## 4.3 双子素数

素数の話題は尽きることがない。話題のひとつに**双子素数**というものがある。2 を除けば素数は奇数であるが、ときどき 17, 19 や 41, 43 のように連続して素数が現れることがある。これらを双子素数と呼んでいる。

素数が無数にあることは証明されている。それはユークリッド[1]による証明が有名である。証明の大筋は次のようなものだ。

素数が有限個しかないと**仮定**し、素数を $2, 3, 5, \ldots, P$ とする。このとき $2 \cdot 3 \cdot 5 \cdots P + 1$ なる数を作れば、この数はどの素数で割っても 1 余る。これは $P$ より大きい新たな素数があることを意味し、素数が $P$ までしかないことに矛盾する。この矛盾は素数が有限個であると仮定したことに起因している。すなわち素数は有限個ではない。

別の視点では、$2 \cdot 3 \cdot 5 \cdots P + 1$ は有限個の素数とは異なる数だから合成数で、ある素数で割れるはずである。既存の素数では割れないので、別の新たな素数で割れなければならない。やはり矛盾である。

このように、ある仮定から始めて矛盾を引き出し、結果、仮定が間違っていると結論する証明方法は**背理法**と呼ばれる。

ところで、素数がどれぐらいの頻度で現れるかとか、双子素数も無数にあるのかなど、素数には未解決の問題が多い。素数の頻度というのも変な感じであるが、ガウス[2]は素数が現れる頻度について、次のような見解を持っていた。

$$1 \text{ から } N \text{ までには、素数はおよそ} \frac{N}{\log_e N} \text{個含まれる}$$

たとえば $N = 100$ とすると、

$$\frac{100}{\log_e 100} \approx 21.7$$

だが、実際は 25 個の素数がある。ぴったりではないが、まあ近いところをついている。式は $N$ が大きいほど、より正しい頻度を教えてくれる。もし $N$ を与えたとき、本当にぴたり正確な素数の数を求める式が発見されれば、画期的なことであるが、いまのところそのような式は発見されていない。またガウスの式とは別の、素数の頻度を計算する式はあるが、ガウスの式の美しさにはかなわないのだ。何とも不思議なことである。

---

[1] ユークリッド（330?B.C.–275?B.C.）：ギリシアの数学者。
[2] カール・フリードリヒ・ガウス（1777–1855）：ドイツの数学者。

双子があれば三つ子も考えたくなるのが人情で、3, 5, 7 のような三つ子素数がどれくらいあるかと考えてしまう。だが結論を言おう。これ以外の三つ子素数はない。連続する 3 つの奇数は三つ子素数にはならないのだ。少し考えれば理由は分かるだろう。

三つ子素数がないと分かったところで、双子素数に集中しよう。と言っても場当たり的に双子素数を探すのでは効率が悪い。徹底した調査で、どのような条件のとき双子素数が現れるのかが分かれば、マクロも書きやすくなる。そこで、ここでは次の単純な事実をもとに、双子素数を見つけていこう。

2, 3 を除くすべての素数は $6m \pm 1$ の形をしている

そう、素数は決して $6m \pm 2$ や $6m \pm 3$ のような形をしていないのだ。誤解しないでもらいたいことがある。私は「素数は $6m \pm 1$ の形をしている」と主張しているのであって、決して「$6m \pm 1$ の形をしている数は素数である」と言っているのではない。**逆**は必ずしも正しくないとは、まさにこのことを言う。そして、ここでも少し考えれば、すべての素数が $6m \pm 1$ の形をしている理由も分かると思う。

さて、そういうことなら話は簡単だ。$6m - 1$ が素数であるかどうか調べ、素数であるときに限り $6m + 1$ が素数であるかどうか調べればよい。運良く 2 つとも素数なら、それが双子素数ということだ。

```
\newif\ifprime \newif\ifunknown
\newcount\n \newcount\p \newcount\d \newcount\a
\def\testprimality{{\d=3 \global\primetrue
  \loop \trialdivision \ifunknown \advance\d by2 \repeat}}
\def\trialdivision{\a=\p \divide\a by\d
  \ifnum\a>\d \unknowntrue \else \unknownfalse \fi
  \multiply\a by\d
  \ifnum\a=\p \global\primefalse \unknownfalse \fi}
%
\newcount\q
\newcommand\twinp[1]{(3, 5)\n=6
  \loop \p=\n \advance\p-1
  \ifnum\p<#1 \printiftwin \advance\n6 \repeat}
\newcommand\printtwinp{, (\number\q, \number\p)}
\newcommand\printiftwin{\testprimality
  \ifprime \advance\p2 \testprimality
  \ifprime \q=\p \advance\q-2 \printtwinp \fi\fi}
```

