Historia Mathematica Vol. 12, No. 2, 1999, 47-54

나머지 수 체계의 부활*

아주대학교 정보 및 컴퓨터공학부 **예홍진**

Abstract

We introduce some historical facts on number theory, especially prime numbers and modular arithmetic. And then, with the viewpoint of computer arithmetic, residue number systems are considered as an alternate to positional number systems so that high performance and high speed computation can be achieved in a specified domain such as cryptography and digital signal processing.

0. 서론

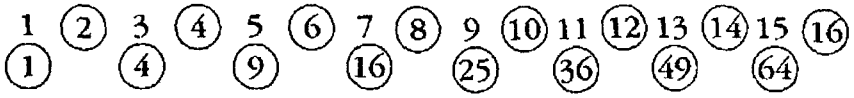
정수론은 수학의 여러 분야 중에서도 매우 오래된 학문 분야 중의 하나이다. 기원전 5세기경 피타고라스를 중심으로 한 학파는 모든 사물을 자연수(natural number)로 설명하였으며, 자연을 합리적으로 이해하려면 어떤 특정 수(數, number)의 성질을 분석하면 된다고 생각할 정도였다. 예를 들어 이성(理性)은 진리(眞理)를 생성한다는 의미에서 1은 이성을 나타내고, 4는 타당성(妥當性)에 대한 기호로 정하여 수(數)의 과학(科學)을 신비(神秘)의 철학(哲學)으로 발전시켜 나갔다. 또한, 2와 3이 각각 남자와 여자를 나타낸다고 할 때, 5는 2와 3의 합, 즉 남자와 여자의 결합이라는 의미에서 결혼(結婚)을 나타낸다고 보았다[8].

이러한 수에 대한 연구는 시대와 문화의 변천에 따라 다양한 수 체계(number systems)로 발전하게 되었고, 분수(分數)나 소수(小數)의 도입은 물론 일정한 규칙에 따라 수를 배열(配列, arrangement)함으로써 파스칼의 삼각형과 같은 특별한 패턴(pattern)을 찾아내게 되었다. 예를 들어, Moessner가 제시한 재미있는 배열을 소개하면 다음과 같다[1].

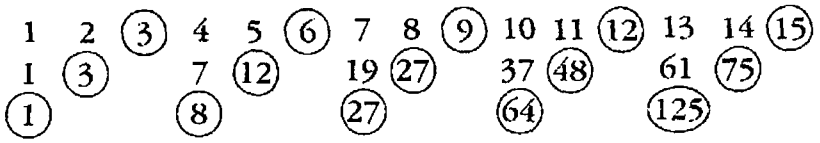
먼저 숫자 위에 원을 그리는 규칙을 정하고 난 뒤에 원이 그려지지 않은 숫자들을 왼쪽부터 차례로 더하여 그 아래에 기록하면 신기한 사실을 발견할 수 있다. 즉, 왼쪽부터 차례로 자연수를 배열하고 한 칸씩 건너서 원을 그리면 수의 배열로부터 모든 제곱수를 구할 수 있으며, 두 칸씩 건너서 원을 그리면 다음과 같이 모든 세제곱수를 얻을 수 있다. 이러한 방법

* 본 논문은 1994년도 한국학술진흥재단의 공모과제 연구비 지원에 의하여 수행되었음.

으로 임의의 N제곱수를 계산하려면 N칸씩 건너서 원을 그리고 덧셈을 반복하면 된다.



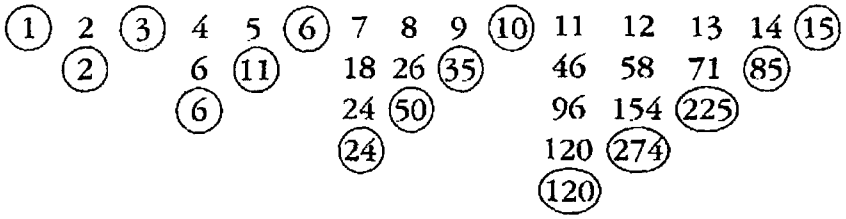
(a) 제곱수의 배열



(b) 세제곱수의 배열

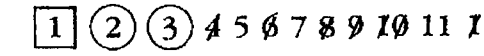
(그림1) 수의 배열과 덧셈만을 이용한 제곱수와 세제곱수의 계산

만일 일정한 간격이 아니라 1에 원을 그리고 한 칸 건너 3에 원을 그리고 두 칸 건너 6에 원을 그리는 식으로 간격이 하나씩 증가하면 다음과 같이 몇 번의 덧셈만으로 $n!$의 값을 손쉽게 계산할 수도 있다[1]. 얼마나 신기한 수(數)의 장난인가?



