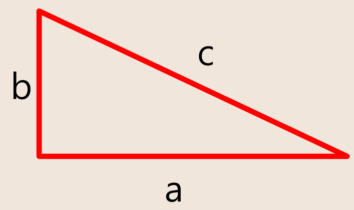
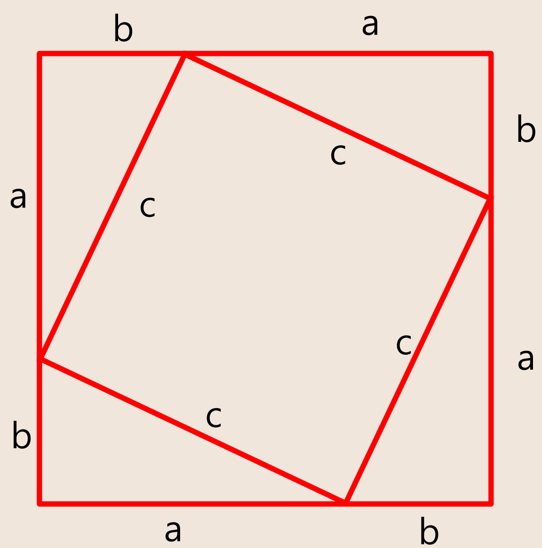
정수론 (INTEGER THEORY)

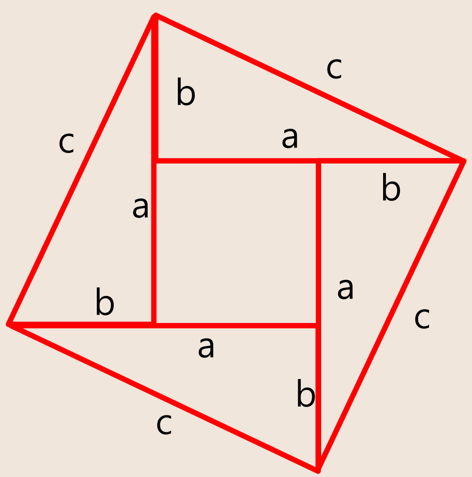
피타고라스 정리 증명

- $a^2 + b^2 = c^2$
- 가장 이해하기 쉬운 증명 2가지



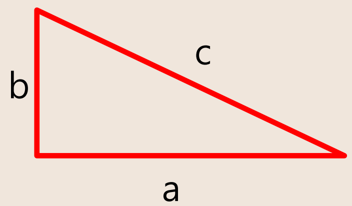


$(a+b)^2 = 2ab + c^2 \longrightarrow a^2 + b^2 + 2ab = 2ab + c^2$



$$\begin{aligned} c^2 &= 2ab + (a-b)^2 \\ &= 2ab + a^2 + b^2 - 2ab \end{aligned}$$

피타고라스 수



- $a^2 + b^2 = c^2$
- Pythagorean triple: 위 식을 만족하는 세 양의 정수
  - '피타고라스 수' 또는 '피타고라스 삼조'라 부름
  - Ex: (3,4,5), (6,8,10), …
- Primitive Pythagorean triple: a,b,c가 서로소일 때 즉, gcd(a,b,c)=1
  - '원시 피타고라스 수' 또는 '원시 피타고라스 삼조'라 부름
  - Ex: (3,4,5), (5,12,13), …
- 임의의 홀수 m에 대해: $\left(m, \frac{m^2-1}{2}, \frac{m^2+1}{2}\right)$은 피타고라스 수
  - Ex: (3,4,5), (5,12,13), (7,24,25), (9,40,41), (11,60,61), …

피타고라스 수

- 임의의 자연수 $m(> 1)$에 대해: $\left(2m,\ \ m^2 - 1,\ m^2 + 1\right)$은 피타고라스 수
  - 증명: $(2m)^2 + (m^2 - 1)^2 = (m^2 + 1)^2$
  - Ex: (4,3,5), (6,8,10), (8,15,17), (10,24,26)

- 위의 방법으로 피타고라스 수를 구하면 (4,3,5), (6,8,10) 등이 나와 원시 피타고라수 수가 아닌 것이 있다.

- 어떻게 원시 피타고라스 수를 구할까?

