제 5장 Introduction to Number Theory

5.1 Divisors

5.2 Representations of Integers and Integer Algorithms

5.3 The Euclidean Algorithm

5.4 The RSA Public Key Cryptosystem

5.1 Divisors

- 정의 5.1.1 n과 d가 정수이고 $d \neq 0$ 일 때, $n = dq$ 를 만족시키는 정수 q가 존재하면 'd가 n을 나눈다divide'라고 정의한다. 여기서 q 를 몫quotient, d를 n 의 약수divisor 또는 인수factor 라고 한다. d가 n을 나누면 $d \mid n$ 로 표기. 아니면, $d \nmid n$로 표현.
- n과 d가 양의 정수이고 $d \mid n$ 이면, $d \leq n$ 이다.
  - ∵ $d \mid n$이면, $n = dq$를 만족시키는 정수 q가 존재한다. n과 d가 양의 정수이므로, $1 \leq q$ 이다. 따라서 $d \leq dq = n$
- $d > 0$인 정수 d가 정수 n을 나눌 수 있든 없든 몫-나머지 정리 (정 리 2.5.6)에 따라 유일한 몫 q와 나머지 r을 얻을 수 있다. 즉 $n = dq + r,\ 0 \leq r < d$를 만족시키는 유일한 정수 q(몫)와 r(나머지)이 존재한다.
- 나머지 r이 0이 될 필요충분조건은 d가 n을 나눈다.

5.1 Divisors

- 정리 5.1.3 m, n, d가 정수일 때

(a) If $d \mid m$ and $d \mid n$, then $d \mid (m + n)$.

(b) If $d \mid m$ and $d \mid n$, then $d \mid (m - n)$.

(c) If $d \mid m$, then $d \mid mn$.

- 증명 (a) $d \mid m$이고 $d \mid n$이라고 가정하면, 정의 5.1.1 에 따라 어떤 정수 q_1에 대해

$$m = dq_1$$

이며, 어떤 정수 q_2에 대해

$$n = dq_2$$

이다. 이 식들을 더하면

$$m + n = dq_1 + dq_2 = d(q_1 + q_2)$$

이다. 그러므로 d는 $m + n$을 나눈다. (몫은 $q_1 + q_2$이다)

5.1 Divisors(소수, 합성수)

- 정의 5.1.4 1 보다 큰 정수가 1 과 자신만을 양의 약수로 가진다면, 이 정수를 소수prime라고 한다. 1 보다 큰 소수가 아닌 정수를 합성수composite라고 한다.
- 정리 5.1.7 1 보다 큰 양의 정수 n이 합성수일 필요충분조건은 n이 $2 \leq d \leq \sqrt{n}$인 d를 약수로 가진다.
- 증명 n이 합성수라고 가정하자.

  정의 5.1.4에 의해 합성수는

  n은 1과 자신이 아닌 다른 양의 정수 d'를 약수로 가진다.

  즉, $2 \leq d' < n$

5.1 Divisors

- 두 경우를 살펴보자.
- 만약 $d' \leq \sqrt{n}$ 이면, n은 $2 \leq d \leq \sqrt{n}$ 인 $d(d = d'$ 으로 보면)를 약수로 가진다.
- 또 다른 경우로 $d' > \sqrt{n}$ 인 경우를 살펴보면, d' 이 n을 나누므로 정의 5.1.1 에 의해서 $n = d'q$인 정수 q가 존재한다.
  따라서 q 역시 n의 약수이다.
  ($q \leq \sqrt{n}$임을 모순에 의한 증명을 하자)
  $q > \sqrt{n}$를 가정하면 $d' > \sqrt{n}$와 $q > \sqrt{n}$의 곱은
  $$n = d'q > \sqrt{n}\sqrt{n} = n$$
  이 되므로 모순이 발생한다. 따라서 $q \leq \sqrt{n}$ 이다.
  그러므로 $2 \leq d \leq \sqrt{n}$ 인 $d(d = q$로 보면)를 약수로 가진다.

5.1 Divisors(소수 검사 알고리즘)

- Algorithm 5.1.8

  1 보다 큰 정수 n이 소수인지를 결정한다.
  n이 소수이면 알고리즘은 0을 반환하고,
  n이 합성수이면 $2 \le d \le \sqrt{n}$ 이고 소수인 약수 d를 반환한다.

