6.4 それでも速い方が...

完全数を見つけるマクロが遅い理由は、約数の調べ方にある。それは、律儀に $n-1$ までの割り算を試しているからだ。しかし前にも述べたように、n の約数を調べるには $\frac{n}{2}$ までの数で割ればよい。ところで n の約数が d だった場合、$\frac{n}{d}$ も同時に約数になっているので、基本的に約数は 2 個同時に見つかる。基本的にというのは、たとえば 49 の約数は 1 と 49 が同時に見つかったあとは、7 と 7 が同時に見つかることになるが、7 を 2 回数えるわけにはいかない。すると $d<\frac{n}{d}$ を満たす d が探すべき約数のすべてとなる。最初からこのように調べればよかったのだろうが、この方式では約数を小さい順に出力するのは難しい。単に約数を出力するなら、順番通りに出力する方が見た目にもよい。しかし完全数を調べる場合は、和が計算できればよいのであって、順番通りに足す必要はない。よって、$d<\frac{n}{d}$ を満たす d までの調査が効率的である。以上の視点でマクロを修正してみよう。

```
\makeatletter
\newcommand\pnfind[1]{%
  \newcount\n \newcount\d \newcount\m \newcount\sum
  \newcount\q
  \m=6
  \@whilenum\m<#1 \do{\d=2 \sum=1
  \loop
    \n=\m \divide\n\d \q=\n
    \ifnum\d<\q \multiply\n\d \advance\n-\m
      \ifnum\n=0{\advance\sum\d \divide\m\d \global\advance\sum\m}\fi
    \advance\d1
  \repeat
  \ifnum\sum=\m \number\m, \fi
  \advance\m1}%
}\makeatother
```

これで、『10000 未満の完全数は`\pnfind{10000}`である。』と書けば『10000 未満の完全数は 6, 28, 496, 8128, である。』が出力される。ちなみに、マクロ名は少し妙な命名であるが perfect number find にちなんだ。

修正は実質的に 1 か所と言ってよいのだが、そのためにいくつかの修正が必要となった。実質的な修正は n が d で割り切れたとき、`\sum` に $\frac{n}{d}$ も一緒に加算することである。それはグルーピングをした上で処理している。なぜなら、当初`\n` に代入されていた $\frac{n}{d}$ の値は、余りが `0` かどうか判定するために`\d` が掛けられて別の値になってしまっている。したがってもう一度 $\frac{n}{d}$ の値を得たいのだが、それは`\m` を`\d` で割って求めるしかない。しかし、`\m` の値はその後も必要とするのでグルーピングによって保護されなくてはならなかった。

このように\m や\n の値が変化してしまうため、\d が $d < \frac{n}{d}$ の条件を満たしているかを知るために、$\frac{n}{d}$ の値を保持する必要に迫られた。それで変数\q が追加されているのである。そして、ここが微妙なところなのだが、\d が満たす条件が $d \leqq \frac{n}{d}$ ではなく $d < \frac{n}{d}$ であることだ。\n が\d でちょうど割り切れたとき、d と $\frac{n}{d}$ を一緒に\sum に加えるのだが、たまたま $n = d^2$ であると d が 2 回加えられてしまう。これを回避するためには、$d < n$ の判定を早めに終えておかなくてはならない。そこで、\ifnum\d<\q の判定を前に持ってきたのだ。すると平方数は調べられることなく飛ばされるけれど、完全数が $2^{n-1}(2^n - 1)$ の形をしていることに甘えさせてもらおう。厳密にやりたいならば、平方数が飛ばされないようにコードを追加してほしい。

あとひとつおまけの修正がある。以前は約数を探すとき\d=1 から割っていたが、それは\d=2 から割り始めることにし\sum に 1 を与えておいた。1 の相手になる約数は n だが、n は加算しないからである。

さて、これでだいぶ速く処理ができるようになった。ほとんど待たされることなく 10000 までの完全数を見つけられる。だったら、もうひとつ先の完全数を探すこともできそうな気がするだろう。でも、やめておいた方がよい。なぜなら、それは 8 桁の整数だからだ。TEX は 9 桁強の整数までが守備範囲とはいえ、最低でも 10000 倍多くの数を調べなくてはならない。たとえ 8128 を 0.1 秒で見つけられたとしても、次の完全数を見つけるまでに 1000 秒（16 分 40 秒）もかかってしまう。それは TEX に失礼というものだ。