피타고라스 수

- (a, b, c)가 **원시 피타고라스 수**일 필요충분조건
  - $\left(m^2 - n^2,\ 2mn,\ m^2 + n^2\right)$
  - 여기서 m, n은 자연수이고, $m > n$ 이며, 둘 중 하나는 짝수, 하나는 홀수이

m n	a	b	c
	m^2 - n^2	2mn	m^2 +n^2
2 1	3	4	5
3 2	5	12	13
4 1	15	8	17
4 3	7	24	25
5 2	21	20	29
5 4	9	40	41
6 1	35	12	37
6 5	11	60	61
7 2	45	28	53
7 4	33	56	65
7 6	13	84	85

피타고라스 수

(a, b, c)가 **원시 피타고라스 수**일 때:

- 따름정리 1: a, c 는 항상 홀수이고 b는 4의 배수이다.
- 따름정리 2: a 또는 b 중 적어도 하나는 3의 배수이다.
- 따름정리 3: a, b, c 중 적어도 하나는 5의 배수이다.
- 따름정리 4: ab는 12의 배수이고 abc는 60의 배수이다.

m n	a	b	c
	m^2 - n^2	2mn	m^2 +n^2
2 1	3	4	5
3 2	5	12	13
4 1	15	8	17
4 3	7	24	25
5 2	21	20	29
5 4	9	40	41
6 1	35	12	37
6 5	11	60	61
7 2	45	28	53
7 4	33	56	65
7 6	13	84	85

피타고라스 수

- (a, b, c)가 **원시 피타고라스 수**일 필요충분조건
  - $(m^2 - n^2,\ 2mn,\ m^2 + n^2)$
  - 여기서, m, n은 자연수이고, $m > n$ 이며, 둘 중 하나는 짝수, 하나는 홀수이다

- **따름정리 1: $\boldsymbol{a, c}$ 는 항상 홀수이고 $\boldsymbol{b}$는 4의 배수이다.**
  - 증명:

    m, n 중 하나만이 홀수 이므로 $a = m^2 - n^2$ 이 홀수이다.

    또한 $c = m^2 + n^2$ 홀수이다.

    $b = 2mn$ 이 4의 배수임은 쉽게 알 수 있다.

피타고라스 수

- (a, b, c)가 **원시 피타고라스 수**일 필요충분조건
  - $(m^2 - n^2,\ 2mn,\ m^2 + n^2)$
  - 여기서, m, n은 자연수이고, $m > n$ 이며, 둘 중 하나는 짝수, 하나는 홀수이다

- **따름정리 2: $\boldsymbol{a}$ 또는 $\boldsymbol{b}$ 중 적어도 하나는 3의 배수이다.**
  - 증명:

    만약 m 또는 n 이 3의 배수이면 $b = 2mn$ 는 3의 배수이다.

    만약 m 또는 n 이 3의 배수가 아니면

    $m = 3k + 1,\ n = 3k - 1$ 이다.

    따라서 $a = m^2 - n^2 = (3k + 1)^2 - (3k - 1)^2 = 12k$ 는 3의 배수이다.

피타고라스 수

- (a, b, c)가 **원시 피타고라스 수**일 필요충분조건
  - $(m^2 - n^2,\ 2mn,\ m^2 + n^2)$
  - 여기서, m, n은 자연수이고, $m > n$ 이며, 둘 중 하나는 짝수, 하나는 홀수이다

- **따름정리 3: a, b, c 중 적어도 하나는 5의 배수이다.**
  - 증명:

    만약 m 또는 n 이 5의 배수이면 $b = 2mn$ 는 5의 배수이다.

    만약 m 또는 n 이 5의 배수가 아니면

    5의 배수가 아닌 4가지 수 $5k+4,\ 5k+3,\ 5k+2,\ 5k+1$ 에서 $m > n$ 이 만족되고,
    둘 중 하나는 짝수, 하나는 홀수가 되도록 m, n을 설정한 후, 각 경우에 대해
    a, c 값을 따져 보면 a, c 둘 중 하나는 5의 배수가 됨을 쉽게 알 수 있다.
    (참고로, 위 조건을 만족하도록 m, n을 설정하는 경우는 총 4가지임)

    예를 들어, $m = 8$, $n = 7$로 두면 a가 5의 배수가 된다.
    $m = 9$, $n = 8$로 두면 c가 5의 배수가 된다.

피타고라스 수

(a, b, c)가 **원시 피타고라스 수**일 때:

- 따름정리 1: a, c 는 항상 홀수이고 b는 4의 배수이다.
- 따름정리 2: a 또는 b 중 적어도 하나는 3의 배수이다.
- 따름정리 3: a, b, c 중 적어도 하나는 5의 배수이다.

- **따름정리 4: $\boldsymbol{ab}$는 12의 배수이고 $\boldsymbol{abc}$는 60의 배수이다.**
  - 증명:

    따름정리 1에 의해 b는 4의 배수, 따름정리 2에 의해 a 또는 b 중 적어도 하나는 3의 배수이다.
    따라서 ab는 12의 배수이다.

    따름정리 1,2,3에 의하면 abc는 60의 배수임 알 수 있다.