  Input: n Output: d

```
is_prime(n) {
  for d = 2 to ⌊√n⌋
    if (n mod d == 0)
      return d
  return 0
}
```

The worst-case time is $\Theta(\sqrt{n})$

- 이 알고리즘이 반환하는 n의 약수를 합성수 a 라 하면, a는 a보다 작은 a'을 약수로 가진다. a'은 n을 나눌 수 있고 $a' < a$이으므로 알고리즘은 $d = a'$ 가 되었을 때 a'을 반환할 것이다. 모순이 발생

5.1 Divisors(산술의 기본정리 또는 인수분해 유일성 정리)

- 예제 5.1.10 알고리즘 5.1.8 의 입력으로 $n = 1274$가 주어지면,
  이 알고리즘은 소수 2를 반환한다. $\because\ 2 \mid 1274$
  $637 = 1274/2$을 입력하면, 7를 반환.
  $91 = 637/7$ 을 입력하면, 7를 반환
  $13 = 91/7$ 을 입력하면, 0를 반환. $\because$ 13은 소수
- 1274를 소수의 곱으로 나타낼 수 있다
  $1274 = 2 \cdot 637 = 2 \cdot 7 \cdot 91 = 2 \cdot 7 \cdot 7 \cdot 13.$
- 정리 5.1.11 산술의 기본 정리 또는 인수분해 유일성 정리
  1 보다 큰 정수는 소수의 곱으로 나타낼 수 있다.
  소수들을 비감소 순서로 나열한다면, 그 인수분해는 유일하다.

5.1 Divisors

정리 5.1.12 소수는 무한히 많다

증명 "어떤 수 p가 소수이면 p보다 큰 소수가 있다".

- $p_1, p_2, \ldots, p_n$을 p 이하인 서로 다른 모든 소수라고 하자.
  다음과 같은 정수를 생각해 보자.

$$m = p_1 p_2 \cdots\ p_n + 1.$$

  어떤 j에 대해 $p_j \mid m$라 가정하자.

  $p_j \mid m$ 이므로 $m = p_j q + 1$ 이고 $q = p_1 p_2 \cdots p_{j-1} p_{j+1} \cdots p_n$

  따라서 p_j는 m을 나누지 않는다

  p'을 m의 소인수(소수인 인수)라고 하자.
  그러면 p'은 어떤 p_j 와도 같지 않다. 그런데 $p_1, p_2, \ldots, p_n$은 p이하인 모든 소수의 리스트 이므로 $p' > p$ 이어야 한다.

5.1 Divisors(공약수, 최대 공약수, 소인수분해)

- 정의 5.1.14 m과 n이 둘 중 하나는 0이 아닌 정수라고 할 때, m과 n의 공약수common divisor는 m과 n를 나누는 정수. m과 n의 최대 공약수는 m과 n의 가장 큰 공약수이고, $\gcd(m, n)$로 표기.

- 예제 5.1.16 30과 105의 최대 공약수는 소인수분해를 살펴봄으로써 구할 수 있다.
  $30 = 2 \cdot 3 \cdot 5 = 2 \cdot 3 \cdot 5 \cdot 7^0 \qquad 105 = 3 \cdot 5 \cdot 7 = 2^0 \cdot 3 \cdot 5 \cdot 7$
  여기서 3은 소인수분해에서 모두 나타나므로 공약수.
  마찬가지로 5 역시 공약수이다.
  또한 $3 \cdot 5 = 15$ 역시 공약수이다.
  이 소인수분해에서 더 이상 큰 공통의 소수 곱 형태가 없으므로 15가 30과 105의 최대 공약수가 된다.

5.1 Divisors(공약수, 최대 공약수, 소인수분해)

정리 5.1.17 정수 $m > 1$과 $n > 1$에 대한 소인수분해는

$$m = p_1^{a_1}\, p_2^{a_2} \ \cdots \ p_k^{a_k}, \qquad n = p_1^{b_1}\, p_2^{b_2} \ \cdots \ p_k^{b_k}.$$

이라고 하자. 그러면 m과 n의 최대 공약수:

$$\gcd(m,n) = p_1^{\min(a_1,b_1)} p_2^{\min(a_2,b_2)} \cdots p_k^{\min(a_k,b_k)}$$

- 증명 $g = \gcd(m,n)$이라고 하자.

g의 소인수분해에 소수 p가 포함되었다고 하면 p는 $p_1, \ldots, p_k$ 중의 하나이어야 만 한다. 그렇지 않다면 g는 m 또는 n을 나누지 못한다. 그러므로 $c_1, \ldots, c_k$에 대하여

$$g = p_1^{c_1} \cdots p_k^{c_k}$$

이다. 따라서 $p_1^{\min(a_1,b_1)} p_2^{\min(a_2,b_2)} \cdots p_k^{\min(a_k,b_k)} \qquad (1.6)$

는 m과 n을 모두 나눈다. 그러나 위의 지수 $\min(a_i, b_i)$ 중의 하나라도 증가하면, 이 수는 m 또는 n을 나누지 못한다. 따라서 (1.6)은 m과 n의 최대 공약수이다.

5.1 Divisors(공배수, 최소 공배수)

- 정의 5.1.19 m과 n을 양의 정수라고 하자. m과 n으로 모두 나눌 수 있는 정수를 m과 n의 공배수common multiple라고 한다. 최소 공배수least common multiple는 m과 n의 공배수 중 가장 작은 양수인데, $\mathrm{lcm}(m, n)$로 표기한다.

