一般論文
受付：2005.9.12
受理：2006.1.12

Microsoft Excel 上に大きな整数とその演算関数を導入する

野　呂　春　文
日本福祉大学　情報社会科学部

The introduction of Big Integer and its arithnmetic functions into Microsoft Excel

Harufumi　Noro
Faculty of Social and Information Sciences, Nihon Fukushi University

Abstract: This report explains how to introduce so-called BigInteger arithmetics into Microsoft Excel. Here, term BigInteger means such integer that has non-limited digits except for the limit of memory size. A BigInteger is represented as an array of 10,000-based Long integers in Visual Basic on Microsoft Excel. Operations, such as addition, multiplication, division and so on, are defined as functions written by Visual Basic. Some functions about numerical theory are also presented.

Keywords: Microsoft Excel，Visual Basic，Big Integer，arithmetics，numerical theory

１．はじめに

そのすぐれた操作性と明示性から，Microsoft Excel の上には，付属する Visual Basic によるプログラムを併用して，多くのプログラム・パッケージが実現されている．簿記・会計，税務，さらに介護保険請求業務等の実用的プログラムもさることながら，統計学，解析学，経済学等を学ぶための教育的なプログラムも多種類開発されている．それらの中で，従来より望まれていたにも関わらず，数論に関するプログラムはごく限定的なものだけにとどまっている[1)]．

その原因は，Microsoft Excel 上では大きな桁数を持つ整数とその演算が不可能なことにある．ワークシートのセルに入出力できる整数は，-999 9999 9999 から 999 9999 9999 に限られ，その範囲を超える整数は浮動小数点数で近似される．MOD( ) 等の整数の演算に関するワークシート関数で処理できる上限は，2 6843 5455 (= 2^28 - 1) である．また，付属する Visual Basic では，Long 型整数が 32 ビットの大きさであるため，表現できるのは –21 4787 3648 から 21 4787 3648 (-2^31 から 2^31 - 1) に限られている．

数論や，それに基礎をおく暗号学においては，数 10 桁から数 100 桁の整数の実現とその演算が不可欠であり，これらの制限の範囲内では，極めて限定的，初歩的な問題にとどまらざるをえない．なお，このような，大きな桁数を持つ，あるいは，桁数制限のない整数は「多倍精度整数」，「多倍長整数」などと呼ばれることがある．ここでは，Java の BigInteger クラスにならって，単純に「大きな整数」と呼ぶことにする．

もちろん，数論や暗号学の本格的な研究を目指すには，計算速度の点から見ても，UBasic[2)]，GAP システム[3)]，あるいは，Java の BigInteger クラスライブラリー等が必要であることは言うまでもない．しかし，それらは 1 つずつの独立したプログラミング言語であり，使いこなすためには，ある程度の修練が不可欠で，初学者

がいきなり取り組むには無理がある．それにひきかえ，Microsoft Excel は初めて PC に向かう人でも使えるように，操作性や明示性が洗練されていて，教育用のソフトの土台に適している．このことは見落とされがちなのであるが，セルに計算式を記述し，その結果がただちに表示されるため，計算の途中経過が常に目に見えることは，特に教育用としてすぐれている．

そこで，今回 Microsoft Excel 上に桁数制限の無い大きな整数とその演算の実現を試みた．その目的は，主として初学者が自ら数論や暗号学を学ぶための基盤整備にある．実際のプログラミングは，Microsoft Excel に付属している Visual Basic (Excel VBA) でおこなった．言語仕様の制約により，数論の高度な計算等においては若干不満な点も残るが，20 から 30 桁の合成数の楕円曲線による素因数分解[4] 程度は可能なので，ここで発表し，諸氏による批判と助言を得たいと思う．

なお，本文中での数は，桁数の小さなものはべたで表わし，桁数が自明でないような大きなものは最下位桁から 4 桁ごとに空白で区切って表わしている．このようにすると桁数がわかりやすいことに加えて，次に説明する大きな整数の内部表現と関係しているからである．

