アルゴリズム論特論　(塩田)

2009 年 5 月 14 日のレジュメ

今回紹介する系 4 とアルゴリズム 5 は、暗号理論、整数論、計算代数において極めて重要な役割を果たす。

$\boxed{1}$　$ax + by$ の形で表せる数

定理 1　$a, b \in \mathbf{Z}$ を与えたとき、$ax + by$ $(x, y \in \mathbf{Z})$ の形で表すことのできる整数全体の集合

$$G = \{\, ax + by \,|\, x, y \in \mathbf{Z} \,\}$$

は、$d = \gcd(a, b)$ の倍数全体の集合 $d\mathbf{Z} = \{\, dz \,|\, z \in \mathbf{Z} \,\}$ に一致する。

系 2　$a, b \in \mathbf{Z}$ を与えたとき、$ax + by = \gcd(a, b)$ を満たす $x, y \in \mathbf{Z}$ が存在する。

補題 3　$\mathbf{Z}$ の空でない部分集合 H が「差について閉じている」、つまり、条件

$$\forall f, g \in H, \quad f - g \in H$$

を満たしているとする。このとき、或る H の元 h が存在して $H = h\mathbf{Z}$ となる。

補題 3 の証明

1° $0 \in H$ であること。

$\because$　$H \neq \emptyset$ ゆえ元 $f \in H$ を取れる。条件より $0 = f - f \in H$.

2° $f \in H$ ならば $-f \in H$ であること。

$\because$　1° と条件から、$-f = 0 - f \in H$.

3° H が 0 しか元をもたないとき。

$h = 0$ として補題が成立する。$\rightarrow$ exit

4° H が 0 以外に元をもつとき。

H の正の元で最小のものを h とする。(2° より存在する。)

5° $h\mathbf{Z} \subseteq H$ であること。

$\because$　1° と条件より $-h = 0 - h \in H$. 以下、帰納的に $-2h = (-h) - h \in H$, $-3h = (-2h) - h \in H, \cdots$. すると 2° より $2h, 3h, \cdots \in H$.

6° $H \subseteq h\mathbf{Z}$ であること。

$\because$　k を H の任意の元とし、k を h で割った商と余りを q, r と置く：

$$k = qh + r \quad (0 \leqq r < h).$$

5° より $qh \in H$ であるので、条件より

$$r = k - qh \in H.$$

もし $r > 0$ ならば h の最小性に反するので $r = 0$. すなわち H の任意の元は h の倍数になり、$H \subseteq h\mathbf{Z}$. $\square$

定理 1 の証明

1° $G = \{\, ax + by \mid x,\, y \in \mathbf{Z} \,\}$ と置くと、

$$(ax + by) - (ax' + by') = a(x - x') + b(y - y')$$

ゆえ G は補題 3 の条件を満たす。

2° 従って或る G の正の元 h が存在して $G = h\mathbf{Z}$.

3° $h \geqq d$ であること。

$\because$ a, b は d の倍数ゆえ $ax + by$ の形の数も d の倍数。特に h も d の倍数ゆえ $h \geqq d$.

4° $d \geqq h$ であること。

$\because$ $a = a \times 1 + b \times 0 \in G$, $b = a \times 0 + b \times 1 \in G$ ゆえ a, b は h の倍数。言い換えれば h は a, b の公約数になる。従って $d \geqq h$. □

系 4 $a, b \in \mathbf{Z}$ が互いに素であれば、

(1) 任意の整数を $ax + by$ $(x,\, y \in \mathbf{Z})$ の形で表すことができる。

(2) $ax + by = 1$ を満たす $x,\, y \in \mathbf{Z}$ が存在する。

$\boxed{2}$ ユークリッドのアルゴリズム拡張版

$ax + by = \gcd(a, b)$ を満たす x, y はユークリッドのアルゴリズムを拡張することにより高速に求めることができる。

アルゴリズム 5 (ユークリッドのアルゴリズム拡張版)

入力：$a, b \in \mathbf{Z}$

出力：$d = \gcd(a, b)$ と、$d = ax + by$ を満たす $x, y \in \mathbf{Z}$

1° 3 つの数列 $\{\, r_n \,\}, \{\, x_n \,\}, \{\, y_n \,\}$ を用意する。

2° 初期値は $\begin{cases} r_0 = a, & x_0 = 1, \quad y_0 = 0 \\ r_1 = b, & x_1 = 0, \quad y_1 = 1 \end{cases}$

3° $n = 1$.

4° $r_n = 0$ ならば Step 6 へ。

5° $\begin{cases} n + 1 \to n \\ q = r_{n-2} \text{ を } r_{n-1} \text{ で割った商} \\ r_n = r_{n-2} - q \times r_{n-1} \\ x_n = x_{n-2} - q \times x_{n-1} \\ y_n = y_{n-2} - q \times y_{n-1} \end{cases}$

6° $\begin{cases} r_{n-1} \geqq 0 \text{ ならば} & d = r_{n-1}, \quad x = x_{n-1}, \quad y = y_{n-1} \\ \text{そうでなければ} & d = -r_{n-1}, \quad x = -x_{n-1}, \quad y = -y_{n-1} \end{cases}$

アルゴリズム 5 の証明　終了時の n で $d = \pm r_{n-1}$ となることは前回示した。

$$r_n = ax_n + by_n \qquad (n = 0, 1, \cdots)$$

が成り立つことは帰納法により示される。□

アルゴリズム 6　(ユークリッドのアルゴリズム拡張版再帰 ver.)

入出力は上に同じ。

1° $b \neq 0$ ならば 3° へ。

2° $\begin{cases} a \geqq 0 \text{ ならば} & d = a, \quad x = 1, \quad y = 0 \\ \text{そうでなければ} & d = -a, \quad x = -1, \quad y = 0 \end{cases}$

を出力して終了。

3° $\begin{cases} q = a \text{ を } b \text{ で割った商} \\ r = a - b \times q \end{cases}$

とし、再帰呼び出しで

$\begin{cases} d_0 = \gcd(b, r) \text{ と} \\ d_0 = bx_0 + ry_0 \text{を満たす } x_0,\ y_0 \end{cases}$

を求め $d = d_0$, $x = y_0$, $y = x_0 - q \times y_0$ を出力。

アルゴリズム 6 の証明　$b = 0$ のとき 2° は $d = |\,a\,| = a \times (\pm 1) + b \times 0$ ゆえ OK. $b \neq 0$ のときは帰納法の仮定により

$$d = d_0 = bx_0 + ry_0 = bx_0 + (a - bq)y_0 = ay_0 + b(x_0 - qy_0). \quad \square$$

定理 7　アルゴリズム 5、アルゴリズム 6 の計算量は $O(\log^3 a)$ である。

注 8　再帰呼び出しの深さに制限がある処理系ではアルゴリズム 6 はエラーを起こす恐れがある (ホームページの `L04.c` の `euclid3`)。アルゴリズム 5 も、そのまま組んだのでは配列が必要になりメモリを喰う (同 `euclid1`)。そこでアルゴリズム 5 を、数列を 3 項ずつ使い回しするように組めば使用するメモリも少なくて良い (同 `euclid2`)。