5.1 Divisors(공배수, 최소 공배수)

정리 5.1.22 $m > 1$과 $n > 1$을 정수라고 하고, 소인수분해는

$$m = p_1^{a_1}\, p_2^{a_2} \;\cdots\; p_k^{a_k},\quad n = p_1^{b_1}\, p_2^{b_2} \;\cdots\; p_k^{b_k}.$$

이라고 하자. 그러면 최소공배수는 다음과 같다.

$$\operatorname{lcm}(m,n) = p_1^{\max(a_1,b_1)} p_2^{\max(a_2,b_2)} \cdots p_k^{\max(a_k,b_k)}$$

증명 $l = \operatorname{lcm}(m,n)$라 하자. l의 소인수분해에 어떤 소수 p가 포함되었다고 하면 p는 $p_1, \ldots, p_k$ 중의 하나이어야 만 한다. 그렇지 않다면, p를 소거하고도 m과 n에 의해서 모두 나누어지는 더 작은 정수를 구할 수 있다. 그러므로 $c_1, \ldots, c_k$에 대하여

$$l = p_1^{c_1} \cdots p_k^{c_k}$$

이다. 따라서

$$p_1^{\max(a_1,b_1)} p_2^{\max(a_2,b_2)} \cdots p_k^{\max(a_k,b_k)} \qquad (1.7)$$

는 m과 n에 의해서 나누어진다. 그러나 위의 지수 $\max(a_i, b_i)$ 중의 어느 하나라도 감소한다면, 이 수는 m 또는 n 에 의해서 나누어지지 않을 것이다. 그러므로 (1.7)는 m과 n의 최소공배수이다.

5.1 Divisors(공배수, 최소 공배수)

- 정리 5.1.25 임의의 정수 $m > 1$과 $n > 1$에 대해서,

$$\gcd(m, n) \cdot \mathrm{lcm}(m, n) = mn$$

- 증명

$$\begin{aligned}
&\gcd(m, n) \cdot \mathrm{lcm}(m, n) \\
&\quad = p_1^{\min(a_1,b_1)} \cdots p_k^{\min(a_k,b_k)} p_1^{\max(a_1,b_1)} \cdots p_k^{\max(a_k,b_k)} \\
&\quad = p_1^{\min(a_1,b_1)+\max(a_1,b_1)} \cdots p_k^{\min(a_k,b_k)+\max(a_k,b_k)} \\
&\quad = p_1^{a_1+b_1} \cdots p_k^{a_k+b_k} \\
&\quad = (p_1^{a_1} \cdots p_k^{a_k})(p_1^{b_1} \cdots p_k^{b_k}) = mn
\end{aligned}$$

$\min(x, y) + \max(x, y) = x + y$ for all x and y because one of $\{\min(x, y), \max(x, y)\}$ equals x and the other equals y.

5.2 Rep. of Int. and Int. Algorithms

- 십진법에서는 10개의 기호 0, 1, 2, 3, 4, 5, 6, 7, 8, 9를 사용하며, 정수를 표현할 때 기호의 위치는 중요하다. 1의 자리 등.

$$3854 = 3 \cdot 10^3 + 8 \cdot 10^2 + 5 \cdot 10^1 + 4 \cdot 10^0$$

- 수를 나타내는 진법이 근거로 하는 수(십진법에서는 10)를 그 진법의 기수base라고 한다.
- 비트bit는 binary digit이다, 즉 0 또는 1 이다.
- 이진법binary number system (기수가 2)에서 정수를 표현하기 위해서는 2개의 기호 0과 1 만이 필요하다.

$$101101_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

- 십육진법hexadecimal number system: 0~9, A~F
- 팔진법octal number system: 0~7

5.2 Rep. of Int. and Int. Algorithms(컴퓨터의 정수 표현)

- 예 5.2.1 양의 정수 n을 표현하는 데 필요한 비트의 수는 $\lfloor 1 + \lg n \rfloor$
- 양의 정수 n를 k 비트로 표현할 수 있다고 가정하자:

$$n = 1 \cdot 2^{k-1} + b_{k-2}2^{k-2} + \cdots + b_0 2^0.$$

$2^{k-1} \le n$ 이고

$$\begin{aligned} n &= 1 \cdot 2^{k-1} + b_{k-2}2^{k-2} + \cdots + b_0 2^0 \\ &\le 1 \cdot 2^{k-1} + 1 \cdot 2^{k-2} + \cdots + 1 \cdot 2^0 = 2^k - 1 < 2^k \end{aligned}$$

이다. 따라서

$$\begin{aligned} &2^{k-1} \le n < 2^k \\ &k - 1 \le \lg n < k \qquad (\lg \text{ 취한다}) \\ &k \le 1 + \lg n < k + 1 \quad (1\text{을 더한다}) \end{aligned}$$

따라서, $k = \lfloor 1 + \lg n \rfloor$

$$\boxed{\lg n = \log_2 n}$$

5.2 Rep. of Int. and Int. Algorithms

- Ex 5.2.2 Convert binary number to decimal number

  $101101_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 45_{10}$

- Algorithm 5.2.3 기수 b 인 정수 $c_n c_{n-1} \cdots c_1 c_0$의 십진수로 변환
- Input: c, n, b