정수론-서론

정리 1

- m, n, c가 정수일 때,

  (a) 만약 c가 m, n의 공약수이면 $c|(m+n)$

  (b) 만약 c가 m, n의 공약수이면 $c|(m-n)$

  (c) 만약 $c|m$ 이면 $c|m \cdot n$

정리 2

- 두 정수 $a(\geq 0)$와 $b(>0)$ 가 있을 때,

  $a = b \cdot q + r \ (0 \leq r < b)$ 이면 $\gcd(a,b) = \gcd(b,r)$ 이다

유클리드 알고리즘

유클리드 알고리즘(Euclid algorithm)

- 정리 2에 근거하여 gcd를 빠르게 찾는 알고리즘
- gcd(a,b) = gcd(b, a mod b)
- a mod b: a를 b로 나눈 나머지 (a%b)

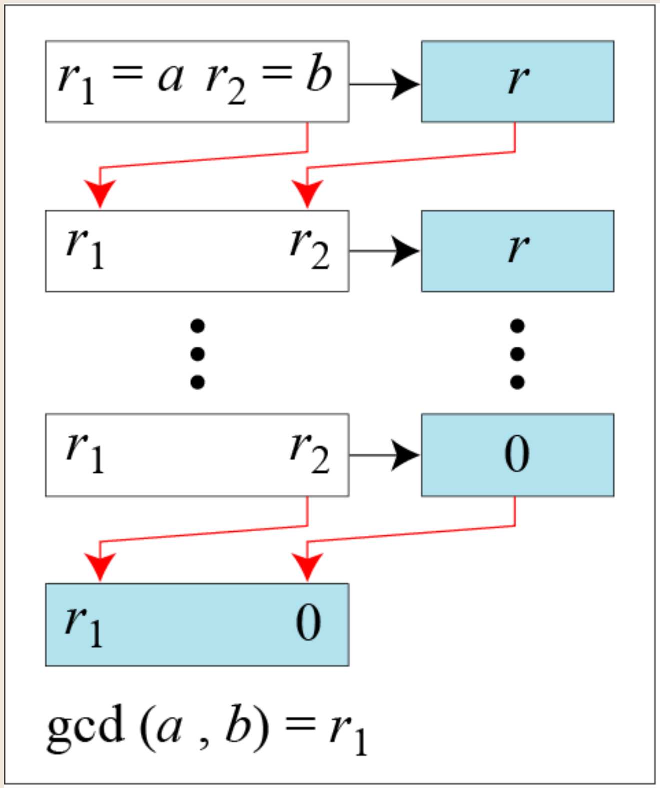
```
gcd(a,b)
    if(b==0) return a;
    return gcd(a, a%b);
```

실행 예

- gcd(385, 175) = gcd(175, 35) = gcd(35, 0) = 35
- gcd(15, 8) = gcd(8, 7) = gcd(7, 1) = gcd(1, 0) = 1

유클리드 알고리즘

처리 과정 및 알고리즘 (non-recursion)



a. Process

$r_1 \leftarrow a; \qquad r_2 \leftarrow b;$ (Initialization)

while ($r_2 > 0$)