２．大きな整数の実現

P を正の整数とするとき，P を基数として整数 A は，A = A_m×P^m-1 + A_m-1×P^m-2 + … + A_2×P + A_1，0 ≦ A_k < P，A_m ≠ 0 と表わされる．これを，整数 A の P 進数表現という．なお，P^n は P の n 乗である．A_1 から A_m を適当な整数型の配列に格納すれば，どのように大きな桁数の整数でも表現できる[5]．例えば，Java の BigInteger クラスでは，P = 2^15 = 32768 とし，各 A_k を 32 ビット整数型の配列に格納している．

今回 Excel VBA 上で開発したプログラムでは，P = 10000 としている．P をこのようにおくと，例えば，A = 20 5465 9681 6316 7672 9626 8561 は，A_1 = 8561，A_2 = 9626，A_3 = 7672，A_4 = 6316，A_5 = 9681，A_6 = 5465，A_7 = 20 と表わすことができる．それぞれを格納するための整数型としては，Long 型を採用する．Long 型を採用すると 1 個の数を表わすために桁数と同じだけのバイト数を使いメモリーの点でロスが発生する．しかし，A_k×A_m の計算において桁あふれ等の問題が起こらずプログラミングが容易になる．

大きな整数のプログラムの内部的表現においては，数を表わす配列の先頭部分に各種の操作を手助けするための情報を格納する．配列の名称を仮に A( ) として内容を示す．

A(0)：　正負を表わすために，正なら +1，負なら −1 を格納する．

A(1)：　総桁数を格納する．

A(2)：　数を表わすために使っている Long 型整数の個数を格納する．

A(3)：　配列の最上位の Long 型整数に格納されている数の桁数を格納する．

A(4)：　数の最下位の 4 桁までを格納する．以下，必要なだけ，数を格納する Long 型整数が並ぶ．なお，負の数の場合，A(0) に符号を格納すると共に，数を表わす各配列にも負の数を格納する．

例を示す．A = -20 5465 9681 6316 7672 9626 8561 とする．

A(0) = -1，A(1) = 26，A(2) = 7，A(3) = 2，A(4) = -8561，A(5) = -9626，A(6) = -7672，A(7) = - 6316，A(8) = -9681，A(9) = -5465，A(10) = -20 となる．これを「大きな整数の内部表現」と呼ぶ．

A = 0 は，A(0) から A(4) まで 0 の配列とする．言うまでもないが，配列の各要素には数値が格納されているのであって，「数字」が格納されているのではない．

３．各種の演算

3.1　入出力の仕様

各種の演算を行なう関数を定義するために，初めに入出力の仕様を定義する必要がある．すでに述べたように，Microsoft Excel のワークシートのセルに入出力できる整数は，-999 9999 9999 から 999 9999 9999 に限られ，その範囲を超える整数は浮動小数点数で近似される．したがって，この問題を解決しないと，付属する Excel VBA でプログラムを書いたところで，ほとんど実用性が無いことになる．

この問題の解決法は，入出力を文字列でおこなうことである．数字を文字列としてセルに入力するには，セルの書式設定機能によって「表示」を「文字列」に設定する．セルに入出力できる文字列の文字数は十分に大きい．ただし，255 文字を超えると，当該のセルにはシャープ記号による代替表示がなされるが入出力は正確に行なわれ，正しい値が数式バーに表示される．

したがって，ユーザーがワークシートのセルに記述して使う関数（ユーザー関数）における大きな整数の流れは次のようになる．

セル上で大きな整数を文字列として入力する　⇒　大きな整数の内部表現に変換して VBA によるプログラムで処理をおこない，結果を整数の文字列に変換する　⇒　セルに返して出力する．

このようなユーザー関数は，ワークシート上で使えるだけでなく，Excel VBA のプログラム中で使うこともできる．大きな整数の内部表現を直接扱う関数は，使い方が難しいので，複雑なアルゴリズムを直接記述するには適さない．一方，ユーザー関数は一部を除いて文字列を引き数に取り文字列を返す，という単純なインターフェースを持つため，プログラムが書きやすくなる．

