

암호학과 네트워크 보안

-9장 암호수학(1)-

민 윤 홍(muh4316@gmail.com)

세종대학교 프로토콜공학연구실

목 차

- 소수
- 소수 판정
- 지수와 로그

목 차

- 소수
- 소수 판정
- 지수와 로그

소수

- 개요

- 양의 정수(Positive Integers)

- 숫자 1

- 소수

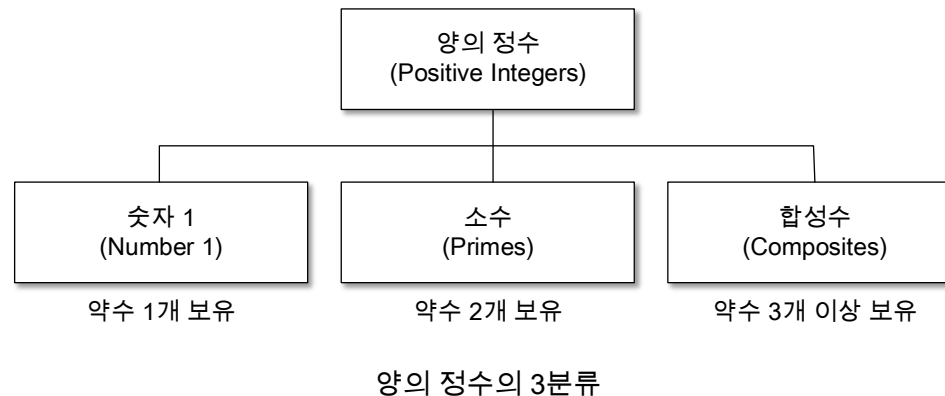
- 양의 정수에서 오직 1이나 자신으로만 나누어 떨어지는 수

- 합성수

- 1과 자기 자신 이외의 양의 약수를 가진 양의 정수

- 서로소(Coprime)

- 최대공약수가 1인 두 양의 정수의 관계



소수

- 소수 집합의 크기 (1/3)

- 무한히 많은 소수
 - 증명

p	가장 큰 소수
q	$q \leq p$ 인 소수
r	소수를 모두 곱한 수

- 가정 : 소수의 개수는 유한개
- 모순 증명을 이용하여 다음과 같이 증명
- 정수 $r + 1$ 은 q 를 약수로 갖지 못함
 - $(r + 1) \bmod q = 1$
- 만약 그런 소수 q 가 존재한다면 q 는 r 과 $r + 1$ 를 모두 나눌 수 있음
- 그런 수는 오직 1밖에 존재하지 않음
- 1은 소수가 아니므로 q 는 1이 될 수 없음
- 처음 가정이 틀렸음
- 따라서 소수의 개수는 무한개임

소수

• 소수 집합의 크기 (2/3)

• 예제 9.1

소수 전체 집합(p)이 $\{2, 3, 5, 7, 11, 13, 17\}$ 일 경우, 17보다 큰 소수가 존재함을 증명하시오

- 소수 전체 집합의 곱 $r = 510510$
- $r + 1 = 510511 = 19 \times 97 \times 277$
- 19, 97, 277은 소수 전체 집합에 미포함
- 17보다 큰 소수가 3개 있다는 것이 증명

p	17
q	$\{2, 3, 5, 7, 11, 13, 17\}$
r	510510

소수

- 소수 집합의 크기 (3/3)

- 소수의 개수

- 공식화

- 함수 $\pi(n)$ 은 n 보다 작거나 같은 소수의 개수를 나타내는 함수라고 정의

- e.g., $\pi(1) = 0$, $\pi(2) = 1$, $\pi(3) = 2$, $\pi(10) = 4$, $\pi(100) = 25$

- 만약 n 이 매우 커지게 된다면, $\pi(n)$ 은 근사로 측정

$$[n/(\ln n)] < \pi(n) < [n/(\ln n - 1.08366)]$$

소수

- 소수 판정 방법 (1/3)

- 주어진 수 n 이 $\sqrt{n}$ 보다 작은 소수로 나누어지는지 확인
 - 나누어지면 합성수, 나누어지지 않으면 소수
- 예제 9.2

수 97은 소수인지 판별해보자

- $9 < \sqrt{97} < 10$ 이므로 9보다 작은 소수는 2, 3, 5, 7
 - 97이 네 개의 소수로 나누어지는지 확인해보면 어느 소수도 97을 나누는 것은 불가능
- $\therefore$ 97은 소수

소수

- 소수 판정 방법 (2/3)
 - 에라토스테네스의 체(Sieve of Eratosthenes)
 - 필요성
 - 주어진 수 n 보다 작은 모든 소수를 표를 그려 찾는 방법
 - 방법
 - $\sqrt{n}$ 이하의 소수들로 나누어 떨어지는 수를 모두 지워 확인함
 - 특징
 - 직관적임
 - n 이 작을 수록 빠름
 - n 이 클 경우 모든 수를 표기하지 못하기 때문에 한계가 있음

소수

- 소수 판정 방법 (3/3)

- 에라토스테네스의 체(Sieve of Eratosthenes)
 - 예제 9.3

100이하의 숫자 중에서 소수를 에라토스테네스의 체를 이용하여 구해보자

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100



	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

100이하의 소수는 2, 3, 5, 7, 11, 13, 17, 19, 23, 29
31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

소수

- 오일러의 φ 함수 (Euler's Phi-function) (1/2)

- 정의

- 양의 정수 n 보다 작으면서 n 과 서로소인 정수의 개수

- 성질

1. $\varphi(1) = 1$

2. $\varphi(p) = p - 1$, p 는 소수

e.g., $\varphi(13) = 13 - 1 = 12$

3. $\varphi(m \times n) = \varphi(m) \times \varphi(n)$, m 과 n 은 서로소

e.g., $\varphi(15) = \varphi(3) \times \varphi(5) = 2 \times 4 = 8$

4. $\varphi(p^e) = p^e - p^{e-1}$, p 는 소수

e.g., $\varphi(16) = \varphi(2^4) = (2^4 - 2^3) = 8$

$*p^e$
여기서의 e 는 정수 집합 오일러의 φ 함수는
양의 정수에서만 정의되므로 $e \geq 1$

소수

- 오일러의 φ 함수(Euler's Phi-function) (2/2)

- 성질 3, 4를 이용하여 $\varphi(n)$ 구하기

- $\varphi(n) = (p_1^{e_1} - p_1^{e_1-1}) \times (p_2^{e_2} - p_2^{e_2-1}) \times \dots \times (p_k^{e_k} - p_k^{e_k-1})$
($n = p^{e_1} \times p^{e_2} \times \dots \times p^{e_k}$)

- 예제 9.4

$\varphi(240)$ 를 구하시오

- $240 = 2^4 \times 3^1 \times 5^1$
 - $\varphi(240) = \varphi(2^4) \times \varphi(3^1) \times \varphi(5^1)$
 $= (2^4 - 2^3) \times (3^1 - 3^0) \times (5^1 - 5^0)$
 $= 64$

