논문 16-41-05-01 The Journal of Korean Institute of Communications and Information Sciences '16-05 Vol.41 No.05
http://dx.doi.org/10.7840/kics.2016.41.5.499

유한체상에서 세제곱근을 찾는 두 종류의 알고리즘

조 국 화◆°

Two Types of Algorithms for Finding the Cube Root in Finite Fields

Gook Hwa Cho◆°

요 약

Cipolla-Lehmer 알고리즘을 향상시킨 새로운 알고리즘을 통해 효율적으로 세제곱근을 찾을 수 있는 방법을 연구하였다. 본 논문에서는 일반적인 Cipolla-Lehmer 알고리즘보다 곱셈량을 줄인 향상된 두 가지 알고리즘을 소개한다. 유한체상에서 세제곱근을 찾는 곱셈량이 비슷한 두 가지 알고리즘을 제안하고, 곱셈량이 비슷하더라도 저장변수의 개수가 적을수록 효율적임을 보인다.

Key Words : cube root algorithm, finite field, Cipolla-Lehmer algorithm, linear recurrence relation

ABSTRACT

We study algorithms that can efficiently find cube roots by modifying Cipolla-Lehmer algorithm. In this paper, we present two type algorithms for finding cube roots in finite field, which improves Cipolla-Lehmer algorithm. If the number of multiplications of two type algorithms has a little bit of a difference, then it is more efficient algorithm which have less storage variables.

Ⅰ. 서 론

유한체 F_p상에서 제곱근을 찾는 문제는 많은 곳에서 응용이 되고 있다. 예를 들어, 미 NIST(National Institute of Standards and Technology)에서 제안한 규정집 (Federal Information Processing Standard 186-3)의 전자서명 프로토콜에서는1 타원곡선을 이용하여 전자서명을 구현하는 방법을 설명하였는데 여기에서 유한체상의 제곱근을 찾는 알고리즘이 필요하다.[7]

마찬가지로 유한체상에서 세제곱근을 찾는 알고리즘 역시 높은 차원의 타원곡선의 암호 시스템에서 응용될 수 있다.

유한체상에서 세제곱근을 찾는 표준적인 두 개의 알고리즘으로는 Tonelli-Shanks 알고리즘[8,9], 그리고 Cipolla-Lehmer 알고리즘[2,5]이 있다. Tonelli-Shanks 알고리즘은 $3^s|p-1$이고 $3^s|p-1$인 3의 지수 s에 의존하고 최악의 경우 복잡도는 $O(\log^4 p)$이지만 Cipolla-Lehmer의 복잡도는 $O(\log^3 p)$이다. 그러므로 s가 상당히 큰 경우에는 Cipolla-Lehmer 알고리즘이 Tonelli-Shanks 알고리즘보다 효율적이다.

일반적인 Cipolla-Lehmer 알고리즘은 기약 다항식 $f(x)=x^3-ax^2+bx-c$이고 c가 삼차 잉여 ($c^{(p-1)/3}=1 \bmod p$)일 때, c의 세제곱근은 $\alpha^{\frac{p^2+p+1}{3}}$으로 찾을 수 있다. 이 때, α는 f의 F_p상의 해이다.

본 논문에서는 일반적인 Cipolla-Lehmer 알고리즘[2,5]보다 위 지수 계산의 곱셈량을 줄이기 위해 특수한

※ 본 논문은 2009년 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (과제번호: 2009-0093827)
◆° First Author : Ewha Womans University, Institute of Mathematical Sciences, ghcho@ewha.ac.kr, 정회원
논문번호 : KICS2016-02-035, Received February 25, 2016; Revised April 26, 2016; Accepted April 27, 2016

형태의 다항식을 선택하여 선형점화식을 이용한 두 가지 알고리즘을 소개하고, 비슷한 곱셈량의 알고리즘이더라도 저장변수의 개수가 적을수록 더욱 효율적임을 보이도록 하겠다.

본 논문의 나머지 부분은 다음과 같다 : 2장에서는, Cipolla-Lehmer가 제시한 세제곱근을 찾는 방법에 대하여 설명한다. 3장에서는 두 가지 방법의 "Double and add"선형 점화식을 이용하여 세제곱근을 찾는 방법에 대해 설명한다. 4장에서는 복잡도와 실험 결과를 보여준다. 마지막으로 5장에서 결론짓는다.