これで、『100 未満の双子素数は\twinp{100}である。』と書けば『100 未満の双子素数は (3, 5), (5, 7), (11, 13), (17, 19), (29, 31), (41, 43), (59, 61), (71, 73) である。』が出力される。100 未満の判定が双子素数のどっちと比べているかというと、小さい方と比べている。したがって、\twinp{101} ではこれと同じ出力になるが、\twinp{102}だと (101, 103) が追加されることになる。

マクロは、まず素数判定用の関数\testprimality と\trialdivision を TEXbook から拝借している。私が定義したマクロは\twinp とそのサブルーティンの部分である。では、\testprimality が素数を見つける方法について話しておこう。

たとえば 49 が素数かどうか判定するには、2 から 48 までの数で割ってみればよい。何ひとつ割ることができなければ 49 は素数だ。しかし考えてみてほしい。素数は 2 を除いてすべて奇数である。奇数は偶数で割れっこない。したがって偶数での割り算を試す必要はないのだ。その結果、49 が素数かどうか調べるには、3 から 47 までの奇数で割ってみるだけでよい。これで計算量を半分に減らせる。

さらに、$n$ が素数かどうか調べるのに、$n-1$ までの数で割る必要などない。結論を言えば $\sqrt{n}$ までの数で割るだけで十分である。たとえば 51 は 3 で割れる—商は 17 である—から、同時に 17 でも割れてしまう。数 $n$ が $p$ で割れるなら、同時に $\frac{n}{p}$ でも割れているのだ。このことから $n$ が素数かどうか調べるのに、$\sqrt{n}$ までの数で割ればよいことが結論できる。以上の考察によって、素数を判定するアルゴリズムは次のようになる。

a) 奇数 $n$ だけを対象に素数かどうかを調べる

b) 候補となる $n$ を 3 から $\sqrt{n}$ までの奇数で割る

c) 割れれば合成数、割れなければ素数 → b) へ戻る

しかし、\testprimality はこの手順と少し違う。割り算をする変数\d は、最大\n まで試されることになる。マクロの書きやすさと処理の効率は天秤にかける必要があるが、この場合は効率より書きやすさを選択したようだ。ひとくちに素数かどうか調べるといっても、マクロの書き方はいろいろあるのだ。

さて関数を拝借したおかげで、双子素数を見つける関数は大変簡素になった。拝借した関数には多くの変数が用いられているので、追加する必要がある変数は\q だけで済んだ。調べる素数は $6m \pm 1$ の形だが、正直に 6 倍して 1 を加減する代わりに、\n を 6 から始めて 6 ずつ増やすようにした。同じ処理が少ない記述で済む。

\printiftwin が双子素数であることを判定する関数である。双子素数は、連続する奇数がともに素数になったときに意味を持つので、はじめに判定に回ってくる $6m-1$ が素数でなければ次に行きたい。実際\printiftwin は、$6m-1$ が素数のときに限り $6m+1$ を調べることにしている。この作戦は処理を減らす効果がある。

\printiftwin によって候補 2 数が素数であることが判明すると、\printtwinp が 2 数を出力する。ただし\printtwinp が出力できるのは 5 以上の素数に限る。それは\testprimality が 5 以上の奇数しか判定しないからである。ところがいちばん最初の双子素数は (3, 5) だ。もっとも、これは自明であるためマクロ\twinp が仕事を始める前に出力してしまう。何も律儀に調べる必要などないのだ。それに、このように割り切って考えたために、“,” 区切りの出力に悩まなくて済んだ。

調べる上限はマクロ実行時に与えられる引数の値\n だが、双子素数の性質上、$n+1$ まで滑り込む可能性があることはさっき述べた通りだ。また、\printtwinp が出力する\q と\p の順番が逆っぽいが、TEXbook の関数で\p が用いられていた関係で、このような出力が無難だと思ったのだ。