ユーザー関数以外に，大きな整数の内部的表現を直接扱う関数が必要である．それは，ユーザー関数を裏から支えて実際の計算を実行する．これらの関数はワークシートからは使えず，Excel VBA によるプログラムの中でのみ使える．そのためシステム開発を行なうユーザーが使うだけなので，この報告では説明を省略する．なお，付録の関数一覧では簡単に紹介している．

これら二群の関数の間を取り持つのが，セル上で入力された大きな整数の文字列を内部表現に変換する関数である StrToBigInt( ) と，数の内部表現を整数の文字列に変換する関数 BigIntToStr( ) である．これらもユーザーが直接使うことはないが，今回開発した関数群のかなめであるため名称のみ紹介しておく．

3.2　単項演算を行なう関数と条件判断を行なうために使う関数

初めに単項演算を行なう関数について簡単に説明する．

（1）BigIntAbs(String) As String：　引き数として与えた大きな整数の文字列の絶対値の文字列を返す．

（2）BigIntNegate(String) As String：　引き数として与えた大きな整数の文字列の符号を反転した数の文字列を返す．

条件判断を行なうために使う関数について簡単に説明する．これらは関数の値を他の用途に使うことが無いため，Long 型整数を返す．

（1）BigIntCompare(aString, bString) As Long：　引き数として与えた大きな整数の文字列の大小を比較し Long型整数の値を返す．aString > bString なら 1 ，aString = bString なら 0 ，aString < bString なら –1 を返す．

（2）BigIntParity(String) As Long：　引き数として与えた大きな整数の文字列の偶奇を Long型整数で返す．偶数なら0 ，奇数なら 1 を返す．

（3）BigIntSign(String) As Long：　引き数として与えた大きな整数の文字列の正負を Long型整数で返す．正なら 1 ，負なら –1 を返す．

3.3　加法，減法に関連する関数

加法，減法に関連する関数について簡単に説明する．

（1）BigIntAdd(aString, bString) As String：　引き数として与えた 2 個の大きな整数の文字列の和を計算し，結果を文字列として返す．

（2）BigIntAddLong(aString, bLong) As String：　第 1 引き数として与えた大きな整数の文字列と第 2 引き数として与えた Long 型整数値の和の文字列を返す．

（3）BigIntSub(aString, bString) As String：　引き数として与えた 2 個の大きな整数の文字列より aString – bString となる数の文字列を返す．

（4）BigIntSubLong(aString, bLong) As String：　第 1 引き数として与えた大きな整数の文字列と第 2 引き数として与えた Long 型整数値より aString – bLong となる数の文字列を返す．

3.4　乗法に関連する関数

乗法に関連する関数を簡単に説明する

（1）BigIntMul(aString, bString) As String：　引き数として与えた 2 個の大きな整数の文字列の積を計算し，結果を文字列として返す．

（2）BigIntMulBy10(inString) As String：　引き数として与えた大きな整数の文字列を 10 倍した数の文字列を返す．

（3）BigIntMulByLong(aString, bLong) As String：　第 1 引き数として与えた大きな整数の文字列と第 2 引き数として与えた Long 型整数値の積の文字列を返す．

（4）BigIntSquare(aString) As String：　引き数として与えた大きな整数の文字列を 2 乗した

数の文字列を返す.

（５）BigIntPow(aString, bLong) As String：　第 1 引き数として与えた大きな整数の文字列を第 2 引き数として与えた Long 型整数値で冪乗した数 aString^bLong の文字列を返す．bLong は非負でなければならない．今回のプログラムは有理数を取り扱わないからである．冪乗はきわめて大きくなることがある．a を m 桁，b を n 桁とすると，a^b は最大 m×n 桁になる．そこで，BigIntPow( ) では，冪の値を Long 型整数値にとどめている．数論等において冪乗はあまり使われず，むしろ，後で説明する「モジュロ p の冪乗」が多用されるからでもある.

（６）BigIntFactorial(aLong) As String：　引き数として与えた Long 型整数値の階乗の文字列を返す．aLong は正でなければならない．冪乗と同じように，階乗はきわめて大きくなることがある．そのため，引き数としては Long 型整数値にとどめている.