{

$\quad q \leftarrow r_1 / r_2;$

$\quad r \leftarrow r_1 - q \times r_2;$

$\quad r_1 \leftarrow r_2; \quad r_2 \leftarrow r;$

}

$\gcd(a, b) \leftarrow r_1$

b. Algorithm

확장된 유클리드 알고리즘

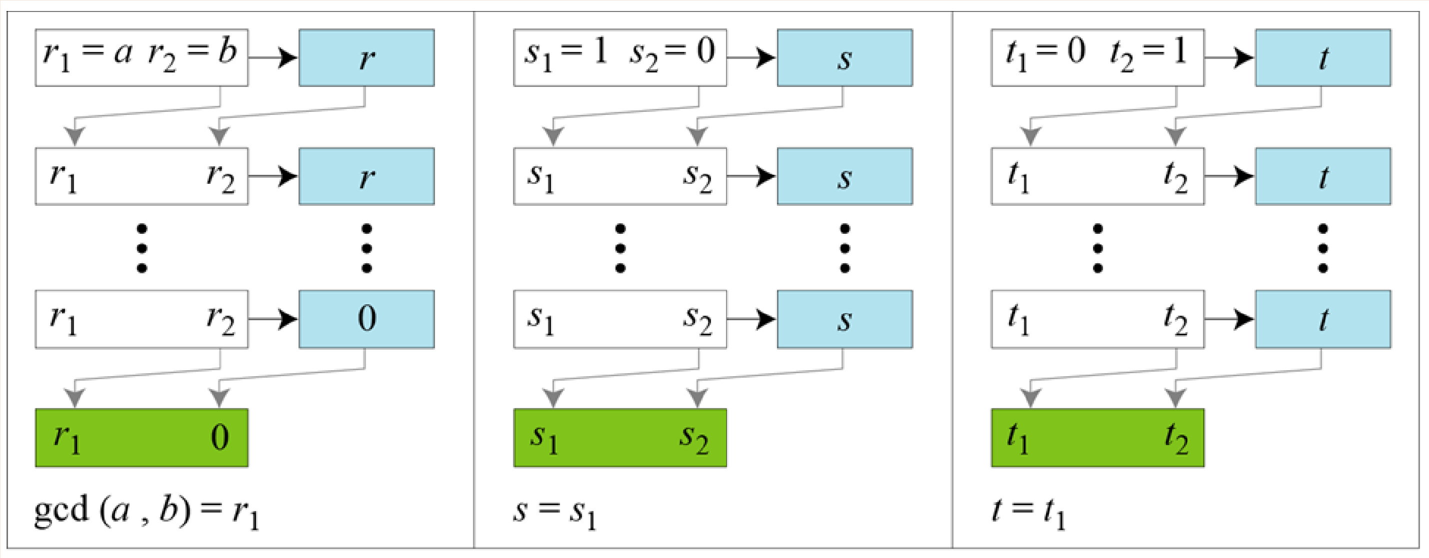
정리 3

- a, b가 양의 정수이면 gcd(a,b) = a·s + b·t 를 만족하는 정수 s, t가 존재한다.

확장된 유클리드 알고리즘(Extended Euclid Algorithm)

- gcd(a,b) = a·s + b·t 를 만족하는 정수 s, t를 찾아 준다.

처리과정



확장된 유클리드 알고리즘

$r_1 \leftarrow a; \quad r_2 \leftarrow b;$
$s_1 \leftarrow 1; \quad s_2 \leftarrow 0;$ (Initialization)
$t_1 \leftarrow 0; \quad t_2 \leftarrow 1;$

while ($r_2 > 0$)
{
$q \leftarrow r_1 / r_2;$

$r \leftarrow r_1 - q \times r_2;$
$r_1 \leftarrow r_2; \; r_2 \leftarrow r;$ (Updating r's)

$s \leftarrow s_1 - q \times s_2;$
$s_1 \leftarrow s_2; \; s_2 \leftarrow s;$ (Updating s's)

$t \leftarrow t_1 - q \times t_2;$
$t_1 \leftarrow t_2; \; t_2 \leftarrow t;$ (Updating t's)
}

gcd $(a, b) \leftarrow r_1; \; s \leftarrow s_1; \; t \leftarrow t_1$

확장된 유클리드 알고리즘 실행 예

- **gcd(375, 275) = ?**
  - r_1 = 375, r_2 = 275, s_1 = 1, s_2 = 0, t_1 = 0, t_2 = 1 ← 초기화

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
	375	275		1	0		0	1	

확장된 유클리드 알고리즘 실행 예

- **gcd(375, 275) = ?**
  - r_1 = 375, r_2 = 275, s_1 = 1, s_2 = 0, t_1 = 0, t_2 = 1 ← 초기화
  - r_1 (375)을 r_2 (275)로 나눈 몫 q (1)를 구한다.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
1	375	275		1	0		0	1	

확장된 유클리드 알고리즘 실행 예

- gcd(375, 275) = ?
  - r_1 = 375, r_2 = 275, s_1 = 1, s_2 = 0, t_1 = 0, t_2 = 1 ← 초기화
  - r_1 (375)을 r_2 (275)로 나눈 몫 q (1)를 구한다.
  - $r = r_1 - q \times r_2$;

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
1	375	275	100	1	0		0	1	

확장된 유클리드 알고리즘 실행 예

- **gcd(375, 275) = ?**
  - r_1 = 375, r_2 = 275, s_1 = 1, s_2 = 0, t_1 = 0, t_2 = 1 ← 초기화
  - r_1 (375)을 r_2 (275)로 나눈 몫 q (1)를 구한다.
  - $r = r_1 - q \times r_2$; $s = s_1 - q \times s_2$;

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
1	375	275	100	1	0	1	0	1	

확장된 유클리드 알고리즘 실행 예

- **gcd(375, 275) = ?**
  - r_1 = 375, r_2 = 275, s_1 = 1, s_2 = 0, t_1 = 0, t_2 = 1 ← 초기화
  - r_1 (375)을 r_2 (275)로 나눈 몫 q (1)를 구한다.
  - $r = r_1 - q \times r_2$; $s = s_1 - q \times s_2$; $t = t_1 - q \times t_2$;

