)#t=.p:x=.x+1
end.
)
    prime1 35
2 3 5 7 11 13 17 19 23 29 31
```

３．素数は無数にある

一般に、次のような事実が知られている。

> 任意の数Ｎとそれを２倍した２Ｎの間には少なくとも１個の素数がある。
>
> （したがって、素数は無数にある！）

M,N を互いに素、すなわち（M，N）＝１とすると

M、M＋N，M＋２N，M＋３N，……

という数列の中には無数の素数が含まれる(『Dirichlet の定理』と呼ばれている)。

レーマという数学者は、１９３２年、$2^{257}-1$が合成数であることを立証したが、解決までに当時の計算機でまる１年を費やしたという。

１９５０年以前に知られている最大の素数は

$2^{127}-1=$ 170141183460469231731687303715884105727

１９６１年には$2^{4423}-1$が素数であることが証明された。

４．素数定理と素因数分解

ある正の数 x (必ずしも自然数でなくともよい)に対して、ｘ以下の素数の数を π（ｘ）と表すと、100 以下の素数は 25 個、1000 までには 168 個の素数があるから、

$\pi(100)=25$，$\pi(1000)=168$，$\pi(10^4)=1229$，$\pi(10^5)=9592$

$\pi(10^6)=78498$，$\pi(10^7)=664579$，$\pi(10^8)=5761455$，$\pi(10^9)=50847534$，

であるし、また素数は無数にあることから、

ｘ→∞とすると、π（ｘ）→∞

となることも分かる。

ガウス(1777-1855)というドイツの数学者は、

> ｘ→∞とするとき、π（ｘ）とｘ/log ｘの比が１に近づく。

ことを予想し、1896 年にアダマールとドゥ・ラ・バレ・プーサンがそれぞれ独立に証明を与えた。これが『素数定理』と呼ばれている命題である。

> １より大きな自然数Ｎはｐ，ｑ，ｒ，……を異なる素数として

$$N = p^a q^b r^c \cdots\cdots$$

のようにただ一通りにあらわすことができる(「素因数分解」という)。

したがって、Nの約数はすべて

$$p^x q^y r^z \cdots\cdots \quad (0 \le x \le a\,.\,0 \le y \le b\,.\,0 \le z \le c\,.\cdots\cdots)$$

という形で表現できる。つまり、Nの約数は

$$(1 + p + p^2 + \cdots\cdots + p^a)(1 + q + q^2 + \cdots\cdots + q^b)(1 + r + r^2 + \cdots\cdots + r^c)\cdots\cdots$$

という式を展開したときの各項がすべてNの約数ということになる。これより、Nの約数のすべての合計をS(N)と表すと

$$S(N) = \frac{p^{a+1} - 1}{p - 1} \times \frac{q^{b+1} - 1}{q - 1} \times \frac{r^{c+1} - 1}{r - 1} \times \cdots\cdots$$

といった関係が得られる。また、Nのすべての約数の個数をT(N)とあらわすと

T(N)＝(a＋1)(b＋1)(c＋1) ……

のように表される。

たとえば、N＝３００の場合を考えてみると

$$300 = 2^2 \times 3 \times 5^2$$

であるから、すべての約数は

$$\{1,2,3,4,5,6,10,12,15,20,25,30,50,60,75,100,150,300\}$$

のように与えられる。また、約数の合計S(N)や約数の個数T(N)は

$$S(300) = \frac{2^3 - 1}{2 - 1} \times \frac{3^2 - 1}{3 - 1} \times \frac{5^3 - 1}{5 - 1} = 7 \times 4 \times 31 = 868$$

$$T(300) = (2 + 1)(1 + 1)(2 + 1) = 18$$

のように算出される。）特に、（m,n）＝１である(つまりm，nが互いに素である)とき

S(m×n)＝S(m)×S(n)，T(m×n)＝T(m)×T(n)

である。

５．完全数・過剰数・不足数

ある自然数の（それ自身を除いた）全約数の合計をs(n)と表す。

s(n)＝n ⟶ nは「完全数」

s(n)＞n ⟶ nは「過剰数」

s(n)＜n ⟶ nは「不足数」

s(n)＝n－1　⟶　nは「概完全数」

s(n)＝n＋1　⟶　nは「準完全数」

例えば

6＝1＋2＋3

28＝1＋2＋4＋7＋14

496＝1＋2＋4＋8＋16＋31＋32＋62＋124＋248

であるから、6、28、496などは完全数である。

また、例えば

12＜1＋2＋3＋4＋6＝16

8＞1＋2＋4＝7

であるから、12は過剰数、8は不足数ということになる。

概完全数の例としては

8－1＝1＋2＋4

16－1＝1＋2＋4＋8

一般に、2の累乗は概完全数であるが、このタイプ以外の概完全数は知られていない。

また、準完全数は奇数の平方数でなければならないが、具体例は見つかっていない。

当然予想されるように、完全数は過剰数や不足数と比べて圧倒的に少ない。ただ、次のような命題の成り立つことが知られている。

もしも、$(1+2+2^2+2^3+\cdots\cdots+2^n)=2^{n+1}-1$が素数であるとすれば、

$$N=2^n(1+2+2^2+2^3+\cdots\cdots+2^n)=2^n(2^{n+1}-1)$$

は完全数である。また、偶数の完全数は全てこのタイプに限る。

例えば、$2^3-1=7$は素数であるから、$2^2(2^3-1)=4\times 7=28$は完全数である。同様に、$2^5-1=31$も素数であるから、$2^4(2^5-1)=16\times 31=496$も完全数になる。さらに、$2^6(2^7-1)=8128$や $2^{12}(2^{13}-1)=33550336$なども完全数である。

最後に、右引数で与えた整数のすべての約数を与える関数を次のように定義する。

