素数生成と素因数分解

素数の生成について

- 素数は無限個存在する
  ユークリッドの証明が有名
- 素数を組織的に生成する方法は知られていない（例えば、n番目の素数を与える関数$p(n)$とか）
- 大きな素数を求める方法
  1. 所望のサイズの乱数nを生成する
  2. 数nが素数かどうかをチェックする
  3. 素数であればnを出力。そうでなければ1.に戻る
- （おまけ）既知の最も大きな素数
  $2^{82589933}-1$（24,862,048桁）

素数の個数（素数定理）

- 数xまでの素数の個数を$\pi(x)$と書く
- この関数は$\pi(x) \sim \frac{x}{\log x}$と近似できることが知られている
- これは$\frac{\pi(x)}{x} \sim \frac{1}{\log x}$と変形すると、$x$までの数の中の**素数の割合が約$\frac{1}{\log x}$**であることがわかる
- 例えば1024ビットの素数は、$\frac{1}{\log 2^{1024}} = \frac{1}{710}$より、710個に1個の割合で存在することになる

素数の生成アルゴリズム(1)

- 自明な方式（**試行割算法**）
  数nが素数かどうかは$\sqrt{n}$以下の全ての数で割ってみれば分かる
  - エラトステネスのふるい（$\sqrt{n}$*までの素数で割ってみる*）
  - 10進数308桁程度(1024bit)の素数チェックには
    5.6×10^{151}回程度の割算が必要→不可能
    （数nのサイズに関する指数オーダーの計算量）
  - 10進数6～7ケタ(100万)程度まででであれば試行割算法が速い

素数の生成アルゴリズム(2)

- フェルマーテスト
  <u>フェルマーの小定理を利用する</u>
  　ｎが素数であれば、$a^{n-1} = 1(mod\, n)$
  　が全ての$a < n$ について成り立つ
  →フェルマーの式を成立させない数aが一つでも見つかれば、nは素数ではない
  - 多くの数aに対してフェルマーの式が成立すれば、nは（ほとんど間違いなく）素数と言えるか？
    →Ｎｏ！
  - ほとんど全ての数aに対してフェルマーの式を成立させるnが存在する
    （カーマイケル数）

フェルマーテストの例

素数かどうかチェックしたい数nに対して、
$a^{n-1} = 1\ (mod\ n)$かどうかを調べる

n \ a	383	25	299	230	264
499 (素数)	1	1	1	1	1
817 (合成数)	729	809	102	729	216
561 (カーマイケル数)	1	1	1	1	528

素数の生成アルゴリズム(3)

- ミラー・ラビンテスト
  カーマイケル数の問題を修正した方法
  →数nが合成数であるのに1回のミラー・ラビンテスト
  をパスできる確率は1/4
  →t回のテストを行えば 確率は$(1/4)^t$
  　t=10のとき、この確率は約100万分の１

  [素数を生成する手順]
  1. 所望のビット数の乱数nを作る
  2. 数nを小さな素数（100万程度まで）で割ってみる
     →割り切れたら１に戻る
  3. 数nをミラー・ラビンテストでチェック
     →パスしなかったら１に戻る

素因数分解アルゴリズムと計算量

- 試行割算法

  数十ビット程度が限界

- 楕円曲線法

  計算量：　$o(\exp(c\sqrt{\log p \cdot \log\log p}))$

- 一般数体ふるい法

  計算量：　$o(\exp(c(\log n)^{1/3}(\log\log n)^{2/3}))$

計算量が $L_n(u,v) = \exp(v(\log n)^u(\log\log n)^{1-u})$ （$0<u<1$ ）となるものを

準指数時間アルゴリズムという

$L_n(0,v) = (\log n)^v$　　$u=0$のとき：多項式時間アルゴリズム

$L_n(1,v) = \exp(v\log n)$　　$u=1$のとき：指数時間アルゴリズム

（参考）２０１０年に、一般数体ふるい法で768bitの数が素因数分解されている（ＮＴＴなど）。現在(2020)の記録は829bit