(그림2) 덧셈으로만 계산된 계승에 대한 배열

소수(素數, prime number)에 얽힌 역사는 더욱 흥미롭다. 기원전 2세기경 에라토스테네스는 소수(素數)를 찾기 위한 방법을 다음과 같이 제시했다. 즉, 왼쪽부터 차례로 자연수를 배열하고 1은 단위(unit)를 나타내는 수로서 제외하고, 왼쪽부터 첫 번째로 남아있는 2위에 원을 그리고 오른쪽으로 2칸씩 이동하면서 모든 2의 배수를 지운다. 다시 왼쪽부터 첫 번째로 남아있는 3위에 원을 그리고 오른쪽으로 3칸씩 이동하면서 모든 3의 배수를 지우고, 이러한 작업을 반복함으로써 소수(素數)를 찾을 수 있다.



(그림3) 에라토스테네스의 방법에 의한 소수 판별법

이러한 작업은 19세기까지 이어져 1909년 D. N. Lehmer의 저서 *Factor Table for the First Ten Million*에서 에라토스테네스의 방법을 이용해 100만부터 1000만 사이의 소수는 물론 합성수인 경우에는 모든 약수들을 구한 결과를 제시하였다[1].

1. 큰 수와 작은 수의 조화

임의의 정수 P가 소수일 때, P로 나눈 나머지들에 대한 많은 연구결과는 정수론은 물론 여러 응용분야에서 새로운 가능성을 보여주는 좋은 계기가 되었다. 예를 들어, 모듈라 산술(modular arithmetic) 혹은 나머지 산술(residue arithmetic)에서 곱셈에 대한 역원(逆元)을 구하는 문제를 생각해 보자. $2\times4\equiv1 \pmod 7$이므로 $\frac{1}{2}\equiv4 \pmod 7$로 표현한다면 $\frac{1}{3}\equiv5 \pmod 7$, … 등과 같이 0을 제외한 7로 나눈 나머지의 곱셈에 대한 역원을 계산할 수 있다.

그렇다면 $\frac{1}{2}, \frac{1}{3}, \cdots, \frac{1}{P-1} \pmod P$을 일반적으로 어떻게 계산할 수 있는가? P가 소수(素數)일 때, 한 예로 $\frac{1}{8} \pmod{101}$을 계산해보자. 101보다 큰 8의 배수 중에서 가장 작은 것은 $8\times13=104$이므로 $8\times13\equiv3 \pmod{101}$이 성립함을 알 수 있다. 이번에는 101보다 큰 3의 배수 중에서 가장 작은 것은 $3\times34=102$이므로 $3\times34\equiv1 \pmod{101}$이 성립한다. 따라서, $8\times13\times34\equiv3\times34\equiv1 \pmod{101}$이라는 식에

$$13\times34\equiv(4\times3+1)\times34\equiv4\times(3\times34)+34\equiv38 \pmod{101}$$

을 대입하면, $8\times38\equiv1 \pmod{101}$로부터 $\frac{1}{8}\equiv38 \pmod{101}$임을 손쉽게 계산할 수 있다. 이와 같이 소수(素數)를 제수(除數, divisor)로 하는 모듈라 산술은 매우 큰 수에 대한 계산 문제를 상대적으로 작은 수들에 대한 계산들을 반복적으로 적용함으로써 해결할 수 있는 실마리를 제공하고 있다.

컴퓨터 관련 분야에서 모듈라 산술(modular arithmetic)이 이용되기 시작한 것은 불과 반세기도 채 안되지만, 그 역사적 뿌리는 정수론(number theory)에서 찾아볼 수 있다. 서기 1세기경으로 거슬러 올라가 중국의 고대 수학자인 Sun-Tsu가 기술한 문헌에 의하면 3, 5, 7