  Output: dec_val

  $base\_b\_to\_dec(c, n, b)$ {

  $\quad dec\_val = 0$

  $\quad b\_to\_the\_i = 1$

  $\quad$ for $i = 0$ to n {

  $\qquad dec\_val = dec\_val + c_i * b\_to\_the\_i$

  $\qquad b\_to\_the\_i = b\_to\_the\_i * b$

  $\quad$ }

  $\quad$ return dec_val

  }

5.2 Rep. of Int. and Int. Algorithms

- 예 5.2.5 Convert the hexadecimal number to decimal

  $B4F_{16} = 11 \cdot 16^2 + 4 \cdot 16^1 + 15 \cdot 16^0 = 2895_{10}$

- Converting a decimal number to base b

  Let's convert the decimal number 91 to binary.

  $91 = 2 \cdot 45+1$ (1), $45 = 2 \cdot 22+1$ (2). Substituting (2) for 45 into (1)

  $91 = 2 \cdot (2 \cdot 22+1) +1 = 2^2 \cdot 22+2+1$ (3), $22 = 2 \cdot 11$ (4), $11 = 2 \cdot 5 + 1$ (5).

  $5 = 2 \cdot 2 + 1$ (6). From (3), (4), (5), and (6),

  $91 = 2^2 \cdot 2 \cdot 11+2+1 = 2^3 \cdot 11+2+1 = 2^3 \cdot (2 \cdot 5 + 1)+2+1$

  $= 2^4 \cdot 5 + 2^3+2+1 = 2^4 \cdot (2 \cdot 2 + 1) + 2^3+2+1$

  $= 2^6 + 2^4 + 2^3+ 2^1 + 2^0 = 1011011_2$

5.2 Rep. of Int. and Int. Algorithms

- 예 5.2.6 십진수 130을 이진수로 표현하라.
- 풀이) 2로 나눈 나머지를 역순으로

2)<u>130</u>	나머지 = 0	1's bit
2)<u>65</u>	나머지 = 1	2's bit
2)<u>32</u>	나머지 = 0	4's bit
2)<u>16</u>	나머지 = 0	8's bit
2)<u>8</u>	나머지 = 0	16's bit
2)<u>4</u>	나머지 = 0	32's bit
2)<u>2</u>	나머지 = 0	64's bit
2)<u>1</u>	나머지 = 1	128's bit
0		

몫이 0 이 되었을 때, 나누는 작업을 중단한다

$$130_{10} = 10000010_2.$$

5.2 Rep. of Int. and Int. Algorithms

- Algorithm 5.2.7 양의 정수 m을 기수 b 인 정수 $c_n c_{n-1} \cdots c_1 c_0$로 변환.
- Input: m, b

  Output: c, n

  $dec\_to\_base\_b\_(m, b, c, n)$ {

  $n = -1$

  while $(m > 0)$ {

  $n = n + 1$

  $c_n = m \bmod b$

  $m = \lfloor m/b \rfloor$

  }

  }

5.2 Rep. of Int. and Int. Algorithms

- 예 5.2.9 Convert the decimal number 20385 to hexadecimal.
- Sol) 16로 나눈 나머지를 역순으로

16)20385　　나머지 = 1　1's place

16)1274　　나머지 = 10 = A　16's place

16)79　　나머지 = 15 = F　16^2's place

16)4　　나머지 = 4　16^3's place

0

몫이 0 이 되었을 때, 나누는 작업을 중단한다

$$20385_{10} = 4FA1_{16}.$$

5.2 Rep. of Int. and Int. Algorithms

- 예 5.2.10 Add the binary numbers 10011011 and 1011011
- Sol)

$$\begin{array}{r} 10011011 \\ +\quad 1011011 \\ \hline \end{array}$$

Beginning from the right, adding 1 and 1 gives10_2, thus we write 0 and carry 1. At this point the computation is

$$\begin{array}{r} 1 \\ 10011011 \\ +\quad 1011011 \\ \hline 0 \end{array}$$

Adding 1 and 1 and 1 gives 11_2 . Write 1 and carry 1

$$\begin{array}{r} 1 \\ 10011011 \\ +\quad 1011011 \\ \hline 10 \end{array} \quad => \quad \begin{array}{r} 10011011 \\ +\quad 1011011 \\ \hline 11110110 \end{array}$$

5.2 Rep. of Int. and Int. Algorithms

- Algorithm 5.2.12 Adding Binary Numbers

  This algorithm adds the binary numbers $b_n b_{n-1} \cdots b_1 b_0$ and $b'_n b'_{n-1} \cdots b'_1 \, b'_0$ and stores the sum in $s_{n+1} s_n s_{n-1} \cdots s_1 s_0$

  Input: b, b', n

  Output: s

  binary addition(b, b , n, s) {

  carry = 0

  for $i = 0$ to n {

  $s_i = (b_i + b'_i + \text{carry}) \bmod 2$

  $\text{carry} = (b_i + b'_i + \text{carry})/2$

  }

  $s_{n+1} = \text{carry}$

  }

5.2 Rep. of Int. and Int. Algorithms

- a^n: $n-1$번 곱셈, $a^{29} = a^1 \cdot a^4 \cdot a^8 \cdot a^{16}$: 4 + 3 번 곱셈
- Algorithm 5.2.16 Exponentiation by Repeated Squaring
- 거듭제곱법을 이용하여 a^n을 계산한다..