Ⅱ. Cipolla-Lehmer 알고리즘

만약 $p \equiv 1 \bmod 3$인 소수 p에 대해서 c가 삼차 잉여일 때 c의 세제곱근을 찾기 위해 기약 다항식 $f(x) = x^3 - a_2x^2 + a_1x - c$가 필요하다. 만약 $\alpha \in F_{p^3}$가 f의 해이라면, c의 세제곱근은 $\alpha^{\frac{p^2+p+1}{3}}$로 찾을 수 있다. 표 1이 세제곱근을 찾는 Cipolla-Lehmer 알고리즘[2,5]이다. 이 알고리즘은 f의 기약성과 $\alpha^{\frac{p^2+p+1}{3}}$의 지수 계산에서 많은 곱셈량을 필요로 한다. 그리고 이 곱셈량은 f의 계수와 밀접한 관계가 있다.

Nishihara[6]는 삼항 다항식을 이용하여 속도를 향상시켰고, Dickson[3]은 아래 두 가지 성질을 만족하면 기약 다항식 $f(x) = x^3 + bx - c$이 되는 것을 이용하였다. 두 가지 성질을 다음과 같다.

1. $D = -(4b^3 + 27c^2)$가 이차 잉여 ($D^{(p-1)/2} = 1 \bmod p$)이다.

표 1. Cipolla-Lehmer의 삼차 해 알고리즘
Table 1. Cipolla-Lehmer cube root algorithm

입력: 삼차 잉여 $c \in F_p$ **출력**: c의 삼차 해
Step 1. $a, b \in F_p$를 임의로 선택
Step 2. $f(x) \leftarrow x^3 - ax^2 + bx - c$ **if** f: 기약다항식, **then Step 1**
Step 3. $x^{\frac{p^2+p+1}{3}} \bmod f(x)$ 반환

표 2. Dickson의 삼차 다항식 기약성 테스트 알고리즘
Table 2. Dickson's irreducibility testing of cubic polynomials

입력: 삼차 잉여 $c \in F_p$ **출력**: 상수항이 $-c$인 기약 다항식
Step 1. $b \in F_p$를 임의로 선택 $f(x) \leftarrow x^3 + bx - c$, $D(f) \leftarrow -(4b^3 + 27c^2)$
Step 2. **if** $D(f)$가 0이거나 이차 비잉여, **then Step 1** **else** $\alpha \leftarrow \frac{1}{2}(c + 3^{-2}\sqrt{-3D(f)})$ (i.e., α: $x^2 - cx - 3^{-3}b^3 = 0$의 해)
Step 3. **if** α가 삼차 비잉여, **then** $f(x)$ 반환 **else Step 1**

2. $\alpha = \frac{1}{2}(c + 3^{-2}\sqrt{-3D})$가 삼차 비잉여 ($\alpha^{(p-1)/3} \neq 1 \bmod p$)이다.

임의적으로 기약다항식을 선택하는 것보다 더욱 효율적이다. 표 2가 Dickson의 기약성 테스트 알고리즘이다.

Ⅲ. 선형 점화식을 이용한 세제곱근을 찾는 두 종류의 알고리즘

3.1 G.H. Cho[1]의 세제곱근 알고리즘

먼저 G.H. Cho[1] 알고리즘은 삼차 특성수열(characteristic sequence)을 이용하여 세제곱근을 찾는 알고리즘을 소개하였다. 다음 기약 다항식 $f(x) = x^3 - 3x^2 + bx - 1$의 해가 $\alpha \in F_{p^3}$이면 $h(1-\alpha) = 0$인 기약 다항식 $h(x) = x^3 + (b-3)x - (b-3)$이 존재한다. 대각합(trace) 정의가 다음과 같다; $Tr(\alpha) = \alpha + \alpha^p + \alpha^{p^2}$.

G.H. Cho[1] 알고리즘의 결과는 아래와 같다.

정리 1. ([1])
만약 $p \equiv 1 \bmod 9$이고 c가 삼차 잉여라고 가정하자. t로부터 $b = ct^3 + 3$인 $f(x) = x^3 - 3x^2 + bx - 1$이고 $f(\alpha) = 0$일 때, 만약 $f(x)$가 기약 다항식이면, $t^{-1} \cdot Tr(\alpha^{\frac{p^2+p-2}{9}})$는 c의 삼차 해이다.

