剰余環を用いた冪根の高速計算

ゆうき

平成 15 年 12 月 13 日

概 要

剰余環 $\mathbb{Z}/r\mathbb{Z}$ に於けるニュートン法などを用いて、perfect powers の冪根を高速に求める方法を考える。

目 次

1 平方根

今、2 の冪を法としたニュートン法を用いようとしているので、平方根は他の素数乗根とは別に考える必要がある。

Lemma 1.1. 1 以上の自然数 b について、$2i \equiv 2j \pmod{2^{b+1}}$ ならば $i^2 \equiv j^2 \pmod{2^{b+1}}$

Proof. 仮定より $2(i-j)$ が $\pmod{2^{b+1}}$ の倍数だから $i-j$ は 2^b の倍数。

特に、i, j の偶奇は一致するので、$i+j$ は 2 の倍数。

したがって、$i^2 - j^2 = (i+j)(i-j)$ は $2 \cdot 2^b = 2^{b+1}$ の倍数。 □

Algorithm 1.2 (剰余環上の平方根の逆数). `nsqrt_2exp`(y, b)

入力 $y \in \mathbb{N}_{\mathrm{odd}}, b \in \mathbb{N}$

出力 $yr^2 \equiv 1 \pmod{2^{b+1}}$ なる $r \in \mathbb{N}$。もしそのような r が存在しなければ 0 を返す。

1. $y \bmod 4 \neq 1$ ならば 0 を返して終了
2. $b = 1$ のとき、1 を返して終了
3. $b_0 := 2$, $r_0 := y \bmod 8$
4. $r_0 \neq 1$ ならば 0 を返して終了
5. $b = 2$ のとき、1 を返して終了
6. `for` $i = 1, 2, \ldots$
   $b_i := 2b_{i-1} - 1$
   $b_i \geq b$ ならば、ループを終了
   $r_i := (3r_{i-1} - yr_{i-1}^3)/2 \bmod 2^{b_i}$

7. $b_i \geq b$ となった最大の i、すなわちループを終了したときの i を、i_* とする。
8. $r := (3r_{i_*-1} - yr_{i_*-1}^3)/2 \bmod 2^b$ を返してアルゴリズムを終了

Lemma 1.3. *Algorithm 1.2* は正しく動作する。

Proof. $b = 1, 2$ の場合は自明。以下、$b > 2$ について考える。

Step 1 によって 0 が返った場合、$yr^2 \equiv 1 \pmod 4$ を満たす r は存在しない。$4 \mid 2^{b+1}$ なので、$\bmod 2^{b+1}$ においても存在しない。従って、返り値 0 は正しい。

Step 3 においても同様。以下、Step 8 における返り値を吟味する。

$$r_{i_*} := (3r_{i_*-1} - yr_{i_*-1}^3)/2 \bmod 2^{b_{i_*}}$$

と定義する。$b_{i_*} \geq b$ より、$0 \leq i \leq i_*$ の各 i について、$yr_i^2 \equiv 1 \pmod{2^{b_i+1}}$ を証明すれば十分。i についての帰納法によってこれを証明する。

$i = 0$ については、step 1 により自明。今、ある i_0 について成立したものと仮定して、$i = i_0 + 1$ について証明する。

帰納法の仮定より $yr_{i-1}^2 \equiv 1 \pmod{2^{b_{i-1}+1}}$ である。従って r_{i-1} は奇であり、r_i は well-defined.

また前式より、次のような $j \in \mathbb{N}$ が存在する。

$$yr_{i-1}^2 = 1 + 2^{b_{i-1}+1} j \tag{1}$$

ここで r_i の定義より

$$\begin{aligned} 2r_i &\equiv 3r_{i-1} - yr_{i-1}^3 \pmod{2^{b_i+1}} \\ &= r_{i-1}\left(3 - yr_{i-1}^2\right) \\ &= r_{i-1}\left(3 - 1 - 2^{b_{i-1}+1} j\right) \quad (\because (1)) \\ &= 2r_{i-1}\left(1 - 2^{b_{i-1}} j\right) \end{aligned}$$

すると、Lemma 1.1 より

$$r_i^2 \equiv r_{i-1}^2 \left(1 - 2^{b_{i-1}} j\right)^2 \pmod{2^{b_i+1}}$$

従って、

$$\begin{aligned} yr_i^2 &\equiv yr_{i-1}^2 \left(1 - 2^{b_{i-1}} j\right)^2 \pmod{2^{b_i+1}} \\ &= \left(1 + 2^{b_{i-1}+1} j\right)\left(1 - 2^{b_{i-1}+1} j + 2^{2b_{i-1}} j^2\right) \\ &\equiv \left(1 + 2^{b_{i-1}+1} j\right)\left(1 - 2^{b_{i-1}+1} j\right) \pmod{2^{b_i+1}} \qquad (\because 2b_{i-1} = b_i + 1) \\ &= \left(1 - 2^{2b_{i-1}+2} j^2\right) \\ &\equiv 1 \pmod{2^{b_i+1}} \qquad (\because 2b_{i-1} + 2 > b_i + 1) \end{aligned}$$