Input: a, n Output: a^n

$exp\_via\_repeated\_squaring(a, n)$ {
  $result = 1$
  $x = a$
  while ($n > 0$) {
    if ($n \bmod 2 == 1$)
      $result = result * x$
    $x = x * x$
    $n = \lfloor n/2 \rfloor$
  }
  return $result$
}

1	1	1	0	1
a^{16}	a^8	a^4	a^2	a^1
$29_{10} = 11101_2$				

5.2 Rep. of Int. and Int. Algorithms

$\boxed{n = dq + r}$

- Theorem 5.2.17 (a^n mod z 계산) a, b, z를 양의 정수라 하면,

$$ab \bmod z = [(a \bmod z)(b \bmod z)] \bmod z.$$

- 증명 $w = ab \bmod z, x = a \bmod z, y = b \bmod z$라고 하자

w는 ab를 z로 나눌 때의 나머지이므로 몫-나머지 정리에 의해

$$ab = q_1 z + w$$

을 만족하는 q_1이 존재한다. 따라서 $w = ab - q_1 z$. 정리에 의해

$$a = q_2 z + x, \qquad b = q_3 z + y.$$

을 만족하는 정수 q_2, q_3가 존재한다.

$$w = ab - q_1 z = (q_2 z + x)(q_3 z + y) - q_1 z = qz + xy,$$

여기서 $q = q_2 q_3 z + q_2 y + q_3 x - q_1$. 따라서

$$xy = -qz + w$$

그러므로

$$w = xy \bmod z.$$

5.2 Rep. of Int. and Int. Algorithms

$$572^{29} \cong 9.2 \times 10^{79}$$
$$29 = 1 + 4 + 8 + 16$$

- 예 5.2.18 $572^{29} \bmod 713$

$$\begin{aligned}
572^{2} \bmod 713 &= (572 \bmod 713)(572 \bmod 713) \bmod 713 \\
&= (572 \cdot 572) \bmod 713 = 630 \\
572^{4} \bmod 713 &= (572^{2} \bmod 713)(572^{2} \bmod 713) \bmod 713 \\
&= (630 \cdot 630) \bmod 713 = 472 \\
572^{8} \bmod 713 &= (572^{4} \bmod 713)(572^{4} \bmod 713) \bmod 713 \\
&= (472 \cdot 472) \bmod 713 = 328 \\
572^{16} \bmod 713 &= (572^{8} \bmod 713)(572^{8} \bmod 713) \bmod 713 \\
&= (328 \cdot 328) \bmod 713 = 634 \\
572^{5} \bmod 713 &= (572 \bmod 713)(572^{4} \bmod 713) \bmod 713 \\
&= (572 \cdot 472) \bmod 713 = 470 \\
572^{13} \bmod 713 &= (572^{5} \bmod 713)(572^{8} \bmod 713) \bmod 713 \\
&= (470 \cdot 328) \bmod 713 = 152 \\
572^{29} \bmod 713 &= (572^{13} \bmod 713)(572^{16} \bmod 713) \bmod 713 \\
&= (152 \cdot 634) \bmod 713 = 113.
\end{aligned}$$

5.2 Rep. of Int. and Int. Algorithms

- Algorithm 5.2.19 거듭제곱에 의한 누승수의 mod z

  거듭제곱법을 이용하여 $a^n \bmod z$를 계산한다.

  Input: a, n, z Output: $a^n \bmod z$

  $exp\_mod\_z\_via\_repeated\_squaring(a, n, z)$ {

  $\quad result = 1$

  $\quad x = a \bmod z$

  $\quad$ while $(n > 0)$ {

  $\quad\quad$ if $(n \bmod 2 == 1)$

  $\quad\quad\quad result = (result * x) \bmod z$

  $\quad\quad x = (x * x) \bmod z$

  $\quad\quad n = \lfloor n/2 \rfloor$

  $\quad$ }

  $\quad$ return $result$

  }

5.3 The Euclidean Algorithm

- 유클리드 알고리즘$^{\text{Euclidean algorithm}}$은 두 정수의 최대 공약수를 찾기 위한 것으로, 오래되고 유명하며 효율적이다.
- 만약 $r = a \bmod b$이면 다음이 성립한다