（７）BigIntLshift(Sting, nLong) As String：　第 1 引き数として与えた大きな整数の文字列を第 2 引き数として与えた Long 型整数値 nLong だけ左へシフトした数の文字列を返す．左へ n シフトすることは 10^n 倍することと同じである.

3.5　除法に関連する関数

除法に関連する関数を簡単に説明する.

（１）BigIntDiv(aString, bString) As String：　引き数として与えた 2 個の大きな整数の文字列の商aString/bString を計算し結果を文字列として返す.

（２）BigIntDivBy2(aString) As String：　引き数として与えた大きな整数の文字列を 2 で割った商の文字列を返す.

（３）BigIntDivBy10(aString) As String：　引き数として与えた大きな整数の文字列を 10 で割った商の文字列を返す.

（４）BigIntDivByLong(aString, bLong) As String：　第 1 引き数として与えた大きな整数の文字列を第 2 引き数として与えた Long 型整数値 bLong で割った商の文字列を返す.

（５）BigIntMod(aString, bString) As String：
引き数として与えた 2 個の大きな整数の文字列のモジュロaString mod bString の文字列を返す．整数 a, b に対して，a = b×q + r，0 ≦ r < |b| が一意に成り立つ．このとき，r = a mod b と書いてモジュロと呼ぶ．a が正であればモジュロは除法における余りと一致するが，a が負の場合両者は異なる．-21 = 13×(-1) + (-8) は余りであり，-21 = 13×(-2) + 5 はモジュロである．数論においては，ほとんどのケースでモジュロを使い，余りを使うことは稀である.

（６）BigIntModByLong(String, Long) As String：
第 1 引き数として与えた大きな整数の文字列と第 2 引き数として与えた Long 型整数値とのモジュロの文字列を返す.

（７）BigIntModN(String) As String：　引き数として与えた大きな整数の文字列のモジュロ N の文字列を返す．N は，2, 3, 4, 5, 6, 7, 8, 9 という文字である．数論的な関数や計算において，これら小さな値によるモジュロがしばしばもとめられる．そのために，特別に効率的なこれらの関数が用意されている.

（８）BigIntRemainder(aString, bString) As String：　引き数として与えた 2 個の大きな整数の文字列の除法における余りの文字列を返す.

（９）BigIntDivAndRemainder(aString, bString) As String：　引き数として与えた 2 個の大きな整数の文字列の除法における商と余りをコロンで連結した文字列を返す.

(10) BigIntRoot(aString, bLong) As String：　第 1 引き数として与えた大きな整数の文字列の，第 2 引き数として与えた Long 型整数値の冪乗根の数と繰り返し計算の回数をコロンで結合した文字列を返す．aString, bLong とも非負でなければならない．正整数 a の n 乗根とは，r^n ≦ r < (r + 1)^n を満足する整数 r である.

(11) BigIntRshift(String, Long) As String：　第 1 引き数として与えた大きな整数の文字列を第 2 引き数として与えた Long 型整数値 nLong だけ右へシフトした数の文字列を返す．右へ n シフトすることは 10^n で割ることと同じであ

る．

４．数論的関数およびその他の関数

今回 Microsoft Excel上に大きな整数を実現した目的は，主として数論および暗号学について初学者が学習するための基盤整備であった．そのために，数論や暗号学のいたるところで出会うことになるいくつかのアルゴリズムをまとめて関数として提供している．ここで，それらについて簡単に説明する．なお，これらの関数は，ワークシート上でも使えるユーザー関数を多く使っている．ソースコードを参照されたい．

4.1　最大公約数と最小公倍数

2 個の整数の最大公約数 GCD を計算するための最も高速なアルゴリズムがユークリッドの互除法である．これは初等整数論から代数的整数論，素数性判定問題，素因数分解問題，公開鍵暗号系など，いたるところで必要である．そのアルゴリズムを示す[6)]．

ユークリッドの互除法による最大公約数の計算アルゴリズム　a, b を正の整数とする．

(1) r ≠ 0 であるあいだ，次を実行する．r = 0 なら (2) へ進む．
　(1-1) a を b で割った余りを r とする．a = b, b = r とする．
(2) このときの b が，GCD(a, b) なので出力して終了する．