□

Corollary 1.4. $n \in \mathbb{N}_{\text{odd}}$, $b \in \mathbb{N}$ に対し、

$$r = n \cdot \texttt{nsqrt\_2exp}(n, b)$$

とすると、$r \neq 0 \Leftrightarrow r^2 \equiv n \pmod{2^{b+1}}$ である。

Proof. $n \bmod 2^{b+1} \neq 0$ より $\Leftarrow$ は明らか。
$\Rightarrow$ は、$s = \texttt{nsqrt\_2exp}(n, b)$ とおけば、Lemma より

$$r^2 = (ns)^2 = n(ns^2) \equiv n \pmod{2^{b+1}}$$

となって成立する。 □

Algorithm 1.5 (奇数の平方根). $\texttt{odd\_sqrt}(n)$

入力 $n \in \mathbb{N}_{\text{odd}}$

出力 $n = r^2$ なる自然数 r。もしそのような r が存在しなければ 0 を返す。

1. $2^{2b} \geq n$ なる最小の b をとる
2. $r := n \cdot \texttt{nsqrt}(n, b) \bmod 2^b$
3. $r = 0$ ならば 0 を返して終了
4. $r^2 = n$ ならば r を返して終了
5. $\left(2^b - r\right)^2 = n$ ならば $2^b - r$ を返して終了
6. 0 を返して終了

Theorem 1.6. *Algorithm 1.5* は正しく動作する。

Proof. Step 3 により 0 が返ったとき、Corollary 1.4 より $\mathbb{Z}/2^{b+1}\mathbb{Z}$ においては n の平方根は存在しない。従って、$\mathbb{Z}$ においても存在せず、返り値 0 は正しい。Step 4,5 により値が返ったとき、その値が正当であることは自明。

以下では残る step 6 においての返り値 0 が正しいことを示す。そのためには、$\mathbb{Z}$ における n の平方根 $s \in \mathbb{N}$ が存在したとき $s = r$ または $s = 2^b - r$ が成立して、step4,5 において必ずその値が返ることを言えば十分である。

Corollary 1.4 より $r^2 \equiv n = s^2 \pmod{2^{b+1}}$ なので、

$$2^{b+1} \mid (s^2 - r^2) = (s + r)(s - r)$$

今、r, s はともに奇だから mod4 において $s \equiv r$ または $s \equiv -r$ が成立する。

- $s \equiv r$ のとき、$s+r \equiv 2 \pmod 4$ だから $2||(s+r)$。従って $2^b|(s-r)$、すなわち $s \equiv r \pmod{2^b}$。
  ここで $0 \leq r, s < 2^b$ より $s = r$
- $s \equiv -r$ のとき、同様にして $s \equiv -r \pmod{2^b}$。$0 \leq r, s < 2^b$ より $s = 2^b - r$

□

2 奇数乗根

ここでは、2 の冪を法とした Newton 法により、奇数 n の奇数乗根を求めることを考える。

なお、今回の目的には $\mathbb{Z}$ の上で考えれば十分であるが、[1] の記述に合わせてまずは一般の単位的可換環における Newton 法を考えた。

Algorithm 2.1 (p 進逆数). $\texttt{inv}_p(f,\, g_0,\, k)$
R を単位的可換環とする。

入力 $k \in \mathbb{N}$, $p \in R$, および $fg_0 \equiv 1 \pmod p$ なる $f, g_0 \in R$

出力 $fg \equiv 1 \pmod{p^k}$ なる $g \in R$

1. $r := \lceil \log k \rceil$
2. `for` $i = 1, ..., r$,
   $g_i := (2g_{i-1} - fg_{i-1}^2) \bmod p^{2^i}$
3. g_r を返して終了。

Lemma 2.2. *Algorithm 2.1* は正しく動作する。

Proof. r についての帰納法により示す。$r = 0$ のときは入力の仮定より明らか。

今、ある r_0 について成立していると仮定する。以下、$r = r_0 + 1$ の場合について示す。

仮定より $p^{2^{r-1}} \mid (fg_{r-1} - 1)$。従って $p^{2^r} \mid (fg_{r-1} - 1)^2$。つまり、

$$(fg_{r-1} - 1)^2 = f^2 g_{r-1^2} - 2fg_{r-1} + 1 \equiv 0 \pmod{p^{2^r}}$$

従って、

$$fg_r = f(2g_{r-1} - fg_{r-1}^2) = 2fg_{r-1} - f^2 g_{r-1}^2 \equiv 1 \pmod{p^{2^r}}$$