$$\gcd(a, b) = \gcd(b, r)$$

- 예제 5.3.1 $\gcd(105, 30)$를 구하라
  위 식에 의해 $105 \bmod 30 = 15$ 이므로

$$\gcd(105, 30) = \gcd(30, 15)$$

  이다. 위 식에 의해 $30 \bmod 15 = 0$ 이므로

$$\gcd(30, 15) = \gcd(15, 0)$$

  이다. $\gcd(15, 0) = 15$ 이므로

$$\gcd(105, 30) = \gcd(30, 15) = \gcd(15, 0) = 15$$

5.3 The Euclidean Algorithm

- 정리 5.3.2 a가 음이 아닌 정수, b가 양의 정수, $r = a \bmod b$이면,

$$\gcd(a, b) = \gcd(b, r)$$

- 증명 a와 b의 공약수의 집합은 b와 r의 공약수의 집합과 같다는 것을 보임으로써 증명할 수 있다.
- 몫-나머지 정리에 의해 다음을 만족시키는 q와 r이 존재한다.

$$a = bq + r \qquad 0 \leq r < b$$

- c를 a와 b의 공약수라고 하자. 정리 5.1.3(c)에 의해 $c \mid bq$이다. $c \mid a$이고 $c \mid bq$이므로 정리 5.1.3(b) 에 의해 $c \mid a - bq\ (= r)$이다. 따라서 c는 b와 r의 공약수이다.
- 반대로 c가 b와 r의 공약수이면 $c \mid bq$이고, $c \mid bq + r (= a)$이다. 따라서 c는 a와 b의 공약수이다.

(b) if $d \mid m$ and $d \mid n$, then $d \mid (m - n)$.
(c) if $d \mid m$ then $d \mid mn$.
(a) if $d \mid m$ and $d \mid n$, then $d \mid (m + n)$.

5.3 The Euclidean Algorithm

- Algorithm 5.3.3 Euclidean Algorithm
- Input: a, b (음이 아니고 동시에 0 이 아님)
- Output: a와 b의 최대 공약수
-
```
gcd(a, b) {
    if (a < b)
        swap(a, b)
    while (b ¬= 0) {
        r = a mod b
        a = b
        b = r
    }
    return a
}
```

$$\begin{aligned} &\gcd(396,108) \\ &= \gcd(108,72) \\ &= \gcd(72,36) \\ &= \gcd(36,0) \\ &= 36 \end{aligned}$$

정리 5.3.6 $0 < a, b < m$ $(m \geq 8)$일 때, 나눗셈 횟수는 최대 $\log_{3/2} \frac{2m}{3}$ 이다

5.3 The Euclidean Algorithm

. 2740과 1760의 최대공약수 계산

q	r_1	r_2	r
1	2740	1760	980
1	1760	980	780
1	980	780	200
3	780	200	180
1	200	180	20
9	180	20	0
	20	0	

. 25와 60의 최대공약수 계산

q	r_1	r_2	r
0	25	60	25
2	60	25	10
2	25	10	5
2	10	5	0
	5	0	

5.3 The Euclidean Algorithm

- Theorem 5.3.7 a와 b가 음이 아니고 동시에 0이 아닌 정수라 하면 다음을 만족시키는 정수 s와 t가 존재한다.

$$\gcd(a,b) = sa + tb$$

- 증명) $a > b \geq 0$이 주어지고,
  $r_0 = a, r_1 = b$ 라하고,
  r_{i+1}는 알고리즘에서 while 반복문이 i번 실행된 후의 r 값이라 하자.
  (예, $r_2 = a \bmod b$).
  r_n이 처음으로 0이 된다고 가정한다.
  그러면 $\gcd(a,b) = r_{n-1}$이다.

```
gcd(a, b) {
    if (a < b)
        swap(a, b)
    while (b ¬= 0) {
        r = a mod b
        a = b
        b = r
    }
    return a
}
```

- 일반적으로 다음이 성립한다.
  $r_{i+2} = r_i \bmod r_{i+1}$에서

$$r_{i+2} = -q_{i+2} r_{i+1} + r_i \qquad (3.10)$$

5.3 The Euclidean Algorithm

$$r_{i+2} = -q_{i+2} r_{i+1} + r_i \qquad (3.10)$$

- (3.10)에서 $i = n - 3$ 로 하면,

$$r_{n-1} = -q_{n-1} r_{n-2} + r_{n-3} = t_{n-3} r_{n-2} + s_{n-3} r_{n-3}$$

(3.10)에서 $i = n - 4$ 로 하면, $r_{n-2} = -q_{n-2} r_{n-3} + r_{n-4}$

$$\begin{aligned} r_{n-1} &= t_{n-3}(-q_{n-2} r_{n-3} + r_{n-4}) + s_{n-3} r_{n-3} \\ &= [-t_{n-3} q_{n-2} + s_{n-3}] r_{n-3} + t_{n-3} r_{n-4} \\ &= t_{n-4} r_{n-3} + s_{n-4} r_{n-4} \end{aligned}$$

이런 방식으로 계속하면 결국 다음식이 성립한다.

$$\gcd(r_0, r_1) = r_{n-1} = t_0 r_1 + s_0 r_0 = t_0 b + s_0 a = sa + tb.$$

5.3 The Euclidean Algorithm

- 예제 5.3.8 유클리드 알고리즘으로 gcd(273,110) 계산

  $r = 273 \bmod 110 = 53 \quad (3.5)$, $\quad r = 110 \bmod 53 = 4 \quad (3.6)$

  $r = 53 \bmod 4 = 1 \quad (3.7)$, $\quad r = 4 \bmod 1 = 0$

  $r = 0$이므로 알고리즘은 종료하며, $\gcd(273,110) = 1$.