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
1	375	275	100	1	0	1	0	1	-1

확장된 유클리드 알고리즘 실행 예

- gcd(375, 275) = ?
  - r_1 = 375, r_2 = 275, s_1 = 1, s_2 = 0, t_1 = 0, t_2 = 1 ← 초기화
  - r_1 (375)을 r_2 (275)로 나눈 몫 q (1)를 구한다.
  - $r = r_1 - q \times r_2$; $s = s_1 - q \times s_2$; $t = t_1 - q \times t_2$;
  - r 이 0 이 아니므로 반복 계속

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
1	375	275	100	1	0	1	0	1	-1

확장된 유클리드 알고리즘 실행 예

- **gcd(375, 275) = ?**
  - $r_1 = 275$, $r_2 = 100$, $s_1 = 0$, $s_2 = 1$, $t_1 = 1$, $t_2 = -1$

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
1	375	275	100	1	0	1	0	1	-1
	275	100		0	1		1	-1	

확장된 유클리드 알고리즘 실행 예

gcd(375, 275) = ?

- r_1 = 275, r_2 = 100, s_1 = 0, s_2 = 1, t_1 = 1, t_2 = -1
- r_1 (275)을 r_2 (100)로 나눈 몫 q (2)를 구한다.
- $r = r_1 - q \times r_2$; $s = s_1 - q \times s_2$; $t = t_1 - q \times t_2$;
- r 이 0 이 아니므로 반복 계속

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
1	375	275	100	1	0	1	0	1	-1
2	275	100	75	0	1	-2	1	-1	3

확장된 유클리드 알고리즘 실행 예

- gcd(375, 275) = ?

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
1	375	275	100	1	0	1	0	1	-1
2	275	100	75	0	1	-2	1	-1	3
1	100	75	25	1	-2	3	-1	3	-4
3	75	25	0	-2	3	-11	3	-4	13
	25	0		**3**	-11		**-4**	13	

- gcd(375, 275) = 25 = 375·(3) + 275·(-4)

일차 디오판토스 방정식(Linear Diophantine Equation)

- **다음 식을 만족하는 정수 x, y가 존재하는가?**
  - $2x + y = 7$
  - $2x + 4y = 9$
  - $3x + 6y = 10$

- **정리 4 (디오판토스 방정식)**
  - a, b, c가 정수일 때,

    $a \cdot x + b \cdot y = c$ 를 만족하는 정수 x, y가 존재 $\leftrightarrow \gcd(a,b) \mid c$

일차 디오판토스 방정식(Linear Diophantine Equation)

예: 85·x + 34·y = 51 을 만족하는 x, y를 구하라.

- 85·x + 34·y = 51 ------------------------------ 식 (1)
- 확장된 유클리드 알고리즘을 이용하여

  gcd(85, 34) = 17 = (85)·(1) + 34·(-2)
- 즉, (85)·(1) + 34·(-2) = 17 --------------------- 식 (2)
- 17|51 이다. 정리 4에 의해 해가 존재
- 51/17 = 3 이다.
- 식(2) 양변에 x3

  (85)·(1)·(3) + 34·(-2)·(3) = (17)·(3) <--- 이는 식 (1)과 일치
- 따라서 x = (1)·(3) = 3, y = (-2)·(3) = -6

일차 디오판토스 방정식(Linear Diophantine Equation)

일반화: $a \cdot x + b \cdot y = c$ 를 만족하는 x, y를 구하라.

- $a \cdot x + b \cdot y = c$
- 확장된 유클리드 알고리즘을 이용하여

  $\gcd(a, b) = g = a \cdot s + b \cdot t$
- 즉, $a \cdot s + b \cdot t = g$

- 여기서, $g|c$ 인지를 검사하여 해의 존재 여부를 판단
- $g|c$ 이면 $k = c/g \Rightarrow c = k \cdot g$
- $a \cdot s + b \cdot t = g$ 이므로 $a \cdot s \cdot k + b \cdot t \cdot k = g \cdot k = c$
- 따라서, $x = s \cdot k$, $y = t \cdot k$

일차 디오판토스 방정식(Linear Diophantine Equation)

예: 85·x + 34·y = 51 을 만족하는 x, y를 구하라.

