アルゴリズム B 第 9 回　　2022 年 12 月 5 日

9　自然数の四則演算・高速冪乗計算

位取り記数法

b を 2 以上の整数とする．自然数 n が

$$n = \sum_{i=0}^{m-1} a_i b^i \quad (m \text{ は自然数}, a_i \in \{0, 1, \ldots, b-1\}, a_{m-1} \neq 0)$$

を満たすとき，$n = (a_{m-1} \cdots a_1 a_0)_b$ と表す．この記数法を **b 進法** (b-ary system, base-b system) といい，b を**底** (base) または**基数** (radix) という．通常用いられるのは **10 進法** (decimal system) であり，計算機を扱う際は **2 進法** (binary system)，**8 進法** (octal system)，**16 進法** (hexadecimal system) もよく用いられる．16 進法では，10, 11, . . . , 15 を A, B, . . . , F または a, b, . . . , f で表すことが多い．

自然数の四則演算

自然数の四則演算の計算量について考える．計算量を測るには様々なモデルが用いられ，精密な評価の際には多テープチューリング機械などが用いられるが，ここではより単純なビット演算量を用いる．

■加算　2 進法で表された 2 つの自然数の加算を考える．k ビット（2 進法で k 桁）の自然数の加算は，次の操作を k 回繰り返すことで行われる．

- 足されるビット x，足すビット y，下の桁からの繰上ビット z に対して，

$$x + y + z = 2c + r$$

によって，足した結果ビット r，上の桁への繰上ビット c を定める．（$x, y, z, r, c \in \{0, 1\}$ とする．）

このビット演算を単位として求めた計算量を**ビット演算量** (bit complexity) という．したがって，高々 k ビットの 2 つの自然数の加算のビット演算量は高々 k である．

■減算　高々 k ビットの 2 つの自然数の減算は，高々 k のビット演算量でできる．ただし，ビット演算は次の操作とする．

- 引かれるビット x，引くビット y，下の桁からの繰下ビット z に対して，

$$2c + x - y - z = r$$

によって，引いた結果ビット r，上の桁への繰下ビット c を定める．（$x, y, z, r, c \in \{0, 1\}$ とする．）

■乗算　k ビットの自然数と l ビットの自然数の乗算は，筆算と同様に行うと，高々 kl のビット演算量でできる．ここで，桁をずらす操作の計算量は小さいので無視している．

■除算　k ビットの自然数を l ビットの自然数で割って商と剰余を求める計算は，筆算と同様に行うと，高々 kl のビット演算量でできる．

以上をまとめると，高々 n ビットの 2 つの自然数について，加減算は $O(n)$ のビット演算量で，乗除算は $O(n^2)$ のビット演算量で実行できる．また，0 以下の整数も含めた整数の四則演算についても，符号を適切に扱うことで同じ計算量となる．

カラツバ法

筆算より高速な自然数の乗算アルゴリズムに**カラツバ法** (Karatsuba method) がある．

x, y を高々 n ビットの自然数とする．$m = \lceil n/2 \rceil$ として，

$$x = a2^m + b, \quad y = c2^m + d, \quad 0 \le a, b, c, d < 2^m$$

と表す．このとき，

$$xy = ac2^{2m} + ((a+b)(c+d) - ac - bd)2^m + bd$$

となるから，xy を計算するには 3 回の乗算 ac, bd, $(a+b)(c+d)$，桁をずらす操作，加減算を行えばよい．これを再帰的に繰り返すことで自然数の乗算を行うことができる．全体のビット演算量は $O(n^{\log_2 3})$ になることが証明できる．$\log_2 3 = 1.58\cdots$ だから，これは筆算より漸近的に高速である．

注意. さらに高速な乗算アルゴリズムも知られている．以下では，多テープチューリング機械によって計算量を評価し，高々 n ビットの 2 つの自然数の乗算を考える．1971 年，Schönhage, Strassen は，高速フーリエ変換 (FFT) を用いることで，計算量 $O(n \log n \log\log n)$ の乗算アルゴリズムを構成した．2007 年，Fürer はこの上界を改善することに初めて成功し，計算量を $n \log n\, 2^{O(\lg^* n)}$ とした[*1]．2019 年には Harvey, van der Hoeven が計算量 $O(n \log n)$ の乗算アルゴリズムを構成した[*2]．

また，除算についても，ニュートン法を用いることで乗算の定数倍の計算量で計算できる．

高速冪乗計算

x を乗法の定義された代数系（正確には半群）の要素，n を自然数として，x^n の計算を考える．素朴な方法は，順に $x^2, x^3, \ldots, x^n$ と，x による乗算を $n-1$ 回繰り返す方法である．乗算回数がより少ない方法を以下で述べる．n を 2 進法で $n = (n_k \cdots n_1 n_0)_2$ と表す．ただし，$n_k = 1$ とする．

$$x_0 = x, \quad x_i = {x_{i-1}}^2 = x^{2^i} \quad (i = 1, 2, \ldots, k)$$

と定める．この計算は k 回の乗算でできる．このとき，

$$x^n = \prod_{\substack{n_i = 1 \\ 0 \le i \le k}} x_i$$

によって x^n が計算できる．

$$\nu(n) = \#\{i \mid n_i = 1, 0 \le i \le k\}$$

とおくと，x^n の計算に必要な乗算の回数は，$k + \nu(n) - 1$ である．したがって，x^n の計算は $O(\log n)$ の乗算で実行できる．この方法を**繰り返し 2 乗法** (repeated squaring) という．（繰り返し 2 乗法には，乗算の順序がこれと異なるものもある．）

[*1] 2 を底とする対数を lg で表し，$\lg^{(0)} n = n$, $\lg^{(i+1)} n = \lg \lg^{(i)} n$ としたとき，$\lg^* n = \min\{i \ge 0 \mid \lg^{(i)} n \le 1\}$ と定義される．

[*2] Fürer の論文は 2009 年に，Harvey, van der Hoeven の論文は 2021 年にそれぞれ出版された．