소수

- 페르마의 소 정리(Fermat's Little Theorem) (1/3)
 - 필요성
 - 소수와 정수의 거듭제곱 간 모듈러 연산의 규칙성을 설명
 - 거듭제곱의 계산을 줄일 수 있음
- 버전1
 - 만약 p 가 소수이고, p 는 a 를 나누지 못하는 정수일 경우
 - $a^{p-1} \equiv 1 \pmod{p}$
 - e.g., $6^{10} \pmod{11} = 1, (p = 11)$
- 버전2
 - 만약 p 가 소수이고 버전1 이외의 경우
 - $a^p \equiv a \pmod{p}$
 - e.g., $3^{12} \pmod{11} = (3^{11} \times 3^1) \pmod{11} = (3^{11} \pmod{11}) \times (3^1 \pmod{11})$
 $= (3 \times 3) \pmod{11} = 9$

소수

- 페르마의 소 정리(Fermat's Little Theorem) (2/3)

- 곱에 관한 역원

- 필요성

- 페르마의 소정리를 이용하여, 모듈로 값이 소수일 경우 확장 유클리드 알고리즘을 사용하지 않더라도 곱에 관한 역원을 구할 수 있음

- 만약 p 가 소수이고 $p \nmid a$ 라면, $a^{-1} \bmod p = a^{p-2} \bmod p$

- 증명

- $a^{-1} \bmod p = a^{p-2} \bmod p$ 양변에 a 를 곱함
 - $1 \bmod p = a^{p-1} \bmod p$
 - (페르마의 소 정리 중 버전1을 적용)

소수

- 페르마의 소 정리(Fermat's Little Theorem) (3/3)
- 예제 9.5

$8^{-1} \bmod 17$ 을 구해보자

$$8^{-1} \bmod 17 = 8^{17-2} \bmod 17 = 8^{15} \bmod 17$$

$$8^2 \bmod 17 = 64 \bmod 17 = 13$$

$$8^3 \bmod 17 = (13 \times 8) \bmod 17 = 2$$

$$8^4 \bmod 17 = (2 \times 8) \bmod 17 = 16$$

$\vdots$

$$8^{15} \bmod 17 = (8 \times 4) \bmod 17 = 15$$

- 예제 9.6

$5^{-1} \bmod 23$ 을 구해보자

$$5^{-1} \bmod 23 = 5^{23-2} \bmod 23 = 5^{21} \bmod 23 = 14 \bmod 23 = 14$$

소수

- 오일러 정리(Euler's Theorem) (1/6)

- 필요성

- 모듈로 값이 소수일 경우 뿐만 아니라 정수일 경우에도 규칙성이 있다는 것을 설명

- 버전1

- a 와 n 가 서로소인 경우

- $a^{\varphi(n)} \equiv 1 \pmod n$

- 버전2

- a 와 n 이 서로소가 아니고 k 가 정수일 경우

- $a^{k \times \varphi(n) + 1} \equiv a \pmod n$

소수

- 오일러 정리(Euler's Theorem) (2/6)

- 버전1 증명

- n 이하 자연수에서 n 과 서로소인 수들의 집합 S 에 a 를 곱한 집합은 다음과 같음

$$S = \{s_1, s_2, \dots, s_{\varphi(n)}\}, aS = \{as_1, as_2, \dots, as_{\varphi(n)}\}$$

- 집합 aS 를 n 으로 나누면 나오는 나머지가 모두 다름

a 와 n 이 서로소이므로 $as \bmod n$ 은 서로 다른 나머지를 갖는다
같은 나머지를 갖는다면 $a(s_i - s_j) \equiv 0 \pmod{n}$ 이 되는데 a 와 n 이 서로소이므로 $s_i = s_j$ 여야만 한다

소수

- 오일러 정리(Euler's Theorem) (3/6)

- 버전1 증명

- aS 의 원소들을 n 으로 나눈 나머지는 S 의 원소들과 대응됨

$\gcd(a, n) = 1$ 일 경우, a^{-1} 가 존재하므로 aS 와 S 는 일대일대응임

- 집합 S 와 집합 aS 는 원소의 값과 수가 같으므로 다음과 같음

$$aS_1 \times aS_2 \times \cdots \times aS_{\varphi(n)} \equiv S_1 \times S_2 \times \cdots \times S_{\varphi(n)} \pmod{n}$$

- 양변을 S 의 원소들로 나눔

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

소수

- 오일러 정리(Euler's Theorem) (4/6)

- 버전2 증명

- a 가 p 의 배수도 아니고 q 의 배수도 아니라면 a 와 n 은 서로소

$$a^{k \times \varphi(n) + 1} \bmod n = (a^{\varphi(n)})^k \times a \bmod n = (1)^k \times a \bmod n = a \bmod n$$

- a 가 p 의 배수가 되어 $a = i \times q$ 이지만 q 의 배수는 아닌 경우

$$a^{\varphi(n)} \bmod q = (a^{\varphi(q)} \bmod q)^{\varphi(p)} \bmod q = 1 \rightarrow a^{\varphi(n)} \bmod q = 1$$

$$a^{k \times \varphi(n)} \bmod q = (a^{\varphi(q)} \bmod q)^k \bmod q = 1 \rightarrow a^{k \times \varphi(n)} \bmod q = 1$$

$$a^{k \times \varphi(n)} \bmod q = 1, \quad a^{k \times \varphi(n)} = 1 + j \times q \text{ (합동)}$$

$$a^{k \times \varphi(n) + 1} = a \times (1 + j \times q) = a + j \times q \times a = a + (i \times q) \times q \times p = a + (i \times j) \times n$$

$$a^{k \times \varphi(n) + 1} = a + (i \times j) \times n \rightarrow a^{k \times \varphi(n) + 1} = a \bmod n \text{ (합동 관계)}$$

소수

- 오일러 정리(Euler's Theorem) (5/6)
- 버전2 증명
 - a 가 q 의 배수로서 $a = i \times q$ 이지만 p 의 배수는 아닌 경우
증명은 2에서 p 와 q 의 역할만 교체

소수

- 오일러 정리(Euler's Theorem) (6/6)

- 곱에 대한 역원

- 필요성

- 오일러 정리를 이용하면 소수 모듈로와 합성수 모듈로를 갖는 경우에 곱에 대한 역원을 가질 수 있음

- n 과 a 가 서로소일 경우

- $a^{-1} \bmod n = a^{\varphi(n)-1} \bmod n$
 - 증명

$$a \times a^{-1} \bmod n = a \times a^{\varphi(n)-1} \bmod n \rightarrow a^{\varphi(n)} \bmod n = 1 \bmod n$$

- 예제 9.7

$8^{-1} \bmod 77$ 을 구해보자

$$8^{-1} \bmod 77 = 8^{\varphi(77)-1} \bmod 77 = 8^{59} \bmod 77 = 29 \bmod 77$$

소수

- 소수 생성 (1/2)