- s와 t를 찾기 위해 역순으로 $r \neq 0$ 인 마지막 식부터 시작한다.

  (3.7)에서, $1 = 53 - 4 \cdot 13$. (3.6)에서, $4 = 110 - 53 \cdot 2$.

  $$1 = 53 - (110 - 53 \cdot 2) \cdot 13 = 27 \cdot 53 - 13 \cdot 110$$

  (3.5)에서, $53 = 273 - 110 \cdot 2$.

  $$1 = 27 \cdot 53 - 13 \cdot 110 = 27 \cdot (273 - 110 \cdot 2) - 13 \cdot 110$$

  $$= 27 \cdot 273 - 67 \cdot 110$$

  따라서 $s = 27, t = -67$로 하면,

  $$\gcd(273, 110) = 1 = s \cdot 273 + t \cdot 110.$$

5.3 The Euclidean Algorithm

- Algorithm 5.3.9 Recursive Euclidean Algorithm

  음이 아니고 동시에 0 이 아닌 정수 a, b의 최대 공약수를 재귀적으로 찾는 알고리즘이다

  Input: a와 b (음이 아니고 동시에 0 이 아닌 정수)
  Output: a와 b의 최대 공약수

```
gcdr(a, b) {
   if (a < b)
         swap(a, b)
   if (b == 0)
      return a
   r = a mod b
   return gcdr(b, r)
}
```

5.3 The Euclidean Algorithm

- Algorithm 5.3.10 Computing s and t of Theorem 5.3.7

  음이 아니고 동시에 $\gcd(a,b) = sa + tb$ 를 만족시키는 s와 t를 계산

  Input: a와 b (음이 아니고 동시에 0이 아닌 정수)

  Output: s, t, $\gcd(a,b)$

$STgcdr(a,b,s,t)$ {

  $if\ (a < b)$ // make a largest

   $\mathrm{swap}(a,b)$

  $if\ (b == 0)$ {   // $\gcd(a,0) = a = 1a + 0b.$

   $s = 1$

   $t = 0$ // now a = sa + tb

   return a

  }

  $q = \lfloor a/b \rfloor$

  $r = a \bmod b$ // $a = bq + r \quad \therefore r = a - bq$

  $g = STgcdr(b,r,s',t')$ // $g = s'b + t'r$

  $s = t'$ // $g = s'b + t'(a - bq)$

  $t = s' - t' * q$ // $\therefore\ g = t'a + (s' - t'q)b$

  return g

}

5.3 The Euclidean Algorithm

- An Inverse Modulo an Integer (나머지 연산의 역원)
- $n > 0, \phi > 1$를 $\gcd(n, \phi) = 1$인 정수라 하자. $ns \bmod \phi = 1$, $0 < s < \phi$를 만족하는 정수 s를 $n \bmod \phi$ 의 역원이라 한다
- 유클리드 알고리즘을 사용하여 $s'n + t'\phi = \gcd(n, \phi) = 1$이 되는 정수 s′ 과 t'을 찾을 수 있다. 그러면 $ns' = -t'\phi + 1$이 되고, $\phi > 1$이므로 1이 나머지가 된다. 따라서

$$ns' \bmod \phi = 1 \qquad (3.13)$$

- $0 < s' < \phi$를 만족시키지 않을 수 있다. $s = s' \bmod \phi$라 하자. 이제 $0 \le s < \phi$이 된다. 만약 $s = 0$이면 $\phi \mid s'$가 되어 (3.13)과 모순되므로 $0 < s < \phi$이다.

  $s = s' \bmod \phi$이므로 $s' = q\phi + s$인 정수 q가 존재. 앞 식을 결합,

$$ns = ns' - \phi nq = -t'\phi + 1 - \phi nq = \phi(-t' - nq) + 1.$$

  따라서

$$ns \bmod \phi = 1 \qquad (3.14)$$

5.3 The Euclidean Algorithm

- 예제: n=3, ϕ =7 일때 3 mod 7의 역수?

  1단계. 0에서 ϕ -1까지의 S값에 대해 n * S mod ϕ 를 계산

  3 * 0 mod 7 = 0

  3 * 1 mod 7 = 3

  3 * 2 mod 7 = 6

  3 * 3 ≡ 9 mod 7 = 2

  3 * 4 ≡ 12 mod 7 = 5

  3 * 5 ≡ 15 mod 7 = 1 <------ 역수

  3 * 6 ≡ 18 mod 7 = 4 (mod 7)

  2단계. n mod ϕ 의 Mod역수는 n * S mod ϕ = 1를 만족하는 S값

  5*3 mod 7 = 1이므로 3 mod 7의 역수는 5.