BigIntGcd(aString, bString) As String：　引き数として与えた 2 個の大きな整数の文字列の最大公約数の文字列を返す．

a, b を正の整数とし，その最大公約数を d とすると，a, b の最小公倍数LCM(a, b) = a×b/d である．
BigIntLcm(aString, bString) As String：　引き数として与えた 2 個の大きな整数の文字列の最小公倍数の文字列を返す．

4.2　モジュロ p の乗法逆元

a を整数とし，p を素数とするとき，ax ≡ 1 (mod p) を満たす整数 x が存在する．これを $x \equiv a^{-1} \pmod p$ と書いて，a に対するモジュロ p の乗法逆元と呼ぶ．数論のいたるところで出会う問題であるところが最大公約数を求める問題に似ている[6)]．

この問題を最も効率的に解く方法も，最大公約数の場合と同じで，ユークリッドの互除法である．

なお，p が素数でなくても，a と p が互いに素（a と p の最大公約数が 1）であれば，ax ≡ 1 (mod p) を満たす整数 x は存在する．a と p が互いに素でなければ，そのような x は存在しない．

ユークリッドの互除法によるモジュロ p の乗法逆元の計算アルゴリズム

(0) a, p の入力．念のため，a = a mod p としておく．
(1) r1 = a, r2 = p, x1 = 1, x2 = 0 とする．
(2) r2 ≠ 0 であるあいだ，次を実行する．r2 = 0 なら (3) へ進む．
　(2-1) r1 = q×r2 + r3, 0 ≦ r3 < r2 なる q, r3 を計算する．
　(2-2) x3 = x1 − qx2 を計算する．
　(2-3) r1 = r2, r2 = r3, x1 = x2, x2 = x3 とする．
(3) x1 を出力して終了する．

BigIntModInv(aString, pString) As String：　引き数として与えた 2 個の大きな整数の文字列からモジュロ pString の乗法逆元を計算し，その数の文字列を返す．pString は正整数とする．

4.3　モジュロ p の冪乗

a, m, p を正整数とするとき，n ≡ a^m (mod p) を満たす整数 n を計算する問題である．a×a を計算し p によるモジュロを計算する，これを m −1 回繰り返す，という素朴な算法では，m が数 10 桁以上にも及ぶことが普通におこる数論の問題には対処できない．

冪指数の 2 進展開を使うと高速なアルゴリズムが構成できる．例えば，a^13 を計算したいとする．a^13 = a^(1 + 4 + 8) = a×a^4×a^8 と分解する．a → a×a = a^2 → a^2×a^2 = a^4 → a^4×a^4 = a^8 の計算を同時に進めれば，3 回の乗算と 3 回の 2 乗で a^13 が計算できる．このようにして冪の 2 進展開を利用して冪乗を計算するアルゴリズムを繰り返し 2 乗法と呼ぶ．なお，乗算と 2 乗を計算するときにモジュロ p を計算すれば，途中で現れる最大の数が a^2 でおさえられる．そのため，提供している関数では，a, m, p のいずれも大きな整数が使える．

繰り返し 2 乗法による n ≡ a^m (mod p) の冪乗計算アルゴリズム

(0) n = 1, a = a mod p と初期化する．
(1) m > 0 であるあいだ，以下を実行する．m = 0 なら (2) へ進む．
　(1-1) m が奇数なら，n = n×a mod p を計算する．
　(1-2) m = m/2, a = a×a mod p を計算する．
(2) n を出力して終了する．

BigIntPowMod(aString, mString, pString) As String：　引き数として与えた 3 個の大きな整数の文字列から aString^mString (mod pString) を計算し，その数の文字列を返す．aString, pString は正整数とする．単純な冪乗関数 BigIntPow( ) と異なり，冪指数mString の値として大きな整数が使える．指数が負の場合は，乗法逆元の冪乗を返す．

4.4　ヤコビ記号

整数 a，素数 p に対するヤコビ記号 (a/p) は，a が (mod p) の平方剰余であるか，平方非剰余であるか調べるために用いる関数である．

整数 a が (mod p) の平方剰余であるとは，s^2 ≡ a (mod p) を満たす整数 s が存在することである．そのような s が存在しないとき平方非剰余と言い，a が p の倍数であるとき平方剰余であるとも平方非剰余であるとも言わない．