- 소수를 생성할 수 있는 공식을 만들고자 함
- 메르센 소수(Mersenne Prime)
 - 정의
 - 메르센 수 중에서 소수인 수
 - 메르센 수 : $M_p = 2^p - 1$
 - p 가 소수라면, M_p 는 소수라고 생각되었으나 반례가 존재함

$$M_2 = 2^2 - 1 = 3$$

$$M_3 = 2^3 - 1 = 7$$

$$M_5 = 2^5 - 1 = 31$$

$$M_7 = 2^7 - 1 = 127$$

$$M_{11} = 2^{11} - 1 = 2047 - \text{반례}, 2047 = 23 \times 89$$

$$M_{13} = 2^{13} - 1 = 8191$$

$$M_{17} = 2^{17} - 1 = 131071$$

소수

- 소수 생성 (2/2)
- 페르마 소수(Fermat Prime)
 - 정의
 - 페르마 수 중에서 소수인 수
 - 페르마 수 : $F_n = 2^{2^n} + 1$
 - F_4 까지는 검증되었으나 F_5 는 소수가 아니라는 것이 증명

$$F_0 = 2^1 + 1 = 3$$

$$F_1 = 2^2 + 1 = 5$$

$$F_2 = 2^4 + 1 = 17$$

$$F_3 = 2^8 + 1 = 257$$

$$F_4 = 2^{16} + 1 = 65537$$

$$F_5 = 2^{32} + 1 = 4294967297 - \text{반례}(641 \times 6700417)$$

목 차

- 소수
- 소수 판정
- 지수와 로그

소수 판정

- 개요

- 암호학에서 키를 생성하기 위해 선택한 수가 소수인지 판별하기 위해 필요

- 알고리즘 종류

- 결정적 알고리즘(Deterministic Algorithm)

- 같은 입력에 대해 항상 같은 출력을 주고 같은 연산 과정을 거치는 알고리즘
 - 정확하고 예측 가능한 결과를 원할 때 사용

- 확률적 알고리즘(Probabilistic Algorithm)

- 같은 입력에도 실행마다 다른 결과나 연산 과정을 가질 수 있는 알고리즘
 - 오차가 존재하나 빠른 결과를 원할 때 사용

소수 판정

- 결정적 소수 검증 알고리즘 (1/6)
 - 정의
 - 정수를 입력 받아 합성수인지 소수인지 판별하는 알고리즘
- 알고리즘 종류
 - 나눗셈 알고리즘(Divisibility Algorithm)
 - AKS 알고리즘(Agrawal-Kayal-Saxena Algorithm)

소수 판정

- 결정적 소수 검증 알고리즘 (2/6)

- 나눗셈 알고리즘(Divisibility Algorithm)

- 정의

- 입력받는 수를 $\sqrt{n}$ 보다 작은 모든 양의정수로 나누어 소수를 판별하는 알고리즘
 - $\sqrt{n}$ 보다 작은 수들 중 나누어지는 수가 있으면 n 은 합성수로 판별

- 비트-연산 복잡도

- $f(n_b) = \sqrt{2^{n_b}} = 2^{\frac{n_b}{2}}$
(n_b 는 n 을 2진수로 변경하였을 때 비트 수)

- Big-O 기호 = $O(2^{n_b})$

- 특징

- n_b 가 커지면 비실용적
- 소수 판별의 가장 기초적인 방법으로 사용

```
Divisibility_Test ( $n$ ) //  $n$ 에 대해서 소수 검증
{
     $r \leftarrow 2$ 
    while ( $r < \sqrt{n}$ )
    {
        if ( $r \mid n$ ) return "a composite"
         $r \leftarrow r + 1$ 
    }
    return "a prime"
}
```

소수 판정

- 결정적 소수 검증 알고리즘 (3/6)
- 나눗셈 알고리즘(Divisibility Algorithm)
 - 예제 9.8

n 의 비트 수를 200이라고 가정했을 때 나눗셈 알고리즘을 수행하는데 필요한 비트 연산의 회수는?

- $n_b = 200, f(200) = \sqrt{2^{200}} = 2^{100}$ 번의 비트 연산 필요
- 초당 2^{30} 번의 비트 연산을 수행할 수 있는 컴퓨터라면
 2^{70} 초의 시간 필요

소수 판정

- 결정적 소수 검증 알고리즘 (4/6)
 - AKS 알고리즘(Agrawal-Kayal-Saxena Algorithm)
 - 정의
 - 정수 $n(\geq 2)$ $(x - a)^n \equiv (x^p - a) \bmod p$ 를 이용하여 p 소수인지를 판별하는 알고리즘
 - 위 식이 성립하지 않으면 p 는 합성수
 - 비트-연산 복잡도
 - $f(n_b) = (\log_2 n_b)^{12}$
 - Big O 기호 - $O((\log_2 n_b)^{12})$
 - 특징
 - 나눗셈 알고리즘에 비해 큰 입력값 계산 가능함
 - 다항적 시간 복잡도를 가짐

소수 판정

- 결정적 소수 검증 알고리즘 (5/6)
- AKS 알고리즘(Agrawal-Kayal-Saxena Algorithm)
 - 알고리즘 의사코드

```
AKS_Primality_Test(n)    // n이 소수인지 판별
{
    r ← 2
    while (ord_r(n) ≤ log(n)2)
        r ← r + 1

    for a ← 2 to min(r, n-1)
        if (1 < gcd(a, n) < n)
            return "composite"

    if (n ≤ r)
        return "prime"

    for a ← 1 to [sqrt(totient(r)) × log n]
        if ((x + a)n ≠ xn + a mod (xr - 1, n))
            return "composite"

    return "prime"
}
```

소수 판정

- 결정적 소수 검증 알고리즘 (6/6)
 - AKS 알고리즘(Agrawal-Kayal-Saxena Algorithm)
 - 예제 9.9

n_b 의 비트 수를 200이라고 가정했을 때 AKS 알고리즘을 수행하는데 필요한 비트 연산의 회수는?

- 이 알고리즘의 비트-연산 복잡도는 $O((\log_2 200)^{12})$
- 소수 판별을 위해 $(\log_2 200)^{12} = 39,547,615,483$ 의 비트 연산 필요
- 초당 10억번의 비트 연산을 수행하는 컴퓨터를 사용하는 경우 해당 알고리즘을 이용해 40초만에 소수 판별 가능

소수 판정

- 확률적 소수 검증 알고리즘 (1/15)

- 정의

- 어떤 수가 소수인지 확률적으로 판단하는 알고리즘

규칙

- a. 만약 검증하려는 정수가 소수라면, 알고리즘은 반드시 소수라고 판정
 - b. 만약 검증하려는 정수가 합성수라면, 알고리즘은 이 수를 합성수라고 판정할 확률이 $1 - \epsilon$ 이고, 소수라고 판정할 확률은 ϵ

- 알고리즘 종류

- 페르마 소수 검증(Fermat Primality Test)
 - 제곱근 소수 검증(Square Root Primality Test)
 - 밀러 라빈 소수 검증(Miller-Rabin Primality Test)

소수 판정

- 확률적 소수 검증 알고리즘 (2/15)
- 페르마 소수 검증(Fermat Primality Test)
 - 정의
 - 어떤 수 n 이 소수인지 여부를 파악하기 위해 $a^{n-1} \bmod n$ 의 값이 1이 되는지 확인하는 검증