$Tr(\alpha^{\frac{p^2+p-2}{9}})$를 계산하기 위해서, 삼차 특성 수열을 이용하였다. 삼차 특성 수열은 $f(x)=x^3-ax^2+bx-c\,(a,b,c\in F_p)$가 기약 다항식이면 다음과 같이 정의된다; $s_k=as_{k-1}-bs_{k-2}+cs_{k-3},\ k\geq 3$. 만약 $f(\alpha)=0$이면 $s_k(\alpha)=Tr(\alpha^k)=\alpha^k+\alpha^{kp}+\alpha^{kp^2}$ 등식을 만족한다. 삼차 수열은 잘 알려진 "Double and add"관계식이 있다.[4] 초깃값이 $s_0=3, s_1=3, s_2=9-2b$이므로 $s_{2n}=s_n^2-2s_{-n}$과 $s_{n+m}=s_ns_m-s_{n-m}s_{-m}+s_{n-2m}$이다. 이 관계식을 이용하여 $s_{\frac{p^2+p-2}{9}}$를 계산하면 G.H. Cho[1] 알고리즘은 평균적으로 $15\log p$ 곱셈량을 필요로 한다. 표 3은

표 3. s_m을 계산하는 알고리즘
Table 3. Algorithm for computing s_m

입력: $b\in F_p$이고 $m=\sum_{j=0}^{l}k_j2^j$

출력: s_m

Step 1.

$d_0\leftarrow 3,\ d_1\leftarrow 3,\ d_2\leftarrow 9-2b$

$a_0\leftarrow 3,\ a_1\leftarrow b,\ a_2\leftarrow b^2-6$

Step 2.

for j **from** $l-1$ **down to** 1 **do**

if $k_j=0$ **then**

$d_0\leftarrow a_2-ba_1+d_0d_1$
$d_1\leftarrow d_1^2-2a_1$
$d_2\leftarrow a_0-3a_1+d_1d_2$
$a_0\leftarrow d_2-3d_1+a_0a_1$
$a_1\leftarrow a_1^2-2d_1$
$a_2\leftarrow d_0-bd_1+a_1a_2$

if $k_j=1$ **then**

$d_0\leftarrow d_1^2-2a_1$
$d_1\leftarrow a_0-3a_1+d_1d_2$
$d_2\leftarrow d_2^2-2a_2$
$a_0\leftarrow a_1^2-2d_1$
$a_1\leftarrow d_0-bd_1+a_1a_2$
$a_2\leftarrow a_2^2-2d_2$

Step 3.

$w_1\leftarrow a_0-3a_1+d_1d_2,\ w_2\leftarrow d_1^2-2a_1$

if $k_0=1$ **then** w_1 반환

else w_2 반환

위에서 설명한 "Double and add"관계식을 이용하여 s_m을 계산하기 위한 알고리즘이다.

3.2 새로운 세제곱근 알고리즘

G.H. Cho[1]가 제안한 알고리즘에서 $Tr(\alpha^{\frac{p^2+p-2}{9}})$을 계산하기 위해 삼차 특성 수열을 이용하지 않고 고전적인 "Double and add"관계식을 이용해서 구해보도록 하겠다. 즉 $f(x)=x^3-3x^2+bx-1$으로부터 $s_{\frac{p^2+p-2}{9}}(\alpha)$를 계산하기 위해 $\{1,\alpha,\alpha^2\}$을 기저로 하는 $s_m(\alpha)=x_m+y_m\alpha+z_m\alpha^2$로 표현되는 3개의 배열 $[x_m,y_m,z_m]$을 생각할 수 있다. 초깃값이 $x_1=0,\ y_1=1,\ z_1=0$이므로 아래와 같은 관계식을 얻을 수 있다.

1. $x_{2n}=x_n^2+(2y_n+3z_n)z_n$,
2. $y_{2n}=2(x_n-bz_n)y_n+(1-3b)z_n^2$,
3. $z_{2n}=2z_n(x_n+6y_n)+y_n^2+(9-b)z_n^2$,
4. $x_{2n+1}=z_{2n}$,
5. $y_{2n+1}=x_{2n}-bz_{2n}$,
6. $z_{2n+1}=y_{2n}+3z_{2n}$.

위 관계식을 이용하여 $Tr(\alpha^{\frac{p^2+p-2}{9}})$계산을 $\left[x_{\frac{p^2+p-2}{9}},y_{\frac{p^2+p-2}{9}},z_{\frac{p^2+p-2}{9}}\right]$로 표현할 수 있다.