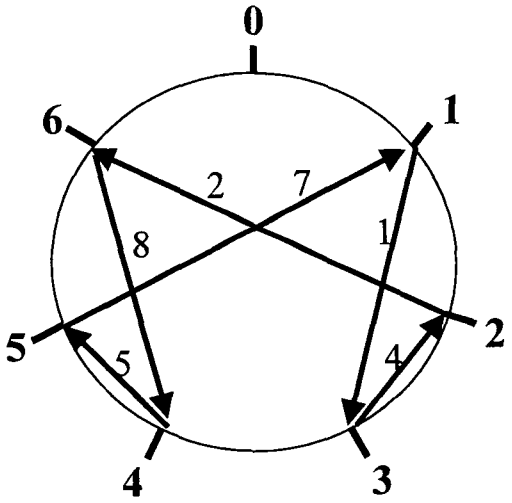
로 나누었을 때 나머지가 각각 2, 3, 2 인 수를 구하는 문제가 최초의 기록인 셈이다. 이후 많은 수학자들이 특정 제수(除數, divisor)들의 집합에 대하여 주어진 나머지들을 이용해 원래의 정수를 찾는 문제들을 다루었으며, 이와 같은 유형의 문제들에 대한 체계적인 연구와 일반적인 해법은 19세기에 이르러 K. F. Gauss에 의하여 정리되었다. 이러한 역사적인 이유로 해서 다음과 같은 정수(number)의 모듈라 표현에 관한 기본적인 정리를 중국인의 나머지 정리(Chinese Remainder Theorem)라고 부른다[5].

[Chinese Remainder Theorem] k개의 서로 다른 양의 정수 $m_1, m_2, \cdots, m_k$에서 임의로 선택한 각 쌍이 서로 소(prime) 즉, $\gcd(m_i, m_j)=1$(단, $i \neq j$)이 성립할 때, 임의의 정수 $a, u_1, u_2, \cdots, u_k$에 대하여 다음의 두 가지 조건을 동시에 만족하는 정수 u는 유일하다.

(조건 1) $a \leq u < a + m_1 m_2 \cdots m_k$

(조건 2) $u \equiv u_j \pmod{m_j}$ (단, $1 \leq j \leq k$)

이 정리는 정수론과 대수학 분야에서 잉여류(剩餘類, residue class)에 대한 많은 이론들이 정립될 수 있는 기반을 제공하게 되었다[3]. 예를 들어, 분수(分數)를 소수(小數, fraction)로 변환하는 문제를 생각해 보자. 예를 들어, 1/7은 0.14285714285714…과 같이 '142857'이라는 순환절(循環節)의 길이가 6인 순환소수(循環小數)로 나타낼 수 있다. 이와 같이 순환절의 길이나 순환되는 숫자들의 순서를 알아내는 것과 분모(分母)에 대한 나머지 계산과는 아주 밀접한 관계가 있다.



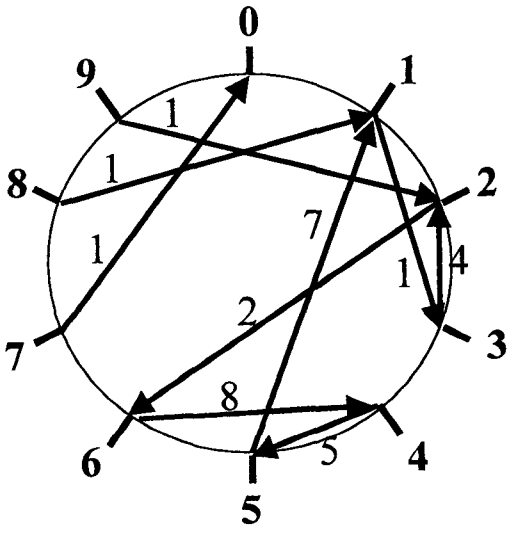
(그림4) 7로 나눈 나머지의 행동도

앞의 그림은 원주 위에 일정한 간격으로 7로 나눈 나머지, 즉 0, 1, …, 6을 일정한 간격으로 배열하고 그 중의 한 숫자 a를 선택하여 $10\times a = b\times 7 + c$가 성립하는 b와 c의 값을 구한 다음 a에서 c로 화살표를 그리고 그 위에 b를 표시한 것이다. 이러한 그림을 나머지(residue)의 행동도(行動圖)혹은 수(數)의 회전목마(merry-go-round)라 부르며, 이를 이용하여 임의의 숫자에서 출발하여 자기 자신으로 돌아오기 위한 '142857'이라는 순환절을 찾는 것은 물론 순환절의 길이도 손쉽게 구할 수 있다[9].