만약 n 이 소수라면, $a^{n-1} \equiv 1 \bmod n$ 이다

만약 n 이 합성수라면, $a^{n-1} \equiv 1 \bmod n$ 일 수도 있다

- 시간 복잡도
 - $O(k^3)$ (k 는 비트 수)
- 특징
 - 소수는 페르마 검증을 하면 소수로 판정
 - 합성수가 소수라고 판정 날 확률은 ε
 - 여러 밑수($a_1, a_2, a_3 \dots$)를 사용해서 검증하게 되면 확률 값을 더 크게 개선 가능

소수 판정

- 확률적 소수 검증 알고리즘 (3/15)
- 페르마 소수 검증(Fermat Primality Test)
 - 예제 9.10

수 561을 페르마 검증 해보자

2를 밑수로 사용할 경우, $2^{561-1} \equiv 1 \pmod{561}$ 으로 페르마 검증을 통과했지만
 $561 = 33 \times 17$ 으로 소수가 아님

소수 판정

- 확률적 소수 검증 알고리즘 (4/15)

- 페르마 소수 검증

- 알고리즘 의사코드

```
Fermat_Primality_Test(n, k)  // n이 소수인지 판별
{
    if (n ≤ 1 or n = 4)
        return "composite"

    if (n ≤ 3)
        return "prime"

    for i ← 1 to k
    {
        a ← random integer in [2, n-2]

        if (a^(n-1) mod n ≠ 1)
            return "composite"
    }

    return "probably prime"
}
```

소수 판정

- 확률적 소수 검증 알고리즘 (5/15)
- 제곱근 소수 검증(Square Root Primality Test)
 - 정의
 - 소수인 n 과 서로소이면서 n 보다 작은 정수 a 에 대하여 $a^2 \equiv 1 \pmod n$ 을 만족하는 모든 a 가 오직 $a \equiv 1 \pmod n$ 혹은 $a \equiv -1 \pmod n$ 일 경우에만 n 이 소수일 가능성이 있다고 판단하는 검증임
 - 방법
 - 주어진 n 소수인 경우 1과 $n-1$ 을 제외한 n 보다 작은 수를 제공해서 $\pmod n$ 을 했을 때 어느 것도 1이 되지 않는다는 것을 확인할 수 있음
 - 특징
 - 정의를 만족하더라도 합성수인 n 이 존재할 수 있음
 - e.g., $n = 4, 14$ 일 때 정의를 만족하지만 합성수임
 - 모듈로 연산에서 -1 이 의미하는 것은 $n-1$ 임에 유의해야 함
 - e.g., $-1 \equiv 5 \pmod 6$

소수 판정

- 확률적 소수 검증 알고리즘 (6/15)
- 제곱근 소수 검증(Square Root Primality Test)
 - 예제 9.11

n 이 7이면 모듈로 n 으로 1의 제곱근을 알아보자

$$1^2 = 1 \bmod 7 = 1, (-1)^2 = 1 \bmod 7 = 1$$

$$2^2 = 4 \bmod 7 = 4, (-2)^2 = 4 \bmod 7 = 4$$

$$3^2 = 9 \bmod 7 = 2, (-3)^2 = 9 \bmod 7 = 2$$

$$4^2 = 16 \bmod 7 = 2, (-4)^2 = 16 \bmod 7 = 2$$

$$5^2 = 25 \bmod 7 = 4, (-5)^2 = 25 \bmod 7 = 4$$

$$6^2 = 36 \bmod 7 = 1, (-6)^2 = 36 \bmod 7 = 1$$

1과 $6(=n-1)$ 을 제외한 수들을 제곱해서 모듈로 연산을 했을 때 결과값이 1이 나오는 수가 존재하지 않으므로 7은 소수일 가능성이 높음

소수 판정

- 확률적 소수 검증 알고리즘 (6/15)
- 제곱근 소수 검증(Square Root Primality Test)
 - 예제 9.12

n 이 4이면 모듈로 n 으로 1의 제곱근을 알아보자

$$1^2 = 1 \bmod 4 = 1, (-1)^2 = 1 \bmod 4 = 1$$

$$3^2 = 9 \bmod 4 = 1, (-3)^2 = 9 \bmod 4 = 1$$

1과 $3(=n-1)$ 을 제외한 수들을 제곱해서 모듈로 연산을 했을 때 결과값이 1이 나오는 수가 존재하지 않으므로 4는 소수일 가능성이 높으나, 4는 합성수임

소수 판정

- 확률적 소수 검증 알고리즘 (7/15)
- 제곱근 소수 검증(Square Root Primality Test)
 - 예제 9.13

n 이 8이면 모듈로 n 으로 1의 제곱근을 알아보자

$$1^2 = 1 \bmod 8 = 1, (-1)^2 = 1 \bmod 8 = 1$$

$$3^2 = 9 \bmod 8 = 1, (-3)^2 = 9 \bmod 8 = 1$$

$$5^2 = 25 \bmod 8 = 1, (-5)^2 = 25 \bmod 8 = 1$$

$$7^2 = 49 \bmod 8 = 1, (-7)^2 = 49 \bmod 8 = 1$$

1과 $7(=n-1)$ 을 제외한 수들을 제곱해서 모듈로 연산을 했을 때 결과값이 1이 나오는 수가 존재하므로 8은 합성수임

소수 판정

- 확률적 소수 검증 알고리즘 (8/15)
- 밀러-라빈 소수 검증(Miller-Rabin Primality Test) (1/5)
 - 정의
 - 페르마 검증과 제곱근 검증을 혼합하여 *강 의사소수를 구하는 방법
 - 특징
 - $n - 1 = m \times 2^k$ (m 은 홀수)로 표현
 - 밑수 a 를 사용한 페르마 검증은 아래와 같이 표현

*강 의사소수(Strong Pseudoprime)
소수일 확률이 매우 큰 수

$$a^{n-1} = a^{m \times 2^k} = (a^m)^{2^k} = (a^m)^{\underbrace{2^{2^{\dots 2}}}_{k \text{ times}}}$$

소수 판정

- 확률적 소수 검증 알고리즘 (9/15)
- 밀러-라빈 소수 검증(Miller-Rabin Primality Test) (2/5)
 - 과정
 - 초기화 단계
 - 밑수 a 를 선택하고 $T = a^m$ 을 구함. (여기서 $m = (n - 1)/2^k$)
 - 만약 T 가 $+1$ 이거나 -1 일 경우 n 은 강 의사소수라고 선언
 - 페르마 검증과 제곱근 검증을 통과
 - 만약 T 가 그 이외의 값을 가질 경우, n 이 소수인지 합성수인지 판단 불가
 - 1단계
 - T 를 제곱
 - 만약 결과가 ± 1 이 된다면, 페르마 검증은 통과하지만 제곱근 검증은 미 통과 (T 는 1이고 앞 단계에서는 ± 1 이 아닌 다른 어떤 값이었기 때문)
 - 만약 결과가 -1 이라면, n 은 페르마 검증 통과, 강 의사소수라고 선언
 - 만약 결과가 ± 1 이 아닌 다른 값을 갖는다면, 소수판별 불가

소수 판정

- 확률적 소수 검증 알고리즘 (10/15)
- 밀러-라빈 소수 검증(Miller-Rabin Primality Test) (3/5)
 - 과정
 - 2단계에서 $k - 1$ 단계
 - 단계 1과 동일하게 반복
 - k 단계
 - 만약 해당 단계의 결과가 1이라면, 페르마 검증 통과
 - 그러나 앞 단계의 결과가 ± 1 이 아니었기 때문에 제곱근 검증 미통과
 - $k - 1$ 단계 후 절차가 끝나지 않았다면 n 은 합성수라고 선언