□

Corollary 2.3. R を単位的可換環、$k \in \mathbb{N}$ とする。このとき以下は同値である。

- $f \in R$ が $\bmod p^k$ で可逆
- $f \in R$ が $\bmod p$ で可逆

Proof. $\Rightarrow$ は $p \mid p^k$ より明らか。

$\Leftarrow$ は、Algorithm 2.1 により $\bmod p^k$ での逆元を作ることができるので成立する。 □

Lemma 2.4 (Netwon 法の 2 次収束性). R を単位的可換環とし、$m \in R$, $\phi \in R[y]$ とする。$g \in R$ が

- $\phi(g) \equiv 0 \pmod{m}$
- $\phi'(g)$ は $\bmod m$ で可逆 *(ただし ϕ' は ϕ の形式的微分)*。

を満たすとき、次の $\bmod\ m^2$ における *Netwon* 法の漸化式によって $h \in R$ を定義する。

$$h \equiv g - \phi(g)\,\phi'(g)^{-1} \pmod{m^2} \tag{2}$$

このとき、次が成立する。

- $h \equiv g \pmod{m}$
- $\phi(h) \equiv 0 \pmod{m^2}$
- $\phi'(h)$ は $\bmod m^2$ で可逆

Note 1. この Lemma を、最終的な目的である $\mathbb{Z}$ の上での $\bmod 2^k$ 上の Newton 法に当てはめて考えると、Newton 法による近似を 1 回行うごとに近似値 p 進距離の意味で収束し、その収束の仕方は 2 次であるということを表している。

これは $\mathbb{R}$ 上の Newton 法が絶対値距離の意味で 2 次収束することに丁度対応する。

Proof. Corollary 2.3 より $\phi'(g)$ は $\mathrm{mod} m^2$ でも可逆である。従って h は well-defined.

$m \mid m^2$ より (2) は mod m においても成立し、また $\phi(g) \equiv 0 \pmod{m}$ だから

$$h \equiv g - \phi(g)\,\phi'(g)^{-1} \equiv g \pmod{m}$$

これより主張の第 1 は示された。なお、これにより $(h-g)^2 = 0 \pmod{m^2}$ が言える。

次に、$\phi(y)$ を g を中心に Taylor 展開して $y = h$ を代入すると、ある $\psi \in R[y]$ が存在して

$$\begin{aligned}
\phi(h) &= \phi(g) + \phi'(g)(h-g) + \psi(h-g)\cdot(h-g)^2 \\
&\equiv \phi(g) + \phi'(g)(h-g) \pmod{m^2} \quad (\because m^2 \mid (h-g)^2) \\
&\equiv \phi(g) + \phi'(g)\cdot(-\phi(g)\,\phi'(g)^{-1}) \pmod{m^2} \quad (\because h \text{ の定義}) \\
&\equiv 0
\end{aligned}$$

以上で主張の第 2 が示された。

さて、仮定より $\phi'(g)$ は mod m で可逆。$g \equiv h \pmod{m}$ より $\phi'(h)$ も mod m で可逆。従って Corollary 2.3 により $\phi'(h)$ は mod m^2 でも可逆。主張の第 3 も示された。 □

Algorithm 2.5 (p 進 Newton 法). $\texttt{poly\_root}_p(\phi,\, k,\, g_0,\, s_0)$

R を単位的可換環、$p \in R$ とする。

入力
- $\phi \in R[y]$
- $k \in \mathbb{N}_{>0}$
- $\phi(g_0) \equiv 0 \pmod{p}$、かつ $\phi'(g_0)$ が mod p で可逆であるような $g_0 \in R$
- $\phi'(g_0)$ の mod p における逆元 s_0

出力 $\phi(g) \equiv 0 \pmod{p^k}$ かつ $g \equiv g_0 \pmod{p}$ なる $g \in R$

1. $r := \lceil \log k \rceil$
2. `for` $i = 1, ..., r-1$,

$$\begin{aligned}
g_i &:= (g_{i-1} - \phi(g_{i-1})s_{i-1}) \bmod p^{2^i} \\
s_i &:= \left(2s_{i-1} - \phi'(g_i)s_{i-1}^2\right) \bmod p^{2^i}
\end{aligned}$$

3. $g := (g_{r-1} - \phi(g_{r-1})s_{r-1}) \bmod p^k$ を返して終了。

Lemma 2.6. *Algorithm 2.5* は正しく動作する。

Proof. $g_r := (g_{r-1} - \phi(g_{r-1})s_{r-1}) \bmod p^{2^r}$ とすると

$$g \equiv g_r \quad (\because 2^r \geq k)$$

だから、

$$\begin{aligned}
\phi(g_r) &\equiv 0 \pmod{p^{2^r}} \\
g_r &\equiv g \pmod{p} \\
s_r &\equiv \phi'(g_r)^{-1} \pmod{p^{2^r}}
\end{aligned}$$