```
    div=:13 :'/:~~.*/"1 d^"1#:i.2^#d=.q:y.'

    div 6
1 2 3 6
```

```
    div 20
1 2 4 5 10 20
```

div 8 1 2 4 8	div 220 1 2 4 5 10 11 20 22 44 55 110 220

さらに、右引数で与えた数以下の全ての合成数とその数の(それ自身を除いた)全約数の合計とをペアで表示する関数を次のように定義する。

```
 div_sum=:3 :0
if. y.<4 do.r=." else. r=.<(s=.4),3
  while.s<y. do.n=.#d=.div s=.s+1
    if.n=2 do. r else.r=.r,<s,+/}:d end.
  end.
end.
)
```

```
 div_sum 20
```

4 3	6 6	8 7	9 4	10 8	12 16	14 10	15 9	16 15	18 21	20 22

```
   2 9 $ div_sum 28
```

4 3	6 6	8 7	9 4	10 8	12 16	14 10	15 9	16 15
18 21	20 22	21 11	22 14	24 36	25 6	26 16	27 13	28 28

上の結果より

{6,28} は完全数

{4,8,16} は概完全数

{12,18,20,24} は過剰数

{9,10,14,15,21,22,25,26,27} は不足数

といったことが分かる。なお準完全数は無い。

6．オイラー(Euler)関数

$\Phi(b)$ を b 以下の正整数 a で b と互いに素なものの集合とし、$\Phi(b)$ の要素の個数を $\phi(b)$ と表して「オイラー関数」という。例えば

$$\Phi(7) = \{1,2,3,4,5,6\} \quad , \quad \phi(7) = 6$$

$$\Phi(9) = \{1,2,4,5,7,8\} \quad , \quad \phi(9) = 6$$

である。さらに

$$\phi(10) = 4 \, , \quad \phi(100) = 40 \, , \quad \cdots\cdots \, , \quad \phi(10^p) = 4 \times 10^{p-1}$$

一般に、 p が素数ならば

$$\phi(p) = p - 1,\quad \phi(p^2) = p(p-1),\quad \cdots\cdots,\quad \phi(p^k) = p^{k-1}(p-1)$$

といった関係が成り立つ。この性質を用いると

$$\phi(27) = \phi(3^3) = 3^2(3-1) = 18,\quad \phi(32) = \phi(2^5) = 2^4(2-1) = 16$$

といったように算出することができる。さらにm、nを互いに素とすると

$$\phi(mn) = \phi(m)\phi(n)$$

といった関係も示される。これより、p（＞３）を素数とすれば

$$\phi(3p) = \phi(3)\phi(p) = 2(p-1)$$

また、上記の『分解定理』を用いると、10^r という形の数のオイラー関数は

$$\phi(10^r) = \phi(2^r)\phi(5^r) = 2^{r-1}(2-1)\times 5^{r-1}(5-1) = 4\times 10^{r-1}$$

となることが分かる。