소수 판정

- 확률적 소수 검증 알고리즘 (11/15)
- 밀러-라빈 소수 검증(Miller-Rabin Primality Test) (4/5)
 - 알고리즘 의사코드

```
Miller_Rabin_Test( $n$ ,  $a$ ) //  $n$ 은 수,  $a$ 는 밑수
{
    Find  $m$  and  $k$  such that  $n - 1 \equiv m \times 2^k$ 
     $T \leftarrow a^m \bmod n$ 
    if( $T = \pm 1$ ) return "a prime"
    for( $i \leftarrow 1$  to  $k - 1$ ) //  $k - 1$ 은 최대 단계 수
    {
         $T \leftarrow T^2 \bmod n$ 
        if( $T = +1$ ) return "a composite"
        if( $T = -1$ ) return "a prime"
    }
    return "a composite"
}
```

소수 판정

- 확률적 소수 검증 알고리즘 (12/15)
- 밀러-라빈 소수 검증(Miller-Rabin Primality Test) (5/5)
 - 예제 9.14

수 561은 밀러-라빈 검증을 통과하는가?

밑수 a 를 2로 사용하면, $561 - 1 = 35 \times 2^4$ 로 표현되므로 $m = 35, k = 4, a = 2$

초기화 $T = 2^{35} \bmod 561 = 263 \bmod 561$

$k = 1 :$ $T = 263^2 \bmod 561 = 166 \bmod 561$

$k = 2 :$ $T = 166^2 \bmod 561 = 67 \bmod 561$

$k = 3 :$ $T = 67^2 \bmod 561 = +1 \bmod 561 \rightarrow$ 합성수

• 예제 9.15

27은 소수가 아닌 것을 알고 있다. 밀러-라빈 검증을 하시오

• 밑수로 2를 사용하면 $27 - 1 = 13 \times 2^1$ 이므로 $m = 13, k = 1, a = 2$

• 이 경우 $k - 1 = 0$ 이므로 초기화 단계만 수행

$$T = 2^{13} \bmod 27 = 11 \bmod 27$$

• 루프를 돌지 않기 때문에 합성수

소수 판정

- 확률적 소수 검증 알고리즘 (13/15)
- 나눗셈 검증과 밀러-라빈 검증을 조합한 검증 방식
 - 홀수 정수 하나 선택(2를 제외한 짝수는 모두 합성수)
 - 알려진 소수에 대해 나눗셈 검증을 하여 합성수를 가지고 검증을 하려는 건 아닌지 확인
 - 검증을 위해 밑수로 사용할 수들의 집합 결정
 - 밑수별로 밀러-라빈 검증, 한 단계라도 통과하지 못 한다면 1 단계로 돌아가서 다른 홀수 선택

소수 판정

- 확률적 소수 검증 알고리즘 (14/15)
- 추천하는 소수 검증
 - 예제 9.16

4033은 37×109 이므로 합성수이다.

이 수는 위에 언급한 추천하는 소수 검증을 통과하는가?

- 우선 나눗셈 검증 실시, 2, 3, 5, 7, 11, 17, 23은 4033의 약수가 아님
- 밑수를 2로 사용하여 밀러-라빈 검증을 실시
- $4033 - 1 = 63 \times 2^6$ 이므로 $m = 63$, $k = 6$

초기화 $T \equiv 2^{63} \pmod{4033} \equiv 3521 \pmod{4033}$

$k = 1 :$ $T \equiv T^2 \equiv 3521^2 \pmod{4033} \equiv -1 \pmod{4033} \rightarrow$ 통과

소수 판정

- 확률적 소수 검증 알고리즘 (15/15)

- 추천하는 소수 검증

- 2가 밑수일 경우에 우연히 통과했을 가능성에 대비해
3이 밑수일 경우도 계산

초기화

$$T \equiv 3^{63} \pmod{4033} \equiv 3551 \pmod{4033}$$

$$k = 1 : T \equiv T^2 \equiv 3551^2 \pmod{4033} \equiv 2443 \pmod{4033}$$

$$k = 2 : T \equiv T^2 \equiv 2443^2 \pmod{4033} \equiv 3442 \pmod{4033}$$

$$k = 3 : T \equiv T^2 \equiv 3442^2 \pmod{4033} \equiv 2443 \pmod{4033}$$

$$k = 4 : T \equiv T^2 \equiv 2443^2 \pmod{4033} \equiv 3442 \pmod{4033}$$

$$k = 5 : T \equiv T^2 \equiv 3442^2 \pmod{4033} \equiv 2443 \pmod{4033} \rightarrow \text{실패(합성수)}$$

목 차

- 소수(Primes)
- 소수 판정
- 지수와 로그

지수와 로그

- 지수(Exponentiation) (1/6)
 - 정의
 - 거듭제곱에서 밑을 곱해야 하는 수를 나타내는 숫자
 - 표현
 - $y = a^x$
 - x 를 지수, a 를 밑이라고 명명
 - 성질
 - 지수와 로그는 역 연산 관계($y = a^x \leftrightarrow x = \log_a y$)
 - $a^x \times a^y = a^{x+y}$
 - $(a^x)^y = a^{x \times y}$

지수와 로그

- 지수(Exponentiation) (2/6)
- 제곱-곱 방법(Square-and-Multiply Method)
 - 개요
 - 지수 계산을 빠르게 하기 위해 고안
 - 제곱과 곱셈 모두 사용
 - 방법
 - 지수를 n_b 비트(x_0 에서 x_{n_b-1})를 갖는 2진수로 간주
 - n_b 비트는 지수를 2진수로 나타내었을 때의 자리 수
 - 일반적으로 아래와 같이 표현

$$x = x_{n_b-1} \times 2^{k-1} + x_{n_b-2} \times 2^{k-2} + \cdots x_2 \times 2^2 + x_1 \times 2^1 + x_0 \times 2^0$$

- e.g., $x = 22 = (10110)_2$ 일 때
 $22 = 2^4 \times 1 + 2^3 \times 0 + 2^2 \times 1 + 2^1 \times 1 + 2^0 \times 0$ 으로 표현 가능

지수와 로그

- 지수(Exponentiation) (3/6)

- 고속 지수 계산

- 방법

- $y = a^x$ 에 적용

$$x = x_{n_b-1} \times 2^{k-1} + x_{n_b-2} \times 2^{k-2} + \dots x_2 \times 2^2 + x_1 \times 2^1 + x_0 \times 2^0$$

x_i 는 0 or 1

$y = a$



$$y = \boxed{a^{2^{n_b-1}} \text{ or } 1} \times \boxed{a^{2^{n_b-2}} \text{ or } 1} \times \dots \times \boxed{a^2 \text{ or } 1} \times \boxed{a^1 \text{ or } 1}$$