- gcd(85, 34) = 17 = (85)·(1) + 34·(-2) ← 확장된 유클리드 알고리즘
- = (85)·(-1) + 34·(3) = (85)·(1) + 34·(-2) = (85)·(3) + 34·(-7) …
- 17 = (85)·(1) + 34·(-2)

→ 1 = (5)·(1) + 2·(-2)

→ 1 = (5)·(3) + 2·(-7)

→ 1 = (5)·(5) + 2·(-12)

→ 1 = (5)·(7) + 2·(-17)

모듈러 연산(Modular Operations)

모듈러 연산(modular operation)

- a (mod n) 은 a를 n으로 나누었을 때 나머지를 의미한다.
  - 11(mod 7) = 4
  - -11(mod 7) = 3
  - 10(mod 5) = 0
- a (mod n) ≡ b (mod n) 을 만족하면 a와 b는 'congruent modulo n' 이라고 함

모듈러 연산의 표기

1) $n|(a-b)$라면 → $a \equiv b \pmod{n}$
2) $a \pmod{n} \equiv b \pmod{n}$ → $a \equiv b \pmod{n}$
3) $a \equiv b \pmod{n}$ → $b \equiv a \pmod{n}$
4) $a \equiv b \pmod{n}$ 그리고 $b \equiv c \pmod{n}$ → $a \equiv c \pmod{n}$

모듈러 연산(Modular Operations)

모듈러 산술연산

1) [a (mod n) + b (mod n)] (mod n) = (a+b) (mod n)
2) [a (mod n) - b (mod n)] (mod n) = (a-b) (mod n)
3) [a (mod n) × b (mod n)] (mod n) = (a×b) (mod n)

Ex: $3^{100}(mod\ 10) = ?$

$$3^{100}(mod\ 10) = [3^{50}(mod\ 10) \times 3^{50}(mod\ 10)](mod\ 10)$$
$$3^{50}(mod\ 10) = [3^{25}(mod\ 10) \times 3^{25}(mod\ 10)](mod\ 10)$$
$$3^{25}(mod\ 10) = [3^{12}(mod\ 10) \times 3^{12}(mod\ 10) \times 3](mod\ 10)$$
$$3^{12}(mod\ 10) = [3^{6}(mod\ 10) \times 3^{6}(mod\ 10)](mod\ 10)$$
$$3^{6}(mod\ 10) = [3^{3}(mod\ 10) \times 3^{3}(mod\ 10)](mod\ 10)$$
$$3^{3}(mod\ 10) = 7$$

모듈러 연산(Modular Operations)

모듈러 산술연산

1) [a (mod n) + b (mod n)] (mod n) = (a+b) (mod n)
2) [a (mod n) - b (mod n)] (mod n) = (a-b) (mod n)
3) [a (mod n) × b (mod n)] (mod n) = (a×b) (mod n)

Ex: $3^{100}(mod\ 10) = ?$

$3^{100}(mod\ 10) = [3^{50}(mod\ 10) \times 3^{50}(mod\ 10)](mod\ 10)$ → 1
$3^{50}(mod\ 10) = [3^{25}(mod\ 10) \times 3^{25}(mod\ 10)](mod\ 10)$ → 9
$3^{25}(mod\ 10) = [3^{12}(mod\ 10) \times 3^{12}(mod\ 10) \times 3](mod\ 10)$ → 3
$3^{12}(mod\ 10) = [3^{6}(mod\ 10) \times 3^{6}(mod\ 10)](mod\ 10)$ → 1
$3^{6}(mod\ 10) = [3^{3}(mod\ 10) \times 3^{3}(mod\ 10)](mod\ 10)$ → 9
$3^{3}(mod\ 10) = 7$

모듈러 연산(Modular Operations)

- Z_n ={0,1,2,3, … ,(n−1)}
  - 임의의 Z(정수)를 n으로 나누었을 때의 나머지 집합

- **덧셈의 역원**
  - 두 정수 a, b가 다음을 만족하면 Z_n 상에서 서로가 덧셈에 대한 역원이다.
  - $a + b \equiv 0 \pmod{n}$

- **곱셈의 역원**
  - 두 정수 a, b가 다음을 만족하면 Z_n 상에서 서로가 곱셈에 대한 역원이다.
  - $a \times b \equiv 1 \pmod{n}$

모듈러 연산(Modular Operations)

Z_7 에서의 덧셈, 곱셈 그리고 역원

$(6+1)(mod 7)=0$

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

(a) 7진법 덧셈

$(4\times2)(mod 7)=1$

+	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

(b) 7진법 곱셈

w	−w	w^{-1}
0	0	–
1	6	1
2	5	4
3	4	5
4	3	2
5	2	3
6	1	6

(c) 7진법 덧셈과 곱셈의 역원

모듈러 연산(Modular Operations)

- Z_8 에서의 덧셈, 곱셈 그리고 역원

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

*	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

w	−w	1/w
0	0	
1	7	1
2	6	
3	5	3
4	4	
5	3	5
6	2	
7	1	7

합동식(congruence equation)

합동식

- 정수 a, b, n이 주어질 때, $a \cdot x \equiv b \pmod{n}$ 을 만족하는 x는?
- 즉, $a \cdot x$ 를 n 으로 나눌 때 나머지가 b가 되는 x 값은?
- 예를 들어, $3x \equiv 1 \pmod{5}$ 를 만족하는 x는 { … 2, 7, 12, …}이다.
- 즉, $x \equiv 2 \pmod{5}$ 로 쓸 수 있다.

