研究ノート

小さな整数に対するモジュロ演算の高速計算アルゴリズム

野　呂　春　文

日本福祉大学　情報社会科学部

Fast algorithms of modulo operation for very small integers

Harufumi Noro

Faculty of Social and Information Sciences, Nihon Fukushi University

Keywords: Big Integer, modulo, Fast algorithm, Microsoft Excel, Visual Basic

1．はじめに

数論や暗号学の計算においてはモジュロ演算 a mod b が頻繁に使われるため高速かつ効率的なアルゴリズムが常に求められ続けている[1) 2)]．一般的な a, b の組み合わせ以外にも，特殊な形の a あるいは b について特化したアルゴリズムも重要である．それらの中で，b が 1 桁の整数であるという，一見するとトリビアルな場合もヤコビ記号の計算や２次合同式の解法等，各種の数論的アルゴリズムにおいて重要な役割をはたしている．

この報告では数論の常識にしたがい，r = a mod b は，除算アルゴリズムを満たすものとする．除算アルゴリズムとは，任意の整数 a, b に対して $a = q \times b + r$, $0 \leqq r < |b|$ を満たす整数 q, r が一意に存在することを言う．したがって, a < 0 の場合, 除算の余りを R < 0 とすると，r = a mod b = R + |b| となる．

今回，高速計算アルゴリズムが自明な b = 2, 3, 5, 9 以外の b について高速アルゴリズムを開発したのでここで報告する．なお，これらは本論集第９巻で別報している Microsoft Excel の上に桁数制限の無い大きな整数とその演算関数を開発するために使われている．そのプログラムでは，通常の計算とは異なり，２進数ではなく 10000 進数を使っている．そのため，ここで報告する内容も 10 進数を強く意識したものになっている．

2．アルゴリズム

以下でアルゴリズムを説明するとき，抽象的な説明ではわかりにくいので例を使っている．そこで使う整数は 10 進数で A = abcdef という形であるとする．つまり，$A = a \times 100000 + b \times 10000 + c \times 1000 + d \times 100 + e \times 10 + f$, $0 \leqq a,b,c,d,e,f \leqq 9$ という値を持つ整数である．また，以下では 0 以上の正整数の演算のみを考える．A が負な場合は，その絶対値 |A| を用いて計算し，後で述べる処理をおこなう．

2.1　自明な場合のアルゴリズムの再整理

A mod b の計算において，b = 2, 3, 5, 9 については

簡単に高速アルゴリズムを構成することができる．

b = 2 とする．最下位の 1 桁について mod 2 を計算すればよい．実際は, 0 から 9 までの数の 2 によるモジュロを計算するのであるから表引きですむ．

b = 3 とする．
A = a × 100000 + b × 10000 + c × 1000 + d × 100 + e × 10 + f
= 3 × (a × 33333 + b × 3333 + c × 333 + d × 33 + e × 3) + (a + b + c + d + e + f)
であるから，高速計算法として A mod 3 = (a + b + c + d + e + f) mod 3，つまり全桁の数字を足して mod 3 を計算するという，小学校でも学ぶアルゴリズムが得られる．

b = 5 とする．最下位の 1 桁について mod 5 を計算すればよい．当然，表引きだけですむ．

b = 9 とする．
A = a × 100000 + b × 10000 + c × 1000 + d × 100 + e × 10 + f
= 9 × (a × 11111 + b × 1111 + c × 111 + d × 11 + e × 1)) + (a + b + c + d + e + f)
であるから，高速計算法として A mod 9 = (a + b + c + d + e + f) mod 9，つまり全桁の数字を足して mod 9 を計算するというアルゴリズムが得られる．