を示せば十分。

r についての帰納法を用いる。$r = 0$ のときには、入力についての仮定から明らか。ある $i-1$ について成立しているとき、i についても成立することを示す。

帰納法の仮定より $p^{2^{i-1}}$ は $\phi(g_{i-1})$ および $s_{i-1}-\phi'(g_{i-1})^{-1}$ を割る。
従って、p^{2^i} はこれらの積 $\phi(g_{i-1})\left(s_{i-1}-\phi'(g_{i-1})^{-1}\right)$ を割る。
これより、

$$g_i \equiv g_{i-1} - \phi(g_{i-1})s_{i-1} \equiv g_{i-1} - \phi(g_{i-1})\,\phi'(g_{i-1})^{-1} \pmod{p^{2^i}}$$

今、Lemma 2.4 において $g = g_{i-1}$, $h = g_i$, $m = p^{2^{i-1}}$ とおけば、最初の 2 つの式は従う。
また、$\phi(g_{i-1}) \equiv 0 \pmod{p^{2^{i-1}}}$ より $g_i \equiv g_{i-1} \pmod{p^{2^{i-1}}}$。これより

$$\phi'(g_i)^{-1} \equiv \phi'(g_{i-1})^{-1} \equiv s_{i-1} \pmod{p^{2^{i-1}}}$$

すると、s_i の漸化式は Algorithm 2.1 において $l = 2$, $f = \phi'(g_i)$, $g_0 = s_{i-1}$, $p = p^{2^{i-1}}$ とおいたものに等しい。従ってその出力 s_i は $\bmod p^{2^i}$ での $\phi'(g_i)$ の逆数である。
以上で最後の式が示された。 □

Algorithm 2.7 (奇数の奇数乗根). `odd_root`(a, n)

入力 $a, n \in \mathbb{N}$。但し、ふたつとも奇数。

出力 $a = b^n$ なる $b \in \mathbb{N}$ が存在すれば b を返す。存在しないときには 0 を返す。

1. $f(y) := y^n - a \in \mathbb{N}[y]$ とする。
2. k を $2^{nk} > a$ なる最小の自然数とする。
3. $b :=$ `poly_root`$_2(f, k, 1, 1)$ とする。
4. $b^n = a$ ならば b を、そうでなければ 0 を返す。

Theorem 2.8. *Algorithm 2.7* は正しく動作する。

Proof. a は奇数なので $f(1) = 1^n - a \equiv 0 \pmod 2$。また、$n$ も奇数なので $f'(1) = n \cdot 1^{n-1} = 1 \pmod 2$。従って、$f(1)$ は $\bmod 2$ において可逆で逆元は 1。
以上から、$(f, k, 1, 1)$ は`poly_root`$_2(\phi,\, k,\, g_0,\, s_0)$ の入力の条件を満たす。従って、$f(b) = b^n - a \equiv 0 \pmod{2^k}$。
ここで、もし $b_0^n = a$ なる $b_0 \in \mathbb{N}$ が存在したなら、k の取り方より $b_0 < 2^k$。b_0 が存在するなら $b_0 = b$ である。
従って $b^n = a$ のとき b が求める根であり、そうでないとき根は存在しない。 □

3 一般の冪根

Algorithm 3.1. `root`(a, n)

入力 $a, n \in \mathbb{N}$。

出力 $a = b^n$ なる $b \in \mathbb{N}$ が存在すれば b を返す。存在しないときには 0 を返す。

1. $2^k || a$ なる $k \in \mathbb{N}$ を求める。
2. $n \not| k$ ならば 0 を返して終了。そうでなければ $a' := a/2^k$ とする。
3. $2^l || n$ なる $l \in \mathbb{N}$ および $n' := n/2^l$ を求める。
4. $b_0 :=$ `odd_root`(a', n')
5. b_0 が 0 なら 0 を返して終了。

6. `for` $i = 1, ..., l$
   $b_i := b_{i-1}$`odd_sqrt`(b_{i-1}) を求め、b_i が 0 なら 0 を返して終了。
7. $b := 2^{k/n} b_l$ を返して終了。

Theorem 3.2. *Algorithm 3.1* は正しく動作する。

Proof. 構成より明らか。 □

参考文献

[1] Joachim von zur Gathen and Jügen Gerhard, *Modern Computer Algebra* 2nd ed., 2003

[2] Daniel J. Bernstein, "DETECTING PERFECT POWERS IN ESSENTIALLY LINEAR TIME", *Mathematics of computation, volume 67, Number 223*, 1998, http://cr.yp.to/papers/powers-ams.pdf