이러한 특성은 컴퓨터 산술에도 곧바로 적용될 수 있다. 즉, 임의의 10진수를 7진수로 변환하거나 또는 7로 나눈 나머지를 계산하기 위하여 가장 높은 자리수(most significant digit)부터 가장 낮은 자리수(least significant digit)방향으로 한 자리씩 차례로 주어질 때 다음 그림과 같이 10진수에 대한 7로 나눈 나머지의 행동도를 이용하면 나눗셈을 사용하지 않고 빠르게 계산할 수 있다. 예를 들어, 십진수 2583을 7로 나눈 나머지를 구하려면

$$2 \equiv 2 \pmod 7$$
$$2\times 10 + 5 \equiv 6 + 5 \equiv 4 \pmod 7$$
$$4\times 10 + 8 \equiv 5 + 1 \equiv 6 \pmod 7$$
$$6\times 10 + 3 \equiv 2 + 3 \equiv 5 \pmod 7$$

을 차례로 계산하는 과정에서 최종적으로 $2583 \equiv 5 \pmod 7$을 얻을 수 있다. 이러한 특성을 일반화하면 M진법의 수를 R로 나눈 나머지를 구하는 데에 적용할 수 있으며, 이와 같이 숫자열에 대하여 가장 높은 자리수부터 가장 낮은 자리수 방향으로 차례로 계산하는 방식을 '온라인 산술'(on-line arithmetic)이라 부른다. 이러한 나머지 계산은 전통적인 위치적기수법(positional number system)인 기수법으로 표현된 수를 나머지 수 체계(residue number system)로 변환하는 데에 효율적으로 사용될 수 있다.



(그림5) 10진수를 7로 나눈 나머지를 계산하기 위한 나머지의 행동도

실제 나눗셈의 계산 없이 임의의 소수 P로 나눈 나머지를 구하는 데에 사용되는 나머지의 행동도는 조그마한 테이블 형태로 저장되어 'table look-up'기술을 활용하면 손쉽게 하드웨어적으로 구현될 수 있으며, 산술연산자(arithmetic operator)를 직접 제작하기 위하여 기존의 VLSI 회로설계 기법을 적용하는 것에 비해 획기적으로 계산속도를 높이고 제작비용을 줄일 수 있다.

그러나, 나머지 수 체계(residue number systems)는 정수들간의 산술연산에서 기존의 위치적 기수법에 비하여 효율적인 반면에, 앞의 예에서 보았듯이 분수(分數)나 소수(小數)를 다루는 데에는 오히려 계산이 더욱 복잡해지는 단점 때문에 범용(general purpose) 컴퓨터에 곧바로 적용하기 위해서는 해결되어야 할 문제들이 아직도 많이 남아있다. 또한 두 수간의 대소 비교(magnitude comparison), 음수 표현(negative number representation) 및 부호 판정(sign detection), 오버플로우(overflow)등과 같은 산술연산을 처리하는 데에도 기존의 수체계에 비하여 상대적으로 많은 문제점을 가지고 있다. 이러한 단점에도 불구하고 현재 컴퓨터 관련 분야로서 암호학(cryptography)이나 디지털 신호 처리(digital signal processing) 등과 같이 큰 수간의 덧셈, 곱셈 등과 같은 반복적인 계산이 요구되는 문제를 해결하는 데에 주로 사용되고 있다[4][6].

2. 나머지 수 체계와 컴퓨터의 만남

나머지 수 체계의 가장 큰 장점 중의 하나는 덧셈이나 곱셈 등과 같은 산술 연산(arithmetic operation)을 계산함에 있어서 두 수간의 산술연산이 각각의 제수(divisor)에 대한 나머지들 간의 산술 연산들로 분리되어 서로 독립적으로 처리될 수 있다는 것이다[2]. 즉, 임의의 모듈리(moduli) 쌍에 대하여 $(m_j, m_k)=1$ (단 $j\neq k$)이 성립하고 $1\leq i\leq k$을 만족하는 모든 i에 대하여 각각 $x_i = X \bmod m_i$, $y_i = Y \bmod m_i$이라 정의하면,

$$\begin{aligned} X \circledcirc Y &= (x_1, x_2, \cdots, x_k) \circledcirc (y_1, y_2, \cdots, y_k) \\ &= [x_1 \circledcirc y_1 \ (\bmod m_1), x_2 \circledcirc y_2 \ (\bmod m_2), \cdots, x_k \circledcirc y_k \ (\bmod m_k)] \end{aligned}$$

특히, 컴퓨터 산술의 관점에서 볼 때 일반적으로 수(數)를 나타내기 위한 물리적인 비트(bit) 수에는 현실적으로 한계가 있다. 이러한 수의 표현방법에 대한 제약은 곧 자연수는 물론 정수나 유리수, 실수 등과 같은 수학적 의미의 수(數)를 자유롭게 표현할 수 없다는 점을 고려하면 나머지 수 체계는 우리에게 다음과 같은 가능성을 제시해 주고 있다.