```
    euler_set=:3 :'(1=y.+.t)#t=.}.i.y.'
    euler_no=:[:#euler_set
```

```
    euler_set 7
1 2 3 4 5 6
    euler_set 9

1 2 4 5 7 8
    euler_set 12
1 5 7 11
    euler_no 40
16
    euler_no 100
40
    euler_no 10000
4000
```

```
    euler_no 7
6
    euler_no 9
6
    euler_no 12
4
    div_set=:13 :'(0=t|y.)#t=.1+i.y.'

    div_set 12
1 2 3 4 6 12
    }:div_set 12
1 2 3 4 6
```

７．オイラーの多項式

“41”から始まる整数列を四角の“渦巻状”に配置するとき、主対角線上に現れる数の多くが素数になることが知られている。

まず、数列を四角の“渦巻状”に配置するプログラムは、次のように定義される：

```
 aswirl=:3 :0
r=.(t=.''),s=.(k=.1){.y.
while.k<%:#y. do.
  r=.r,`(,~)@.(2:|[:#[)(|.^:k)t=.k{.y.-.s=.t,s
  r=.r,.~`,.@.(2:|[:#[)(|.^:k)t=.(k=.k+1){.y.-.s=.t,s
end.
)
```

さらに、素数(1)か否か(0)を判定する関数は、

```
 judge=:(1:=[:#q:)”0
```

のように極めてシンプルな形で与えられる。また、主対角線上に現れる数は

```
 dg_element=:13 :'/:~(<0 1)&|:aswirl 41+i.*:y.'
```

と与えられる。これは、$P(n)=41+n+n^2$ という２次の多項式(Euler の多項式)で与えられる数値と同じである：

```
 Euler=:41”_+]*1:+]
```

そこで、「aswirl」という関数を実行してみると

```
 aswirl 41+i.*:5
53 52 51 50 65
54 43 42 49 64
55 44 41 48 63
56 45 46 47 62
57 58 59 60 61
```

この主対角線上の要素は

dg_element 5	Euler i.5
41 43 47 53 61	41 43 47 53 61

といった関数で与えられる。

さて、主対角線上の要素が素数か否かを判定してみると

```
 judge aswirl 41+i.*:5
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
```

0 0 0 1 0

0 0 1 0 1

のように、全て素数であることが確かめられる。

```
   5 20 $ judge E=:dg_element i.100
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1
1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1
1 0 0 1 0 1 1 0 1 0 1 0 1 1 1 1 0 1 1 1
   40 41 42 { E
1681 1763 1847
   q: 1681
41 41
```

【「ｑ：」は、右引数で与えた数の“素因数分解”を求める関数である】

```
   q: 1763
41 43
```

【1681 や 1763 は合成数であることが分かる】

```
   q: 1847
1847
```

【1847 は素数である】

```
   judge 41+i.4
1 0 1 0
   (judge"0 r);r=.aswirl 41+i.9
```

1 0 0	43 42 49
0 1 0	44 41 48
0 0 1	45 46 47

```
   (judge"0 r);r=.aswirl 41+i.100
```

1 0 0 0 1 0 0 0 0 0	131 130 129 128 127 126 125 124 123 122 132 97 96 95 94 93 92 91 90 121

```
0 1 0 0 0 0 0 0 0 0
0 0 1 0 0 0 1 0 1 0
0 0 0 1 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0
0 1 0 0 0 1 0 0 0 0
1 0 0 0 0 0 1 0 0 0
0 1 0 0 0 1 0 1 0 0
1 0 0 0 1 0 0 0 1 0
0 0 0 1 0 1 0 0 0 1
```