예시

$$y = a^9 = a^{1001_2} = a^8 \times 1 \times 1 \times a$$

지수와 로그

- 지수(Exponentiation) (4/6)
 - 고속 지수 계산
 - 알고리즘 의사코드

```
Square_and_Multiply ( $a, x, n$ )
```

```
{  
   $y \leftarrow 1$   
  for( $i \leftarrow 0$  to  $n_b - 1$ )  
  {  
    if( $x_i = 1$ )  $y \leftarrow a \times y \bmod n$   
     $a \leftarrow a^2 \bmod n$   
  }  
  return  $y$   
}
```

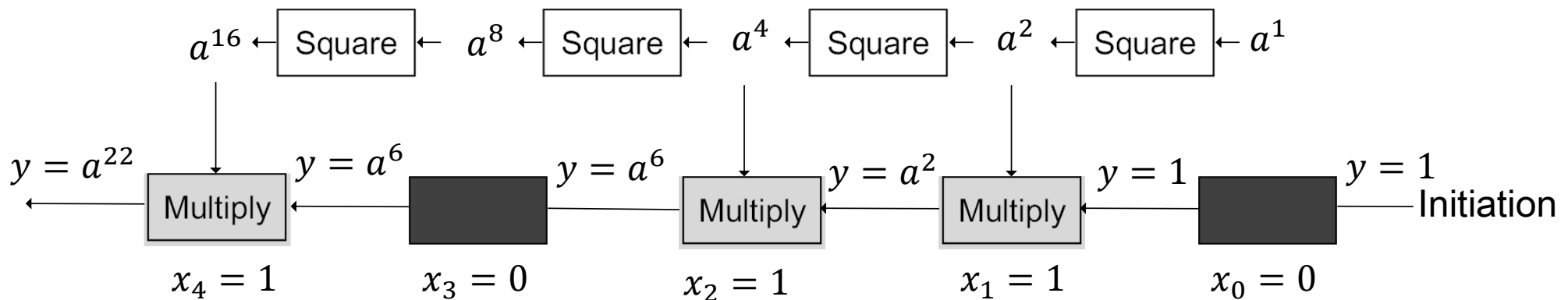
// n_b 는 x 의 비트 수이다
// 비트가 1인 경우만 곱한다
// 마지막 반복에서는 제곱이 필요 없다

지수와 로그

- 지수(Exponentiation) (5/6)

- 고속 지수 계산

- 알고리즘 의사코드를 이용하여 $y = a^{22}$ 를 구하는 과정



- $x = 22 = (10110)_2$
- $y = a^{22}$ 까지 진행되는 과정
- 검은 색으로 된 사각형이 의미하는 것은 곱셈이 생략된 곳
- 이전 단계의 y 값이 다음 단계로 그대로 넘어가는 것을 확인 가능

지수와 로그

- 지수(Exponentiation) (6/6)
 - 고속 지수 계산
 - 예제 9.17

$8^{15} \bmod 17$ 을 고속 지수 계산을 이용하여 구해보자

$$8^{15} = 8^8 \times 8^4 \times 8^2 \times 8^1$$

$$8 \bmod 17 = 8$$

$$8^2 \bmod 17 = 13$$

$$8^4 \bmod 17 = 13^2 \bmod 17 = 16$$

$$8^8 \bmod 17 = 16^2 \bmod 17 = 1$$

$$\begin{aligned} 8^{15} \bmod 17 &= (8^8 \times 8^4 \times 8^2 \times 8^1) \bmod 17 \\ &= (1 \times 16 \times 13 \times 8) \bmod 17 \\ &= (16 \times 13) \bmod 17 \times (1 \times 8) \bmod 17 \\ &= 32 \bmod 17 \\ &= 15 \end{aligned}$$

지수와 로그

- 로그(Logarithm) (1/19)

- 정의

- 곱셈을 덧셈으로 변환시켜 계산하는 표현 방식

- 표현

- $x = \log_a y$
- $a > 0, a \neq 1, y > 0$

- 성질

- 로그와 지수는 역 연산 관계($x = \log_a y \leftrightarrow y = a^x$)
- $\log_a 1 = 0, \log_a a = 1$
- $\log_a MN = \log_a M + \log_a N$
- $\log_a \frac{M}{N} = \log_a M - \log_a N$
- $\log_a L^k = k \log_a L$

지수와 로그

- 로그(Logarithm) (2/19)

- 지수의 역 연산 알고리즘

- $y = a^x \bmod n \leftrightarrow x = \log_a y \bmod n$
 - $y = a^x \bmod n$ 을 연속적으로 계산하여 주어진 y 값을 찾을 때까지 수행하는 알고리즘
 - 비트 연산 복잡도가 $O(2^{n_b})$ 로 비효율적

```
Modular_Logarithm( $a, y, n$ )
{
  for( $x = 1$  to  $n - 1$ )                //  $k$ 는  $x$ 의 비트 수
  {
    if( $y \equiv a^x \bmod n$ ) return  $x$ 
  }
  return failure
}
```

지수와 로그

- 로그(Logarithm) (3/19)
 - 곱셈군에 대한 성질 (1/12)
 - 이산 대수에 대한 개념을 사용하기 위해 필요
 - 이산 대수
 - 정수와 소수에 대해 모듈러 지수 연산을 수행하는 것

일반 수학에서 로그와 지수

$$y = a^x \leftrightarrow x = \log_a y$$

이산 대수와 이산 로그

$$a^x \equiv b \pmod{p} \leftrightarrow \log_a b \pmod{p} = x$$

지수와 로그

- 로그(Logarithm) (4/19)
- 곱셈군에 대한 성질 (2/12)
 - 유한 곱셈군(Multiplicative Finite Group)
 - 정의
 - 군 $G = \langle Z_n^*, \times \rangle$ 에서 Z_n^* 은 n 과 서로소가 되는 1부터 $n - 1$ 안의 정수들을 원소로 갖는 집합, 항등원 e 는 1
 - 군의 크기
 - 정의
 - 유한군 G 의 크기는 그 군에 속하는 원소의 개수
 - $|G|$ 로 표기
 - 예제 9.18

군 $G = \langle Z_{21}^*, \times \rangle$ 의 크기를 구하여라

$|G| = \varphi(21) = \varphi(3) \times \varphi(7) = 2 \times 6 = 12$
군의 원소는 $\{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$

지수와 로그

- 로그(Logarithm) (5/19)

- 곱셈군에 대한 성질 (3/12)

- 원소의 위수*

- 정의

- a 를 몇 번 곱해야 항등원을 얻는지 나타내는 가장 작은 양의 정수

- $G = \langle Z_n^*, \times \rangle$ 에서 원소 a 의 위수는 $a^i \equiv e \pmod{n}$ 이 되는 정수 중에서 가장 작은 정수 i

- a 는 곱셈군의 원소, 항등원 e 는 1

*위수(Order)

군에서 $a^n = e$ 를 만족하는 가장 작은 정수 n 을 원소 a 의 위수라고 한다

$ord(a)$ 로 표기

지수와 로그

- 로그(Logarithm) (6/19)
- 곱셈군에 대한 성질 (4/12)
 - 원소의 위수
 - 예제 9.19

군 $G = \langle Z_{10}^*, \times \rangle$ 의 모든 원소에 대한 위수를 계산하시오

- 이 군의 원소는 $\varphi(10) = \{1, 3, 7, 9\}$, $|G| = 4$
- 각 원소의 위수를 구하기 위해서는 시행착오 방법을 사용
- 라그랑지 정리를 이용할 경우 군에 속하는 원소의 위수는 군의 크기(4)를 나누어야 함을 상기
- 4의 약수는 1, 2, 4로 원소의 위수를 계산하기 위해 3가지 수에 대해서만 거듭 제곱수를 계산