では、今度は友愛数を見つけるマクロを速くしよう。

```
\newcount\n \newcount\d \newcount\m \newcount\sum
\newcount\q
\newcommand\addmeasures{%
  \d=2 \sum=1
  \loop
    \n=\m \divide\n\d \q=\n
    \ifnum\d<\q \multiply\n\d \advance\n-\m
      \ifnum\n=0{\advance\sum\d \divide\m\d \global\advance\sum\m}\fi
    \advance\d1
  \repeat
}
\makeatletter
\newcommand\fnfind[1]{%
  \newcount\t
  \m=6
  \@whilenum\m<#1 \do{%
    \t=\m \addmeasures \m=\sum \addmeasures
    \ifnum\sum=\t \ifnum\t<\m (\number\t, \number\m) \fi\fi
    \m=\t \advance\m1}%
}\makeatother
```

これで、『10000 未満の友愛数は`\fnfind{10000}`である。』と書けば『10000 未満の友愛数は (220, 284) (1184, 1210) (2620, 2924) (5020, 5564) (6232, 6368) である。』が出力される。

約数を加算するルーティンは関数`\addmeasures`として独立させた。こうしておけば使い回しができる。肝心のマクロ`\fnfind`（friend number find にちなんだ）は、約数の和を 2 回調べるようになっている。単純に`\addmeasures`を 2 回実行するだけである。約数の和を求める部分を効率化したために、10000 までの友愛数を探しているにもかかわらず、以前 3000 までの友愛数を探したときより処理が短くなったことが実感できるだろう。プログラミングというものは単純な構造ほど分かりやすくてよいが、多少複雑になっても処理速度が格段に改善されるなら、コードの追加はすべきだという例である。

さあ、あとひとつ。婚約数を見つけるマクロも速くしよう。

```
\newcount\n \newcount\d \newcount\m \newcount\sum
\newcount\q
\newcommand\addmeasuresB{%
  \d=2 \sum=0
  \loop
    \n=\m \divide\n\d \q=\n
    \ifnum\d<\q \multiply\n\d \advance\n-\m
      \ifnum\n=0{\advance\sum\d \divide\m\d \global\advance\sum\m}\fi
    \advance\d1
  \repeat
}
\makeatletter
\newcommand\fnfindB[1]{%
  \newcount\t
  \m=6
  \@whilenum\m<#1 \do{%
    \t=\m \addmeasuresB \m=\sum \addmeasuresB
    \ifnum\sum=\t \ifnum\t<\m (\number\t, \number\m) \fi\fi
    \m=\t \advance\m1}%
}\makeatother
```

これで、『10000 未満の婚約数は`\fnfindB{10000}`である。』と書けば『10000 未満の婚約数は (48, 75) (140, 195) (1050, 1925) (1575, 1648) (2024, 2295) (5775, 6128) (8892, 16587) (9504, 20735) である。』が出力される。

ついさっき、`\addmeasures`を独立した関数にしたのだけれど、婚約数では`\sum=0`から加算しなくてはならない。このわずかな違いのため、`\addmeasures`は別の関数`\addmeasuresB`となってしまった。計画性に欠けることこの上ない。もし、計画的に事を運んでいる印象を与えたいなら、`\addmeasures`に`\sum=1`を記述せず、`\addmeasures`を使う直前に`\sum=1`か`\sum=0`を記述すればよいだろう。`\addmeasuresB`は不要になる。

ところで以前、友愛数と婚約数ではおもむきの違うペアが現れることを指摘した。この道中では、双子素数のときもそうであったように、n 未満のペアを探すと言えば、それは一方が n 未満であることを指していて、もう一方が n を超えることもあった。実際 10000 未満と言ったとき、友愛数はすべて 10000 未満のペアであったが、婚約数はもう一方が 10000 を超える 2 ペアが見つかった。なぜそうなのかは分からないが、その予兆は 3000 未満で現れていた。それはペアの比に見ることができる。友愛数の 2 数の比はそれぞれ 1.29, 1.02, 1.16 であったのに対し、婚約数の 2 数の比はそれぞれ 1.56, 1.39, 1.83, 1.05, 1.13 と総じて高いようだ。この傾向がこの先も続くかどうかは知らないけれど、約数 1 の扱いでこのような差があるとすれば興味深いことである。