ミラー-ラビン素数判定法と強偽素数

a を底とするフェルマーテストを行う際，$a^{n-1} \mod n$ という計算を行う．この値が 1 ならフェルマーテストは合格するというわけである．しかし実際にコンピュータで計算するときに a^{n-1} という計算を行うわけではない．もしそうすればすぐに桁あふれ（オーバーフロー）を起こしてしまう．だから a を一つずつかけて行き，その都度 n で割った余りを出すわけである．そのため，プログラム上でこの部分に最も多くの時間を費やす．それではもっと時間短縮ができないだろうか．例えば

$$a^{n-1} \equiv 1 \pmod{n}$$

が成り立つならば

$$a^{\frac{n-1}{2}} \equiv \pm 1 \pmod{n}$$

が成り立たないだろうか．$n-1$ は通常奇数だし，もし成り立てば計算は半分ですむという発想から議論を進めよう．

定理 1 $a^2 \equiv 1 \Longleftrightarrow a \equiv \pm 1 \pmod{p}$

［証明］$\Longleftarrow$ が真であることは自明なので，$\Longrightarrow$ が正しいことを証明する．

もし $a^2 \equiv 1$ かつ $a \not\equiv \pm 1 \pmod{p}$ であるような a があったと仮定すると，$a^2 \equiv 1$ より

$$a^2 - 1 \equiv 0$$

$$(a+1)(a-1) \equiv 0$$

$$\therefore (a+1)(a-1) = kp$$

p は素数なので $p|a+1$ または $p|a-1$ となり矛盾する．よって証明された．　［証明おわり］

これは至極当たり前の定理なのだが，合成数を法とする場合はこのようにはいかない．もちろん

$$a^2 \equiv 1 \Longleftarrow a \equiv \pm 1 \pmod{n}$$

は成り立つが，この逆は成り立たない場合もある．たとえば

$$a^2 \equiv 1 \pmod{15}$$

の解は

$$a \equiv \pm 1, \pm 4 \pmod{15}$$

の 4 個である．それでは合成を法とする場合で，次のような命題はどうであろうか．

$$p^{2m} \equiv 1 \Longrightarrow p^m \equiv \pm 1 \pmod{n}$$

残念ながら偽である．たとえば，

$$2^{644} \equiv 1 \pmod{645}$$

であるが，

$$2^{322} \equiv 259 \pmod{645}$$

となり成り立たない．結局冒頭のようなフェルマーテストを半分の労力で行うということは不可能であることがわかった．しかしながら，考え方を変えればこのようなテストをすれば，フェルマーテストより強力な素数判定チェックを行えるわけである．ここで反例に挙げた 645 は 2 を底とする偽素数である．つまりフェルマーテストを通っているわけだが，

$$2^{\frac{n-1}{2}} \equiv \pm 1 \pmod{n} \tag{1}$$

が成り立たず，素数ではないことがわかる．このことは有効である．たとえば 100000 以下の 2,3,5,7 を底とする偽素数の共通部分となる偽素数は次の三つしかない．

$$29341, 46657, 75361$$

このうち最初の 29341 は (1) は成り立つのだが，5 を底とする同様のテストに合格しない．

$$5^{\frac{29340}{2}} \equiv 25011 \pmod{29341}$$

ほかの 2 数は，このテストを通過し，他の底で同様のテストを行っても通過する．

このようなテストをさらに発展させたものが**ミラー-ラビン素数判定法 (Miller-Rabin primality test)** と呼ばれるもので，このテストを通った合成数を**強偽素数 (Strong pseudoprime)** と呼ぶ．先ほどの 2 番目の偽素数 46657 を例にとって述べよう．フェルマーテストは通過しているので，当然

$$2^{46646} \equiv 1 \pmod{46657}$$

である．さらに

$$2^{\frac{46646}{2}} \equiv 1 \pmod{46657}$$

ここまでは先ほどの (1) に相当するが，$\dfrac{46656}{2}$ はさらに 2 で割れるので，

$$2^{\frac{46646}{4}} \equiv 1 \pmod{46657}$$

さらに続けると

$$2^{\frac{46646}{8}} \equiv 23570 \pmod{46657}$$

と，ここで化けの皮がはがれるわけである．

偽素数 75361 で調べてみよう．

$$2^{\frac{75360}{8}} \equiv 39898 \pmod{75361}$$

これも合格しない．つまり，5 桁以下の強偽素数（2,3,5,7 を底とする場合の積集合の要素）は一つも存在しないことがわかった．2,3,5,7 を底とする強偽素数の共通部分の最小の強偽素数はなんと 10 桁で，

$$3215031751 = 151 \times 751 \times 28351$$

ということがわかっている．

2 を底とする強偽素数をコンピュータを用いて求めてみよう．冗長部分を解説すると，$\dfrac{n-1}{2}$ 回の計算を行ったあと，すでに終わったはずの $\dfrac{n-1}{4}$ 回の計算などを再び行う部分がある．もちろん改善できるが，偽素数そのものがたまにしか現れないので全体の計算時間にさほど影響を与えていないはずである．

```
#prompt
DIM CELLS(100000) As Long
dim Mi As Long,n as  Qword,S as long, i as long, Pii as long,PIii as long,Piii as long
dim p as Qword,pii as long ,a as Qword,nn as long,nnend as long
Open "2を底とする強偽素数.txt" For Output As #1
LET CELLS(1) = 2
LET Mi = 1
FOR n = 3 TO 1000000 step 2
   LET S = Sqr(n)
   For i = 1 To n - 1 '実際は最後のn-1まで繰り返すことはなく，必ず途中でループを抜け出す．
      LET p = CELLS(i)
      If p > S Then GoTo 5 'nが素数ということが決まったので，配列に記憶して次のnに．
      If n MOD p  = 0 Then GoTo 10 '合成数なので偽素数かのチェックに移る．
   Next i
5    'PRINT n,
     LET Mi = Mi + 1 'この部分は素数のストックをするところ．前のif文の中に入れてもよい．
     LET CELLS(Mi) = n
goto 20
10 'print n
if p<>2 then
         nnend=(n-1)/2 '繰り返しの終わりをきめる．
15 a=1
FOR nn = 1 TO nnend
             LET a = a * 2 mod n '実際はこの部分が時間をとっているものと思われる．
         NEXT nn
          IF a =1 then
if nnend mod 2 =0 then
nnend =nnend/2
goto 15
Else
print n
print #1,n
end if
End If
if a= n-1 then
print n
print #1,n
end if
end if
20   Next n
```

2047 3277 4033 4681 8321 15841 29341 42799 49141 52633 65281 74665 80581 85489 88357 90751 104653 130561 196093 220729 233017 252601 253241 256999 271951 280601 314821 357761 390937 458989 476971 486737 489997 514447 580337 635401 647089 741751 800605 818201 838861 873181 877099 916327 976873 983401

1000000 以下の 2 を底とする強偽素数は全部で 42 個ある．実行時間は数時間であった．睡眠中に計ったので正確な時間はわからない．赤字はカーマイケル数である．フェルマーテストによる偽素数と比べてカーマイケル数の占める割合はかなり減る．

参考文献

[1] 「Wolfram Mathworld」<http://mathworld.wolfram.com/>

[2] 「ウィキペディア」<http://ja.wikipedia.org/wiki/>