```
133  98  71  70  69  68  67  66  89 120
134  99  72  53  52  51  50  65  88 119
135 100  73  54  43  42  49  64  87 118
136 101  74  55  44  41  48  63  86 117
137 102  75  56  45  46  47  62  85 116
138 103  76  57  58  59  60  61  84 115
139 104  77  78  79  80  81  82  83 114
140 105 106 107 108 109 110 111 112 113
```

8. 調和数

整数 n の全ての約数の調和平均が整数になるとき、「調和数」という。

全ての約数を出力する関数を次のように定義する。

```
   div=:13 :'/:~~.*/"1 d^"1#:i.2^#d=.q:y.'
```

さらに、調和平均を求める関数を次のように定義する。これより

```
   meanh=:[:%[:(+/%#)% NB.harmonic mean
```

そこで、右引数で与えた数以下の調和数を出力する関数を次のように定義する。

```
   h_number=:3 :0
if. y.<6 do.r=.1 else.
r=.1 6 [ s=.7+i.y.-6
  while. 0<#s
    do. h=.meanh div t=.{.s
        r=.r,((0:=]->.)h)#t [ s=.}.s
```

```
   h_number 30
1 6 28
   h_number 300
1 6 28 140 270
   h_number 1000
1 6 28 140 270 496 672
```

end. end.)	h_number 10000 1 6 28 140 270 496 672 1638 2970 8128 8190

h_number 100000

1 6 28 140 270 496 672 1638 2970 8128 8190 18600 18620 30240 32760 55860

【10 万以下では、調和数は 15 個しかない】

meanh div 6 2 meanh div 140 5 meanh div 496 5 meanh div 1638 9 meanh div 8128 7	meanh div 28 3 meanh div 270 6 meanh div 672 8 meanh div 2970 11 meanh div 8190 15

９．友愛数

mの除外約数の和がｎで、ｎの除外約数の和がｍとなるとき、（m, n）のペアを「友愛数」という。最も小さい友愛数のペアは（220,284）である。

```
div=:13 :'~./:~*/"1 d^"1#:i.2^#d=.q:y.' NB. 自分自身を除いた全ての約数を出力.
div_sum=:3 :0
if. y.<4 do.r=." else. r=.<(s=.4),3
 while.s<y. do.n=.#d=.div s=.s+1
  if.n=2 do. r else.r=.r,<s,+/d=:d end.
 end.
end.
)
```

(;[:+/]:)@div 220

1 2 4 5 10 11 20 22 44 55 110	284

(;[:+/]:)@div 284

1 2 4 71 142	220

(;[:+/]:)@div 1184

1 2 4 8 16 32 37 74 148 296 592	1210

(;[:+/]:)@div 1210

1 2 5 10 11 22 55 110 121 242 605	1184

```
    friend=:3 :0
s=.(-.;r-:L:0|.L:0 r)#r=.div_sum y.
{."1((-:#t),2)$t=.(s e.|.L:0 s)#s
)
```

friend 300

220 284

friend 1300

220 284	1184 1210

friend 3000

220 284	1184 1210	2620 2924

```
    6!:2 'r=.friend 3000'
6.74049
    6!:2 'r=.friend 10000'
110.877
```

r

220 284	1184 1210	2620 2924	5020 5564	6232 6368

１０．数の耐久性(persistence)

数列

｛６７９、３７８、１６８、４８，３２，６｝

では､各項が直前の数字の積になっている｡そこで､ある数の｢耐久性｣を１桁の数に退化するまでの回数であると定義する。

ある数から次のステップの数を出力する関数は次のように与えられる。

```
mpy=:[:*/[:".,"0":
```

Mpy 679	mpy 378	mpy 168	mpy 48	mpy 32
378	168	48	32	6

そこで､｢耐久性｣の値を求める関数を次のように定義する｡｢pst2｣は｢耐久性｣の値だけを出力する関数である。

```
Pst=:3 :0
r=.y.,s=.(mpy=.[:*/[:".,"0":)y.
while.1<#":s
  do.r=.r,s=.mpy s
end.
(<:#r);r
)
```

```
pst2=:3 :0
if.y.<10 do.0 else.
  #r=.s=.(mpy=.[:*/[:".,"0":)y.
  while.1<#":s
    do.#r=.r,s=.mpy s
  end.
end.
)
```