- a. $1^1 \equiv 1 \pmod{10}$; $\rightarrow \text{ord}(1) = 1$
- b. $3^1 \equiv 3 \pmod{10}$; $3^2 \equiv 9 \pmod{10}$; $3^4 \equiv 1 \pmod{10} \rightarrow \text{ord}(3) = 4$
- c. $7^1 \equiv 7 \pmod{10}$; $7^2 \equiv 9 \pmod{10}$; $7^4 \equiv 1 \pmod{10} \rightarrow \text{ord}(7) = 4$
- d. $9^1 \equiv 9 \pmod{10}$; $9^2 \equiv 1 \pmod{10}$; $\rightarrow \text{ord}(9) = 2$

지수와 로그

- 로그(Logarithm) (7/19)

- 곱셈군에 대한 성질 (5/12)

- 오일러 정리

- 정의

- 만약 a 가 군 $G = \langle Z_{10}^*, \times \rangle$ 의 원소라면 $a^{\varphi(n)} \equiv 1 \pmod n$

- $i = \varphi(n)$ 일 경우, $a^i \equiv 1 \pmod n$ 이 성립

- 예제 9.20

표에 군 $G = \langle Z_8^*, \times \rangle$ 에 대한 $a^i \equiv 1 \pmod 8$ 의 값을 나타냄
 $\varphi(8) = 4$ 이고 원소는 1, 3, 5, 7이라는 점에 유의

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$
$a = 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$
$a = 3$	$x: 3$	$x: 1$	$x: 3$	$x: 1$	$x: 3$	$x: 1$	$x: 3$
$a = 5$	$x: 5$	$x: 1$	$x: 5$	$x: 1$	$x: 5$	$x: 1$	$x: 5$
$a = 7$	$x: 7$	$x: 1$	$x: 7$	$x: 1$	$x: 7$	$x: 1$	$x: 7$

지수와 로그

- 로그(Logarithm) (8/19)
- 곱셈군에 대한 성질 (6/12)
 - 오일러 정리
 - 예제 9.21

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$
$a = 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$
$a = 3$	$x: 3$	$x: 1$	$x: 3$	$x: 1$	$x: 3$	$x: 1$	$x: 3$
$a = 5$	$x: 5$	$x: 1$	$x: 5$	$x: 1$	$x: 5$	$x: 1$	$x: 5$
$a = 7$	$x: 7$	$x: 1$	$x: 7$	$x: 1$	$x: 7$	$x: 1$	$x: 7$

- 음영으로 처리된 부분은 오일러 정리를 적용한 것
- $i = \varphi(8) = 4$ 이고 모든 a 에 대해서 그 결과는 $x = 1$
- 여러 개의 i 값에 대해 x 값은 1
- 최초로 x 값이 1이 되었을 때 i 값이 그 원소의 위수를 표현
- 각 원소의 위수는 $\text{ord}(1) = 1, \text{ord}(3) = 2, \text{ord}(5) = 2, \text{ord}(7) = 2$

지수와 로그

- 로그(Logarithm) (9/19)
- 곱셈군에 대한 성질 (7/12)
 - 원시근(Primitive Root)
 - 정의
 - 군 $G = \langle Z_n^*, \times \rangle$ 에서 한 원소의 위수가 $\varphi(n)$ 과 동일한 수
 - 특징
 - 군 $G = \langle Z_n^*, \times \rangle$ 는 오직 n 이 $2, 4, p^t, 2p^t$ 일 때만 원시근을 가짐
(p 는 2가 아닌 소수, t 는 양의 정수)
 - 한 군이 한 개의 원시근을 가지는 경우, 그 군에는 또 다른 여러 개의 원시근이 존재
 - $G = \langle Z_n^*, \times \rangle$ 가 원시근을 가지는 경우 원시근의 개수는 $\varphi(\varphi(n))$ 개

지수와 로그

- 로그(Logarithm) (10/19)
- 곱셈군에 대한 성질 (8/12)
 - 원시근(Primitive Root)
 - 예제 9.22

군 $G = \langle \mathbb{Z}_8^*, \times \rangle$ 의 원시근을 구하시오

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$
$a = 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$
$a = 3$	$x: 3$	$x: 1$	$x: 3$	$x: 1$	$x: 3$	$x: 1$	$x: 3$
$a = 5$	$x: 5$	$x: 1$	$x: 5$	$x: 1$	$x: 5$	$x: 1$	$x: 5$
$a = 7$	$x: 7$	$x: 1$	$x: 7$	$x: 1$	$x: 7$	$x: 1$	$x: 7$

어떤 원소도 위수가 $\varphi(8) = 4$ 가 되지 않고 위수가 모두 4보다 작으므로
군 $G = \langle \mathbb{Z}_8^*, \times \rangle$ 은 원시근을 가지지 않음

지수와 로그

- 로그(Logarithm) (11/19)
- 곱셈군에 대한 성질 (9/12)
 - 원시근(Primitive Root)
 - 예제 9.23

아래 표는 군 $G = \langle \mathbb{Z}_7^*, \times \rangle$ 에 대한 $a^i \equiv x \pmod{7}$ 의 계산 값이다
군 $G = \langle \mathbb{Z}_7^*, \times \rangle$ 의 원시근을 구하시오

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$
$a = 1$	x: 1	$x: 1$	$x: 1$	$x: 1$	$x: 1$	$x: 1$
$a = 2$	$x: 2$	$x: 4$	x: 1	$x: 2$	$x: 4$	$x: 1$
$a = 3$	$x: 3$	$x: 2$	$x: 6$	$x: 4$	$x: 5$	x: 1
$a = 4$	$x: 4$	$x: 2$	x: 1	$x: 4$	$x: 2$	$x: 1$
$a = 5$	$x: 5$	$x: 4$	$x: 6$	$x: 2$	$x: 3$	x: 1
$a = 6$	$x: 6$	x: 1	$x: 6$	$x: 1$	$x: 6$	$x: 1$

군의 크기는 $\varphi(7) = 6$

원소의 위수는

$$\begin{array}{lll} \text{ord}(1) = 1 & \text{ord}(2) = 3 & \text{ord}(3) = 6 \\ \text{ord}(4) = 3 & \text{ord}(5) = 6 & \text{ord}(6) = 1 \end{array}$$

오직 두 원소 3과 5만이 $i = \varphi(n) = 6$ 에서
위수 6을 가짐

이 군의 2개의 원시근 3과 5를 가짐

지수와 로그

- 로그(Logarithm) (12/19)
- 곱셈군에 대한 성질 (10/12)
 - 원시근(Primitive Root)
 - 예제 9.24

$n = 17, 20, 38, 50$ 중 어떤 n 에 대해서 군 $G = \langle Z_n^*, \times \rangle$ 이 원시근을 갖는가?