- 참고: $18x \equiv 16 \pmod{7} \rightarrow 4x \equiv 2 \pmod{7}$

정리 5

- $ax \equiv b \pmod{n}$ 을 만족하는 x가 존재 ↔ $\gcd(a,n) \mid b$
- 이는 정리 4와 같은 의미

$ax \equiv b \pmod{n}$

$\Rightarrow ax = n \cdot q + b$, $ax - nq = b$, $ax + ny = b \Rightarrow \gcd(a, n)$

$(y = -q)$

합동식

(3의 역원) x2

예:

- 3x ≡ 2 (mod 5) 를 만족하는 x를 구하자. ➔ x ≡ ?? (mod 5)

- gcd(3,5)|2 이므로 해가 존재
- (mod 5) 에서 3의 곱에 대한 역원 3^{-1} 을 구하자.
- 그러면 $3 \cdot 3^{-1} \cdot x \equiv 2 \cdot 3^{-1} \pmod 5$ ➔ $x \equiv 2 \cdot 3^{-1} \pmod 5$ 이 된다.

- (mod 5) 에서 3의 곱에 대한 역원 3^{-1} 을 어떻게 구하나?
- 3^{-1}는 $3 \cdot a \equiv 1 \pmod 5$ 를 만족하는 a 이다.
- 이는 $3 \cdot a + 5 \cdot b = 1$ 을 만족하는 일차디오판토스 식의 해를 구하는 것과 같다.
- 이는 확장된 유클리드 알고리즘을 사용하여 구할 수 있다.
- a = 2, b = -1 이 $3 \cdot a + 5 \cdot b = 1$ 을 만족시킨다.
- 즉 (mod 5)에서 곱에 대한 3의 역원 $3^{-1} = 2$ 이다.

- $x \equiv 2 \cdot 3^{-1} \pmod 5$ ➔ $x \equiv 2 \cdot 2 \pmod 5$ ➔ $x \equiv 4 \pmod 5$

중국인 나머지 정리(Chinese Remainder Theorem)

- CRT **기원** : '3**으로 나누면** 1**이 남고** 5**로 나누면** 2**이 남고** 7**로 나누면** 3**이 남는 수 중에서 제일 작은 수는**?'

  - $x \equiv a_1 \pmod{m_1}$
  - $x \equiv a_2 \pmod{m_2}$

    . . . .
  - $x \equiv a_r \pmod{m_r}$

중국인의 나머지 정리: $m_1, m_2, \ldots, m_r$ 이 양의 정수이면서 서로 소라고 하자.
임의의 정수 $a_1, a_1, \ldots, a_r$ 에 대하여 다음 r 개의 합동식
$x=a_i \pmod{m_i}$ $(i=1,2,\ldots,r)$ 은 공통해를 갖고 서로 다른 두 해의 차이는
$m_1 * m_2 * \ldots * m_r$ 로 나누어 떨어진다.

중국인 나머지 정리(Chinese Remainder Theorem)

- $x \equiv a_1 \pmod{m_1}$
- $x \equiv a_2 \pmod{m_2}$

. . . . . .

- $x \equiv a_r \pmod{m_r}$

문제를 해결하는 큰 흐름

- 첫 두 식을 동시에 만족하는 x 를 구해 $x \equiv A(\mathrm{mod}\ [m_1\ m_2])$ 로 둠
- 앞에서 구한 식과 세번째 식을 동시에 만족하는 식을 구해

  $x \equiv B(\mathrm{mod}\ [m_1,m_2,m_3])$ 둠

이 과정을 반복

중국인 나머지 정리

다음 연립방정식의 해를 구하라.

$x \equiv 5 \pmod{6}$ 식(1)

$x \equiv 3 \pmod{10}$ 식(2)

$x \equiv 8 \pmod{15}$ 식(3)

식(1) $\Rightarrow x = 6s + 5$ 식(4)

식(4) & 식(2) $\Rightarrow 6s + 5 \equiv 3 \pmod{10} \Rightarrow 6s \equiv 8 \pmod{10}$ /2

$\Rightarrow 3s \equiv 4 \pmod{5} \Rightarrow s \equiv 3 \pmod{5}$

즉, $s = 5t + 3$ 식(5)

식(4) & 식(5) $\Rightarrow x = 6(5t + 3) + 5 \Rightarrow x = 30t + 23$ 식(6)

식(6) & 식(3) $\Rightarrow 30t + 23 \equiv 8 \pmod{15} \Rightarrow 30t \equiv 0 \pmod{15}$

$\Rightarrow 2t \equiv 0 \pmod{1} \Rightarrow t = u$ 식(7)

식(7) & 식(6) $\Rightarrow x = 30u + 23 \Rightarrow$ $\mathbf{x \equiv 23 \pmod{30}}$

중국인 나머지 정리

다음 연립방정식의 해를 구하라.