ヤコビ記号 (a/p) は a が平方剰余であるとき 1，平方非剰余であるとき –1，a が p の倍数であるとき 0 となる関数として定義される．

ヤコビ記号 (a/p) はオイラーの基準と呼ばれる定理に基づいて，

(a/p) ≡ a^((p-1)/2) (mod p) によって計算できる．これは，素数性判定の際に使われる関係式であり，p の値は極めて大きくなることがある．例えば，a = 5, p = 2^89 – 1 のとき，p = 618970019642690137449562111 であり，5^309485009821345068724781055 (mod 618970019642690137449562111) を計算しなければならない．すでに説明したモジュロ p の冪乗計算法を使えばごく短時間で計算できる大きさではあるが，ヤコビ記号は多数回使われることがあり，より高速なアルゴリズムがのぞまれる．以下に紹介するのが，その高速アルゴリズムである[7)]．

ヤコビ記号 (a/m) の計算アルゴリズム

(0) a, m の入力．念のため, a = a mod m としておく．
(1) t = 1 と初期化する．
(2) a < 0 なら，a = -a とし，m ≡ 3 (mod 4) なら t = -t とする．
(3) a > 0 であるあいだ，以下を実行する．a = 0 なら (4) へ行く．
　(3-1) a = 2^e×q，q は奇数，となる e, q を計算する．a = q とする．
　(3-2) e が奇数で，m ≡ 3 (mod 8) または m ≡ 5 (mod 8) なら t = -t とする．
　(3-3) a ≡ m ≡ 3 (mod 4) なら t = -t とする．
　(3-4) a と m を交換する．新たな a, m について，a を a/m の余りで置き換える．
(4) m = 1 なら t を出力して終了する．m ≡ 1 なら 0 を出力して終了する．

BigIntJacobi(aString, mString) As Long：　引き数として与えた 2 個の大きな整数の文字列からヤコビ記号を計算し，Long 型整数の値を返す．ヤコビ記号は，その値を利用するよりも，何らかの条件判断に使われるのが普通だからである．

4.5　２次合同式

2 次合同式とは，x^2 ≡ a (mod p) という形の「2 次方程式」である．a が平方剰余であればこの合同関係式を満たす x が存在する．2 次合同式は，楕円曲線を使った素数性判定問題と素因数分解問題，そして，楕円曲線暗号等の有限体上の楕円曲線を使う問題で重要な役割をはたしている．以下に解法のアルゴリズムを示す[7)]．

2 次合同式 x^2 ≡ a (mod p) の解法アルゴリズム

(0) a, p の入力．念のため，a = a mod p としておく．
(1) (mod p) の平方非剰余 g を選ぶ．g = 2, 3, ... としてヤコビ記号を計算する．
(2) p – 1 = 2^r×q，ここで q は奇数，となる r, q を計算する．
(3) h ≡ g^q (mod p), b ≡ a^q (mod p) を計算する．
(4) s = 1，h ≡ h^(-1) (mod p) と初期化する．
(5) i = r – 2 から 0 まで以下を繰り返す．

(5-1) b^(2^i) ≡ -1 (mod p) なら次を実行する．
(5-1-1) s ≡ s×h (mod p), b ≡ b×h^2 (mod p) を計算する．
(5-2) h ≡ h^2 (mod p) を計算する．
(6) x ≡ a^((q+1)/2) (mod p) を出力して終了する．

BigIntModSqrt(aString, pString) As String：　引き数として与えた 2 個の大きな整数の文字列に対する 2 次合同式の解を数の文字列として返す．

4.6　大きな乱数

指定した桁数の乱数を作る問題である．このような乱数は計算によって生成するため，算術乱数，あるいは，擬似乱数と呼ばれる．擬似乱数は，線型合同法 x(k) ≡ a×x(k-1) + b (mod p) によって生成されることが多い．パラメータ a, b, p と生成される擬似乱数列との関係については Knuth による詳しい研究がある[8]．