$G = \langle Z_{17}^*, \times \rangle$ 은 원시근을 가짐
 $17 = 17^1$ 이고 17은 소수이기 때문임($p = 17, t = 1$)

$G = \langle Z_{20}^*, \times \rangle$ 은 원시근을 가지지 않음
 $20 = 2^2 \times 5^1$ 이기 때문임

$G = \langle Z_{38}^*, \times \rangle$ 은 원시근을 가짐
 $38 = 2^1 \times 19^1$ 이고 19는 소수이기 때문임($p = 19, t = 1$)

$G = \langle Z_{50}^*, \times \rangle$ 은 원시근을 가짐
 $50 = 2^1 \times 5^2$ 이고 5는 소수이기 때문임($p = 5, t = 2$)

지수와 로그

- 로그(Logarithm) (13/19)
- 곱셈군에 대한 성질 (11/12)
 - 순환 군(Cyclic Group)
 - 정의
 - 한 원소 g 를 반복해서 군 연산으로 적용해 생성할 수 있는 군
 - 만약 군 $G = \langle Z_n^*, \times \rangle$ 이 원시군을 갖는다면 이 군은 순환 군
 - p 가 소수라면 군 $G = \langle Z_p^*, \times \rangle$ 은 항상 순환 군
 - 만약 g 가 이 군의 원시군이라면, 군 전체 집합 Z_n^* 을 다음과 같이 생성

$$Z_n^* = \{g^1, g^2, g^3, \dots, g^{\varphi(n)}\}$$

지수와 로그

- 로그(Logarithm) (14/19)
- 곱셈군에 대한 성질 (12/12)
 - 순환 군(Cyclic Group)
 - 예제 9.25

군 $G = \langle Z_{10}^*, \times \rangle$ 는 $\varphi(10) = 4$ 이고 $\varphi(\varphi(10)) = 2$ 이므로 두 개의 원시군을 갖는다.
이 두 개의 원시군 3과 7이 각각 어떻게 전체 군의 집합을 생성해내는지 알아보자

$$g = 3 \rightarrow g^1 \bmod 10 = 3, g^2 \bmod 10 = 9, g^3 \bmod 10 = 7, g^4 \bmod 10 = 1$$

$$g = 7 \rightarrow g^1 \bmod 10 = 7, g^2 \bmod 10 = 9, g^3 \bmod 10 = 3, g^4 \bmod 10 = 1$$

3과 7은 위수가 4이므로 군 전체를 생성, Z_{10}^* 의 모든 원소인 $\{1, 3, 7, 9\}$ 를 생성한다
 $\therefore Z_{10}^*$ 은 순환군이다

지수와 로그

- 로그(Logarithm) (15/19)

- 이산 대수의 아이디어

- 이산 대수의 성질

- $a^x \bmod p$ 의 연산에 비해 역 역산인 $\log_a b \bmod p$ 은 계산이 복잡
 - 지수 법칙이 모듈러에서도 성립
 - $a^x \times a^y \equiv a^{x+y} \bmod p$
 - 이산 로그의 해는 위수만큼 주기성을 가지고 반복
 - Z_p^* 는 유한한 곱셈 군이고 생성원 a 가 존재하면 $\{a^0, a^1, \dots, a^{p-2}\}$ 는 모든 원소를 순환적으로 생성
 - $a^k \equiv a^{k \bmod (p-1)} \bmod p$

상용 대수 (Traditional Logarithm)	이산 대수 (Discrete Logarithm)
$\log_a 1 = 0$	$L_g 1 \equiv 0 \pmod{\varphi(n)}$
$\log_a (x \times y) = \log_a x + \log_a y$	$L_g (x \times y) \equiv (L_g x + L_g y) \pmod{\varphi(n)}$
$\log_a x^k = k \times \log_a x$	$L_g x^k \equiv k \times L_g x \pmod{\varphi(n)}$

지수와 로그

- 로그(Logarithm) (16/19)

- 이산 대수의 아이디어

- 군 $G = \langle \mathbb{Z}_p^*, \times \rangle$ 의 성질(p 는 소수)

1. 이 군의 원소는 1부터 $p - 1$ 까지의 모든 수
2. 이 군은 항상 원시근을 보유
3. 이 군은 순환 군, 원소들은 g^x 를 사용하여 생성 가능
(x 는 1부터 $\phi(n) = p - 1$ 안의 정수)
4. 원시근은 로그의 밑수로 생각 가능
 - 이 군이 k 개의 원소를 가진다고 하면 k 개의 서로 다른 밑수로 계산 가능
 - 집합 한 원소 y 에 대해 $x = \log_g y$ 일 경우, g 를 밑수로 갖는 y 의 로그인 새로운 원소 x 가 존재

지수와 로그

- 로그(Logarithm) (17/19)
- 이산 대수를 이용한 모듈로 로그의 해 탐색
 - y 가 주어졌을 때 $y = a^x \pmod n$ 에서 x 를 구하는 법
 - 이산 대수의 도표화
 - 서로 다른 밑수별로 Z_p^* 에 대한 표 사용
 - 군 $G = \langle Z_7^*, \times \rangle$ 에 대한 이산 대수

y	1	2	3	4	5	6
$x = L_3 y$	6	2	1	4	5	3
$x = L_5 y$	6	4	5	2	1	3

- 모든 군과 모든 가능한 밑수별로 다른 이산 대수표가 주어지는 경우, 어떤 대수 문제라도 해결 가능하나 현실적으로 불가능
 - 전통적인 로그를 사용하는 것과 동일한 방법
 - 계산기나 컴퓨터가 발명되기 이전엔 밑수가 10인 상용 대수표를 가지고 로그 값을 계산

지수와 로그

- 로그(Logarithm) (18/19)
- 이산 대수를 이용한 모듈로 로그의 해 탐색

y	1	2	3	4	5	6
$x = L_3 y$	6	2	1	4	5	3
$x = L_5 y$	6	4	5	2	1	3

- 예제 9.26

다음 경우에 대해 각각 x 를 구하시오

a. $4 \equiv 3^x \pmod{7}$ b. $6 \equiv 5^x \pmod{7}$

위의 표를 이용하여

$$4 \equiv 3^x \pmod{7} \rightarrow x = L_3 4 \pmod{7} = 4 \pmod{7}$$

$$6 \equiv 5^x \pmod{7} \rightarrow x = L_5 6 \pmod{7} = 3 \pmod{7} \text{임을 확인}$$

지수와 로그

- 로그(Logarithm) (19/19)
 - 이산 대수를 이용한 모듈로 로그의 해 탐색
 - 이산 대수를 기초로 한 알고리즘 사용
 - n 이 매우 커질 경우 이산 대수 성질과 이산 대수표를 사용해 $y = a^x \pmod n$ 을 푸는 것은 불가능
 - 이 경우 대수 문제를 풀기 위해 기본적인 성질을 활용한 여러 가지 알고리즘들이 개발
 - e.g., 디피-헬만 키 교환(Diffie–Hellman key exchange)
 - 전부 찾기 알고리즘보다는 효율성이 높으나 모두 소인수분해 복잡도 정도의 수준 유지

Thanks!

민 윤 홍(muh4316@gmail.com)