따름정리 1.

정리 1과 같은 조건을 가정하면, 우리는 $1+\alpha^{\frac{p-1}{3}}+\alpha^{\frac{p^2-1}{3}}=0$을 얻을 수 있다. 즉, $1+\alpha^{\frac{1-p}{3}}+\alpha^{\frac{1-p^2}{3}}=1-\alpha$이다.

(증명)

$h(1-\alpha)=0$이므로 $(1-\alpha)^3=(b-3)\alpha$이다.

양변에 $\frac{p-1}{3}$제곱을 해주면

$(1-\alpha)^{p-1}=(b-3)^{\frac{p-1}{3}}\alpha^{\frac{p-1}{3}}=\alpha^{\frac{p-1}{3}}$이다.

$$\begin{aligned}1+\alpha^{\frac{p-1}{3}}+\alpha^{\frac{p^2-1}{3}}&=1+(1-\alpha)^{p-1}+(1-\alpha)^{p^2-1}\\&=1+(1-\alpha)^{-1}[(1-\alpha^p)+(1-\alpha^{p^2})]\\&=1+(1-\alpha)^{-1}(\alpha-1)\\&=0\end{aligned}$$

위 등식의 양변에 α를 곱하면

$1+\alpha^{\frac{1-p}{3}}+\alpha^{\frac{1-p^2}{3}}=1-\alpha$이다.

정리 2.

정리 1과 같은 조건을 가정하면, 다음 등식이 만족한다. $Tr(\alpha^{\frac{p^2+p-2}{9}}) = x_{\frac{p^2+p-2}{9}} - z_{\frac{p^2+p-2}{9}}$.

(증명)위 따름정리 1로부터 증명할 수 있다.

$$\begin{aligned} Tr(\alpha^{\frac{p^2+p-2}{9}}) &= \alpha^{\frac{p^2+p-2}{9}}(1+\alpha^{\frac{1-p}{3}}+\alpha^{\frac{1-p^2}{3}}) \\ &= \alpha^{\frac{p^2+p-2}{9}}(1-\alpha) \\ &= (x_{\frac{p^2+p-2}{9}} + y_{\frac{p^2+p-2}{9}}\alpha + z_{\frac{p^2+p-2}{9}}\alpha^2)(1-\alpha) \\ &= x_{\frac{p^2+p-2}{9}} - z_{\frac{p^2+p-2}{9}} \end{aligned}$$

표 4는 x_n, y_n, z_n을 계산하기 위한 알고리즘이다.

표 4. x_n, y_n, z_n을 계산하는 알고리즘
Table 4. Algorithm for computing x_n, y_n, z_n

입력: 삼차 잉여 $c \in F_p$
출력: x_n, y_n, z_n

Step 1.
$x_1 \leftarrow 0,\ y_1 \leftarrow 1,\ z_1 \leftarrow 0$

Step 2.

$$\frac{p^2+p-2}{9} = \sum_{j=0}^{l} k_j 2^j$$

for j **from** $l-1$ **down to** 0 **do**
$x_n \leftarrow x_n^2 + (2y_n + 3z_n)z_n$
$y_n \leftarrow 2(x_n - bz_n)y_n + (1-3b)z_n^2$
$z_n \leftarrow 2z_n(x_n + 6y_n) + y_n^2 + (9-b)z_n^2$
if $k_j = 1$ **then**
$x_n \leftarrow z_n$
$y_n \leftarrow x_n - bz_n$
$z_n \leftarrow y_n + 3z_n$