첫째, 매우 큰 수를 물리적으로 다룰 수 없을 때 적당히 선택된 제수(divisor)들의 집합에 의해 정의된 나머지 수 체계를 적용함으로써 실제 컴퓨터에서 표현 가능한 작은 수들의 집합으로 나타낼 수 있을 뿐만 아니라, 상대적으로 훨씬 작은 수들에 대한 산술 연산을 처리함으로써 하드웨어적인 수의 표현 한계를 극복할 수 있다.

둘째, 매우 큰 수에 대한 산술 연산을 처리함에 있어서 특별한 하드웨어나 소프트웨어를 개발하는 대신에 범용 마이크로프로세서를 대량으로 병렬로 연결하여 사용(massively parallel processing)함으로써 계산속도를 획기적으로 개선시키거나, VLSI 병렬 구조를 도입하여 ASIC(Application Specification Integrated Circuit)과 같은 형태로 발전시켜 특정 문제 해결에 적합한 해결방안을 제시할 수 있다[7].

셋째, 나머지 수 체계를 정의함에 있어서 $2^n \pm 1$의 형태를 가진 소수(素數), 즉 페르마의 수나 메르센의 수와 같은 특별한 소수를 제수로 선택함으로써 2진법에 기반을 두고있는 컴퓨터의 속성상 나머지의 행동도는 물론 여러 단계의 산술적인 계산과정을 획기적으로 단순화시킬 수 있는 여지가 많다.

이러한 가능성들은 특히 정보통신분야의 기술개발과 밀접한 관계를 맺고 있으며 암호학이나 반도체 분야는 물론 대규모 산술 연산을 동반하는 과학기술분야에 활용가능성이 매우 높다

3. 결론

정보통신 분야의 눈부신 기술발전과 더불어 디지털 시대가 도래하면서 역사상 어느 시대보다도 수학자가 아닌 사람들에 의하여 새로운 관점에서 수학적 지식과 연구결과들이 재해석되고 다양한 분야에 걸쳐 활발하게 적용되고 있다. 그러한 의미에서 인류의 역사만큼이나 유래가 깊은 정수론에 대한 고찰과 함께 최근 컴퓨터 분야에서 제기되고 있는 나머지 수 체계의 활용가능성을 모색하는 것은 마치 위대한 고고학적 유물을 탐사하듯이 가슴이 설레는 연구 주제가 아닐 수 없다. 따라서 수학사에 대한 관심과 연구는 단순히 과거의 수학자들에 대한 업적이나 학문 발전의 흐름을 이해하는 수준을 뛰어넘어, 시대적 요구와 사회적 필요에 따라 재조명되어 새로운 기술 발전의 근본적인 원동력이 될 수 있을 것이다.

참고 문헌

1. Conway, John H., and Guy, Richard K., *The book of Numbers*, Springer-Verlag, 1996.
2. Koren, Israel, *Computer Arithmetic Algorithms*, Prentice-Hall, 1993.
3. Knuth, Donald E., *The Art of Computer Programming*, 2d Ed., Vol. 2, *Seminumerical Algorithms*, Addison-Wesley, 1981.
4. Rhee, Man Young, *Cryptography and Secure Communications*, McGraw-Hill, 1994.
5. Szabo, Nicolas S. and Tanaka, Richard I., *Residue Arithmetic and Its Applications to Computer Technology*, McGraw-Hill, 1967.
6. Soderstrand, M. A., Jenkins, W. K., Jullien, G. A., and Taylor F. J., *Residue Number*

System Arithmetic : Modern Applications in Digital Signal Processing, IEEE Press, 1986.

7. Taylor, Fred J., "Residue Arithmetic: A Tutorial with Examples," *Computer*, vol. 17, no. 5, IEEE press, 1984, pp. 50-62.
8. 김응태, 박승안, **정수론** 제4판, 경문사, 1997.
9. 기무라 요시오, **퍼스컴으로 즐기는 수학-BASIC의 기초부터 그래픽까지**, (원저: Blue Backs B-671, 고단사, 1986, 일본), 김현숙 옮김, 컴퓨터시리즈 6권, 전파과학사, 1991.