ここでは，VBA に用意されている関数 Rnd( ) を利用する．Rnd( ) を 1 回呼ぶたびに，1 桁ずつ追加して n 桁の乱数を作ればよい．

BigIntRandom(nLong) As String：　引き数として与えた Long 型整数 nLong を桁数とする乱数の文字列を返す．

5．使用法

提供しているプログラムを Microsoft Excel で利用するためには，(1)プログラムファイルを読み込み，その上で，(2)ワークシートに若干の準備が必要である．それをまず説明し，次に，ワークシート上で提供している関数を使う方法の例と，Excel VBA プログラムを書く例を示す．関数の使い方の詳細は，プログラムのソースコードが最適である．特に，４．で説明した数論的関数とその他の関数はワークシート上で使える関数を大量に使っているので，それらを参照されたい．

5.1　準備

ここで提供しているプログラムを使うための準備作業を簡単に説明する．

Microsoft Excel を立ち上げ，メニューから「ツール」→「マクロ」→「Visual Basic Editor」を選択する．

Visual Basic Editor が開いたら，メニューから「ファイル」→「ファイルのインポート」を選択する．ファイル選択パネルが開くので，「BigIntBasic.bas」を選択する．

Visual Basic Editor のプロジェクトウィンドウに表示された「標準モジュール」をダブルクリックして開く．「標準モジュール」の下にある「Module1」をダブルクリックして開くと，Visual Basic Editor の編集画面にファイルの内容が表示される．Visual Basic Editor を閉じるか，あるいは，最小化する．

ワークシートの適当なセル（C列か D列が望ましい）に次のように入力する．

=BigIntFactorial(50)

65 桁の数である

30414093201713378043612608166064768844377641568960512000000000000が表示されれば，準備作業の第 1 段階は終了である．なお，もっと大きな数，例えば 100 の階乗なども一瞬で計算され表示されるので試すとよいであろう．

入力を容易にするようにワークシートを準備するのが第2段階の準備作業である．例えば，A列と B列を大きな整数の入力専用にする．そのためには，A列と B列を選び，セルの書式指定機能によって「表示」を「文字列」とする．さらに，見やすくするために列の幅を広げておくとよい．C列以下は数式を入力して計算結果を表示するために標準のままとする．以上で準備が終了する．

読み込んだプログラムファイルは，Microsoft Excel ブックを保存すれば同時に保存される．独自に保存する必要はない．

5.2　使用例

5.2.1　ワークシート上の例

ワークシートのセルに数式を入力し，ある程度まとまった機能を実現する例を紹介する．

最大公約数の計算

A1 と B1 に目的の 10 進数を入力する．C列，D列，E列にユークリッドの互除法の途中経過が表示されるようにする．

(1) A1に例えば 5 0603 6145，B1に例えば 9615 8981 を入力する．セルに入力する際は 4 桁ごとのスペースは空けない．
(2) C1に数式 “=A1”，D1に数式 “=B1”，E1に数式

“=BigIntMod(C1, D1)” を入力する．

(3) C2に数式 “= D1”，D2に数式 “= E1”，E2に数式 “=BigIntMod(C2, D2)” を入力する．

(4) C2から E2まで連続してセルを選択し，適当な行までオートフィルする．

(5) E列の値が0になったとき，その1つ上のセルにある値41が GCD(A1, B1) である．

計算例を示す：

506036145	96158981	506036145	96158981	25241240
		96158981	25241240	20435261
		25241240	20435261	4805979
		20435261	4805979	1211345
		4805979	1211345	1171944
		1211345	1171944	39401
		1171944	39401	29315
		39401	29315	10086
		29315	10086	9143
		10086	9143	943
		9143	943	656
		943	656	287
		656	287	82
		287	82	41
		82	41	0
		41	0	BigIntMod:DivByZero