Step 3.
x_n, y_n, z_n 반환

Ⅳ. 복잡도와 실험 결과

일반적인 Cipolla-Lehmer 알고리즘[2,5]은 임의적으로 a, b를 선택하므로 세제곱근을 찾기 위해서 일반적인 2차 다항식의 제곱 $(b_2x^2 + b_1x + b_0)^2 \bmod f(x)$의 계산을 필요로 한다. 그러므로 적어도 $24\log p$ 곱셈량을 필요로 한다. 하지만 3장에서 소개한 G.H. Cho[1] 알고리즘은 평균적으로 $7.5\log p^2 \approx 15\log p$ 곱셈량을 필요로 하고, 새로운 알고리즘 역시 $8.5\log p^2 \approx 17\log p$ 곱셈량을 필요로 하므로 일반적인 Cipolla-Lehmer 알고리즘보다 향상된 알고리즘이라고 할 수 있다. 또한, 표 3의 알고리즘은 $k_j = 0$일 때, 6개의 저장 변수를 이용하여 8번의 곱셈을 필요로 하고, $k_j = 1$인 경우는 6개의 저장 변수를 이용하여 7번의 곱셈을 필요로 한다. 즉 평균적으로 6개의 저장 변수를 이용하여 7.5번의 곱셈량을 필요로 한다. 표 4의 알고리즘은 $k_j = 0$일 때, 3개의 저장 변수를 이용하여 8번의 곱셈을 필요로 하고, $k_j = 1$인 경우는 6개의 저장 변수를 이용하여 9번의 곱셈을 필요로 한다. 즉 평균적으로 4.5개의 저장 변수를 이용하여 8.5번의 곱셈량을 필요로 한다. 표 5는 수학 계산 프로그램 SAGE를 이용하여 실험한 결과이다.

표 5. 삼차 해를 계산한 실행 시간
Table 5. Running time (in seconds) for cube root computation

p(bit size)	3000	4000	5000	6000	7000
C-L algo.	0.34	0.73	1.29	1.95	3.03
C-K-K algo.	0.28	0.60	1.10	1.63	2.75
New algo.	0.23	0.45	0.79	1.18	1.82

Ⅴ. 결 론

본고에서는 유한체 상에서 세제곱근을 찾는 두 종류의 알고리즘을 소개하였다. 소개한 두 종류의 알고리즘은 일반적인 Cipolla-Lehmer 알고리즘보다 곱셈량을 줄여 보다 효율적임을 보였다. G.H. Cho[1] 알고리즘은 6개의 저장 변수로 7.5개의 곱셈 계산을 필요로 했고, 새로운 알고리즘은 4.5개의 저장 변수로 8.5개의 곱셈 계산을 필요로 하였다. 곱셈량이 크게 차이나지 않는다면 저장 변수가 적을수록 효율적임을 알 수 있다.

References

[1] G. H. Cho, N. Koo, E. Ha, and S. Kwon, "New cube root algorithm based on the third order linear recurrence relations in finite fields," *Designs, Codes and Cryptography*,

vol. 75, no. 3, pp. 483-495, 2015.

[2] M. Cipolla, “Un metodo per la risoluzione della congruenza di secondo grado,” *Rendicono dell’Accademia Scienze Fisiche e Matematiche*, vol. 9, no. 3, pp. 154-163, 1903.

[3] L. E. Dickson, “Criteria for the irreducibility of functions in a finite field,” *Bull. Am. Math. Soc.*, vol. 13, no. 1, pp. 1-8, 1906.

[4] G. Gong, L. Harn, and H. Wu, “The GH public-key cryptosystem,” *Selected Areas in Cryptography*, Springer Berlin Heidelberg, pp. 284-300, Dec. 2001.

[5] D. H. Lehmer, “Computer technology applied to the theory of numbers,” Studies in Number Theory, *Math. Assoc. Am*. (distributed by Prentice-Hall, Englewood Cliffs, N. J.), pp. 117-151, 1969.

[6] N. Nishihara, R. Harasawa, Y. Sueyoshi, and A. Kudo, *A remark on the computation of cube roots in finite fields*, preprint, 2009, from http://eprint.iacr.org/2009/457.pdf.

[7] NIST, *Digital Signature Standard*, Federal Information Processing Standard 186-3, 2000, from http://csrc.nist.gov/publications/fips/.

[8] D. Shanks, “Five number-theoretic algorithms,” in *Proc. Second Manitoba Conf. Numerical Math.*, pp. 51-70, Winnipeg, Canada, Oct. 1972.

[9] A. Tonelli, “Bemerkung über die Auflösung Quadratisher Congruenzen,” *Göttinger Nachrichten*, pp. 344-346, 1891.

조 국 화 (Gook Hwa Cho)

2007년 8월 : 전북대학교 수학과 학사

2011년 2월 : 성균관대학교 수학과 석사

2015년 2월 : 성균관대학교 수학과 박사

2015년 3월~8월 : 전북대학교 강의전담

2015년 9월~현재 : 이화여자대학교 수리과학연구소 박사 후 연구원

<관심분야> 정수론, 공개키 암호 시스템, NFS, Root extraction