2.2　b = 4, 6, 8 に対する高速計算アルゴリズム

b = 4 とする．
A = a × 100000 + b × 10000 + c × 1000 + d × 100 + e × 10 + f
= 4 × (a × 25000 + b × 2500 + c × 250 + d × 25) + e × 10 + f
であるから，高速計算法として A mod 4 = (e × 10 + f) mod 4，つまり，最下位の 2 桁についてだけ mod 4 を計算するというアルゴリズムが得られる．0 以上 99 以下の値であるから表引きでも対応できる．

b = 6 とする．
A = a × 100000 + b × 10000 + c × 1000 + d × 100 + e × 10 + f
= 6 × (a × 16666 + b × 1666 + c × 166 + d × 16 + e × 1) + 4 × (a + b + c + d + e) + f
である．これより，A mod 6 = {4 × (a + b + c + d + e) + f} mod 6 というアルゴリズムが得られる．

b = 8 とする．
A = a × 100000 + b × 10000 + c × 1000 + d × 100 + e × 10 + f
= 8 × (a × 12500 + b × 1250 + c × 125) + d × 100 + e × 10 + f
である．これより，A mod 8 = (d × 100 + e × 10 + f) mod 8，つまり，最下位の 3 桁についてだけ mod 8 を計算するというアルゴリズムが得られる．

2.3　b = 7 に対する高速計算アルゴリズム

b = 7 については，上記のようなアルゴリズムの構成が難しい．そこで，10 mod 7 = 3 という関係の利用を考える．この関係を繰り返し用いると，
100 mod 7 = 3 × 3 mod 7，1000 mod 7 = 3 × 3 × 3 mod 7, . . . となる．つまり，
A = (a × 100000 + b × 10000 + c × 1000 + d × 100 + e × 10 + f) mod 7
= (a × 3 × 3 × 3 × 3 × 3 + b × 3 × 3 × 3 × 3 + c × 3 × 3 × 3 + d × 3 × 3 + e × 3 + f) mod 7
と展開できる．これをもとに次のようなアルゴリズムを構成する．ここで，k 桁の整数 a の最上位の 1 桁の値を a(k) とし，順次，a(k-1), a(k-2), ..., a(1) とする．

a mod 7 の高速計算アルゴリズム
(0) t = 0 とする．
(1) i = k から 1 まで次を繰り返す．
t = (t × 3 + a(i)) mod 7 を計算する．
以上で，t = a mod 7 が得られる．

ここで各ステップにおいて現れる値に注意すると，t ≦ 6，a(i) ≦ 9 であるから，t × 3 + a(i) ≦ 27 である．したがって (t × 3 + a(i)) mod 7 の計算は，(t × 3 + a(i)) の値による表引きに置き換えることができる．

2.4　A が負の場合の処理

A が負の場合は特別な処理が必要になる．例で見る．
A = -16, b = 7 とする．上記のアルゴリズムで r = |A|

mod b = 2 が得られる．一方，除算アルゴリズムによれば, -16 = (-3)×7 + 5であり, 食い違いがある. したがって，負の A に対して上記のアルゴリズムで得られた値を r とするとき，A mod b = b - r が正しい結果となる．

3．計算速度について

計算速度を計算時間で表わすことは，CPU，プログラミング言語等の各種の条件が関係するため困難である．そこで，一般的には計算の手間で考える習慣があるので，それにしたがう．

a mod b において a の桁数を l0 進 n 桁とする．b は 1 桁である．通常のモジュロ演算の手間は割り算と同じだから，a の 1 桁あたり 1 回の割り算と 1 回の掛け算が必要である．結果として 2n 回程度の掛け算が必要である．

b = 2, 4, 5 の場合，表引きを使えば掛け算の回数は 0 である．b = 3, 6, 9 については掛け算の回数が log10(n) 程度になる．b = 8 では掛け算は 6 回である．

問題の b = 7 の場合は，掛け算の回数は n 回となり，手間数の改善は l/2 にすぎず，他の b のように劇的ではない．しかし，数論や暗号の計算では a が数百桁におよぶことが普通であり，この程度でも大きな改善になっている．

4．まとめ

Microsoft Excel の上に桁数制限のない大きな整数とその演算関数を実現する過程で上記のアルゴリズムを得た．その際，プログラムを開発するために Microsoft Excel に付属している Visual Basic を利用する以外の方法がなく，ビットシフト等のビット処理の実現が困難であったため，２進表現の利点を生かすことをあきらめざるをえなかった．

文献

1）Knuth, D.E.：準数値算法　第４分冊，サイエンス社，(1986)
2）Crandall,R. and Pomerance,C.：Prime Numbers：A Computational Perspective,Springer-Verlag,(2001)