5.3 The Euclidean Algorithm

- 예제 5.3.11 $n = 110, \phi = 273$라 하자. 예제 5.3.8에서 $\gcd(n, \phi) = 1$이고, $s' = -67, t' = 27$일 때 $s'n + t'\phi = 1$이 성립하였다. 따라서

$$110(-67) \bmod 273 = ns' \bmod \phi = 1.$$

여기서 $s = s' \bmod \phi = -67 \bmod 273 = 206.$

그러므로, 110 mod 273의 역원은 206이다

- 식 (3.14)에서 s가 유일한다.

다음을 가정해보자

$$ns \bmod \phi = 1, ns' \bmod \phi = 1, \qquad 0 < s < \phi, 0 < s' < \phi.$$

$s = s'$임을 증명해야 한다. $s' = s' \bmod \phi, s = s \bmod \phi$ 이므로

$$s' = s' \cdot 1 = (s' \bmod \phi)(ns \bmod \phi) = s'ns \bmod \phi$$
$$= [(s'n \bmod \phi)(s \bmod \phi)] \bmod \phi = 1 \cdot s = s.$$

5.4 The RSA Public Key Cryptosystem

- 송신자는 메시지를 **암호화**[encrypt]하여 보내고,
  수신자는 받은 메시지를 **복호화**[decrypt] 한다
- 가장 오래되고 가장 간단한 시스템 중 하나는 송신자와 수신자가 **키**[key]를 정해 각각의 문자를 다른 문자로 바꾸어 보낸다. 송신자와 수신자는 key(비밀키)를 공개하지 않는다
- 예제 5.4.1 만약 키를 아래와 같이 정한다면,

  문자: ⊔ABCDEFGHIJKLMNOPQRSTUVWXYZ

  대체: EIJFUAXVHWP⊔GSRKOBTQYDMLZNC

  메시지 SEND⊔MONEY는 암호화 하면 QARUESKRAN이 되고,

  암호화된 메시지 SKRANEKRELIN는 MONEY⊔ON⊔WAY로 복호화 된다

5.4 The RSA Public Key Cryptosystem

- RSA 공개키 암호 시스템

1. Alice가 Bob(수신자)에게 정보를 안전하게 보내고자 한다.
2. B가 공개키와 개인키를 만들어 A에게 공개키를 보낸다. (개인키는 B만 가지고 있다.)
3. A는 B가 보낸 공개키를 이용하여 보낼 정보를 암호화 한다.
4. A가 암호화된 정보를 B에게 보낸다.
5. B가 암호화된 정보를 받고 개인키를 이용하여 복호화 한다.

- RSA에서는 메시지는 숫자로 표시 한다. 만약 빈 칸을 1로, A를 2로, ..., Z를 27로, 표시하면, 메시지 SEND MONEY는 20, 06, 15, ~~05, 01,~~ 14, 16, 15, 06, 26이 된다. 한 개의 정수로 만들면,

$$20061505011416150626$$

5.4 The RSA Public Key Cryptosystem

- RSA 작동, 공개 키(z, n)와 개인키(s)를 생성 (by 수신자)
  1. 소수 p와 q를 선택하여, $z = pq$를 계산
  2. $\phi = (p-1)(q-1)$를 계산, $\gcd(n, \phi) = 1$인 정수 n를 선택
  3. 공개 키(z, n)를 공개
  4. $0 < s < \phi$이고 $ns \bmod \phi = 1$인 유일한 개인키 s를 계산
- 암호화 (by 송신자)

  송신할 정수 a ($0 \le a \le z-1$)를 $c = a^n \bmod z$ 로 암호화해 송신
- 복호화 (by 수신자)

  $c^s \bmod z$를 계산, 이 값이 a이다.

예 $p = 23, q = 31, n = 29$ 를 선택. $z = pq = 713, \phi = 660,$

$29 \cdot 569 \bmod 660 = 1$이므로 $s = 569$(개인키)이다.

$a = 572$를 보내기 위해, 송신자는 $a^n \bmod z = 572^{29} \bmod 713 = 113$ 계산하여, 113을 보낸다. 수신자는 이 메시지를 복호화 하기 위해 $c^s \bmod z = 113^{569} \bmod 713 = 572$를 계산

5.4 The RSA Public Key Cryptosystem

- 개인 키를 계산해서 복호화
  개인 키를 구하기 위해서는 공개 키인 z의 인수 분해가 필요하며, 아직까지 효율적인 인수 분해 알고리즘이 없어 큰 z (300 또는 600자리 이상)에 대해서는 비현실적이다.
- 개인키 s를 모르면 $c^s \bmod z$로 계산하여 a를 구하는 대신, $c = a^n \bmod z$ 를 만족하는 정수 a를 구한다. 아직까지 a를 효율적으로 계산할 수 있는 알고리즘이 없다