最下行では，ゼロによる割り算が起こっているため，エラーメッセージが表示されている．

5.2.2　Excel VBA プログラムの例

Excel VBA プログラムの例として，ニュートン法を使って n 乗根を計算するためのプログラムを示す．

```
Function BigIntRootC(astr As String, n As Long) As String
    Dim a As String
    a = StrClean(Trim(astr))
    If a = "0" Or a = "1" Then
        BigIntRootC = a
    Elseif n = 1 Then
        BigIntRoot = "1"
    Else
        Dim x1 As String, x2 As String, bin As String
        Dim bunbo As String, bunsi As String
        Dim tmp1 As String, tmp2 As String
        Dim digits As Long, Length As Long
        Dim i As Long, comp As Long
        x1 = BigIntRootInit(a, n)
        For i = 1 To 100
            tmp1 = BigIntPow(x1, n - 1)
            bunbo = BigIntMulByLong(tmp1, n)
            tmp2 = BigIntMul(tmp1, x1)
            tmp1 = BigIntMulByLong(tmp2, (n - 1))
            bunsi = BigIntAdd(tmp1, a)
            x2 = BigIntDiv(bunsi, bunbo)
            comp = BigIntCompare(x2, x1)
            If comp >= 0 Then
                Exit For
            End If
            x1 = x2
        Next i
        BigIntRootC = Trim(x1)
    End If
End Function
```

Visual Basic Editor を開き，インポートしたプログラムの末尾に以上のプログラムを入力してワークシートに戻る．

“Else” とある行以下が4.3で紹介したアルゴリズムを素直にうつしたものであることが示されている．

適切なセルに下のように入力する．

=BigIntRootC("11790184577738583171520872861412518665678211592275841109096961",21)

即座に809が表示される．

次のセルに

=BigIntPow("809",21) と入力すると，

116659189245465077370143086363233260309815345023804448378160 09と表示され，その次のセルに

=BigIntPow("810",21) と入力すると，

119725151825620197886027400267170471056810000000000000000000000と表示される．これで結果の正しさが確認できる．

6．おわりに

Microsoft Excel 上で大きな整数を操作する方法を開発した．大きな整数を扱える高度なシステムがいくつも

提供されている中，あえて蛇足とも見えるシステムを，Microsoft Excel 上に新たに開発したのは，はじめにも述べたとおり，初学者の教育と学習に資さんとするためである．

これを利用すれば，「博士の愛した数式」[9] に登場する完全数や友愛数，あるいは，メルセンヌ素数等も，小さな桁数の初歩的な例にとどまらず，実際の研究場面で取り扱われるような数として調べることができる．また，公開鍵暗号系の学習に際しても，実際に使われるレベルの鍵と暗文を試すことができる．

今回提供したプログラムには含まれていないが，さまざまな素数性判定システム，素因数分解システム，RSA 暗号系，エルガマル暗号系，楕円曲線暗号系等の開発も可能である．

最近開発が進んでいる各種の基本計算アルゴリズム[10] は実装していないので，システムの完成度を高めるために，今後開発したいと考えている．

参考文献

（報告の性質上，個々の原著論文は示さず，まとまった著作物をとりあげている．また，訳書がある場合は，それを優先して示した．）

1）鈴木治郎：Excel で楽しむ数論，ピアソン・エデュケーション（2002）

2）木田祐司・牧野潔：UBASIC によるコンピュータ整数論，日本評論社（1994）

3）GAP http://www.gap-system.org

4）Koblitz, N.，櫻井幸一訳：数論アルゴリズムと楕円曲線暗号入門，シュプリンガー・フェアラーク東京（1997）Koblitz,N.: A course in number theory, Springer-Verlag,（1994）

5）和田秀雄：コンピュータと素因子分解（改定版），遊星社発行，星雲社発売（1999）

6）木田祐司：初等整数論，朝倉書店（2001）

7）Bressoud, D.M.，玉井浩訳：素因数分解と素数判定，エスアイビー・アクセス発行，星雲社発売（2004）Bressoud, D.M.: Factorization and primality testing, Springer-Verlag,（1989）

8）Knuth, D.E.，渋谷政昭訳：３．準数値算法/乱数，サイエンス社（1981）Knuth,D.: Seminumerical Algorithms（Second edition）, Addison-Wesley,（1981）

9）小川洋子：博士の愛した数式，新潮社（2003）

10）Crandall,R. and Pomerance,C.: PRIME NUMBERS: A Computational perspective, Springer-Verlag（2001）