```
pst 679
```

5	679 378 168 48 32 6

```
pst2 679
5
```

さらに、与えられた数の耐久性を持つ最小の数を求める関数を次のように定義する。

```
min=:3 :0
r=.pst2 s=.1
while.r<y.
  do. r=.pst2 s=.s+1
end.
s
)
```

min 1	min 3	min 5	min 7

10 min 2 25	39 min 4 77	679 min 6 6788	68889 min 8 2677889

```
   6!:2'r=.min 9'
6740.35
```

【直上の演算時間には6740.35秒を要していることを示している！】

```
   r
26888999
   pst r
```

9	26888999 4478976 338688 27648 2688 768 336 54 20 0

次に、耐久性の値ごとの頻度分布を求める関数を次のように定義する：

```
   freq=:[:(i.@#,:])[:([:+/"1([:i.1:+>./)=/])pst2"0
```

freq i.679 0 1 2 3 4 10 185 265 186 33	freq i.680 0 1 2 3 4 5 10 185 265 186 33 1

この結果から、「679」が耐久性５の最小の整数であることが分かる。

freq i.6788 0 1 2 3 4 5 10 1904 2380 1725 659 110	freq i.6789 0 1 2 3 4 5 6 10 1904 2380 1725 659 110 1

freq i.68889 0 1 2 3 4 5 6 10 23210 25097 12901 6072 1407 192	freq i.68890 0 1 2 3 4 5 6 7 10 23210 25097 12901 6072 1407 192 1

次に、右引数で与えた値より１つだけ少ない耐久性をもつ数値のセットを与える関数を次のように定義する。

```
   less=:3 :0
s=.1+r=.#q=.''
while.r<y. do. s;q=.q,((y.-1)=r=.pst2 s)#s=.s+1
```

end.

)

less 3

39	25 26 27 28 29 34 35 36 37 38

less 4

77	39 47 49 55 57 59 66 68 69 74 75

$ r=.>{:less 5

33

3 11$ r

77 177 268 277 286 348 355 377 378 379 384
387 397 438 446 464 467 476 477 483 489 498
535 553 557 575 628 644 647 668 674 677 678

$ r=.>{:less 6

110

```
   10 11$ r
 679  688  697  769  796  868  886  967  976 1679 1688
1697 1769 1796 1868 1886 1967 1976 2379 2388 2397 2468
2486 2648 2684 2688 2739 2777 2793 2838 2846 2864 2868
2883 2886 2937 2973 3279 3288 3297 3367 3376 3448 3484
3488 3637 3673 3729 3736 3763 3792 3828 3844 3848 3882
3884 3927 3972 4268 4286 4348 4384 4388 4438 4446 4464
4468 4483 4486 4628 4644 4648 4682 4684 4779 4797 4826
4834 4838 4843 4846 4862 4864 4883 4977 5579 5597 5759
5795 5957 5975 6179 6188 6197 6248 6284 6288 6337 6373
6428 6444 6448 6482 6484 6677 6719 6733 6767 6776 6779
```

r の各数に対する次のステップ数を求める関数「mpy」を適用してみると

```
   10 11$  s =.mpy"0 r
 378  384  378 378 378  384 384  378  378  378  384
 378  378  378 384 384  378 378  378  384  378  384
 384  384  384 768 378  686 378  384  384  384  768
 384  768  378 378 378  384 378  378  378  384  384
 768  378  378 378 378  378 378  384  384  768  384
 768  378  378 384 384  384 384  768  384  384  384
 768  384  768 384 384  768 384  768 1764 1764  384
 384  768  384 768 384  768 768 1764 1575 1575 1575
1575 1575 1575 378 384  378 384  384  768  378  378
 384  384  768 384 768 1764 378  378 1764 1764 2646
```

ところで、例えば“378 に退化してしまう” r の要素は

```
   $ (378= s )#r
36
```

のように、36 個もある。そこで s の中で重複している要素を除いてみると、

```
   ~. s
378 384 768 686 1764 1575 2646
```

のようにたった 7 個になってしまう。