$x \equiv 9 \pmod{12}$ 식(1)

$x \equiv 0 \pmod{9}$ 식(2)

$x \equiv 3 \pmod{15}$ 식(3)

$x \equiv 13 \pmod{16}$ 식(3)

식(1) $\Rightarrow x = 12s + 9$ 식(5)

식(5) & 식(2) $\Rightarrow 12s + 9 \equiv 0 \pmod{9} \Rightarrow 4s \equiv 0 \pmod{3} \Rightarrow s \equiv 3t$ 식(6)

식(6)을 식(5)에 대입 $\Rightarrow x = 36t + 9$ 식(7)

식(7) & 식(3) $\Rightarrow 36t + 9 \equiv 3 \pmod{15}$ $\Rightarrow t \equiv 4 \pmod{5}$

$\Rightarrow t \equiv 5u + 4$ 식(8)

식(8)을 식(7)에 대입 $\Rightarrow x = 180u + 153$ 식(9)

식(9) & 식(4) $\Rightarrow 180u + 153 \equiv 13 \pmod{16}$

$\Rightarrow u \equiv 1 \pmod{4} \Rightarrow u \equiv 4v + 1$ 식(10)

식(10)을 식(9)에 대입 $\Rightarrow x = 720v + 333$ $\Rightarrow$ **$x \equiv 333 \pmod{720}$**

오일러 함수(Euler Function)

오일러 함수 Φ(n) (Euler Φ function)

- n 보다 작고 n과 서로소인 양의 정수의 개수. Φ(1) = 1로 정의됨

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ϕ(n)	1	1	2	2	4	2	6	4	6	4	10	4	12	6	8
n	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
ϕ(n)	8	16	6	18	8	12	10	22	8	20	12	18	12	28	8

정리 7

- n과 m이 서로소라면 Φ(nm) = Φ(n)Φ(m)

정리 8

- p가 소수이면 $Φ(p^k) = p^k – p^{k-1}$

정리 9(정리 8의 따름 정리)

(a) p가 소수 ↔ Φ(p) = p – 1

(b) $Φ(2^k) = 2^{k-1}$

오일러 함수(Euler Function)

- **정리** 10

일반적으로, $a = p_1^{m_1} p_2^{m_2} \dots p_k^{m_k}$이라 하면,

$\Phi(a) = a\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\dots\left(1 - \frac{1}{p_k}\right)$가 된다.

- **정리** 11(**오일러 정리**)
  - 양수 m에 대해 gcd(a, m) = 1 이면 $a^{\Phi(m)} \equiv 1 \pmod{m}$

- **정리** 12(**페르마의 소정리**)
  - p가 소수이면, $(0 < a < p)$ 인 모든 a에 대해 $a^{p-1} \equiv 1 \pmod{p}$

RSA 알고리즘

RSA **알고리즘 개요**

- 1978년 발표된 공개키 암호 알고리즘
- Ron **R**ivest, Adi **S**hamir, Leonard **A**dleman 의 이름 첫 글자를 모음

RSA **알고리즘**

1. 서로 다른 두 개의 소수 p, q를 선택한다.
2. $n = p \cdot q$
3. $\Phi(n)=\Phi(p)\ \Phi(q)=(p-1)(q-1)$을 계산
4. $gcd(e, \Phi(n))=1$ $(1 < e < \Phi(n)$인 e를 선택한다.
5. $d \equiv e^{-1} \pmod{\Phi(n)}$ 을 계산. 즉, $d \cdot e \equiv 1 \pmod{\Phi(n)}$ 인 d를 찾음

☞ (n, e)를 공개키로, (n, d)를 개인키로 저장함
☞ 전송할 message가 m 이면, 암호문 $c = m^e \pmod{n}$, $m = c^d \pmod{n}$ 로 계산
☞ 단계 1의 소수 p, q는 밀러-라빈(Miller-Rabin) 알고리즘을 이용하여 구한다.
☞ 단계 4의 e는 유클리드 알고리즘을 이용하여 구한다.
☞ 단계 5의 d는 확장된 유클리드 알고리즘을 이용하여 구한다.

RSA 알고리즘 실행 예

1. p = 61, q = 53
2. n = 61 x 53 = 3233
3. Φ(3233) = Φ(61) x Φ(53) = 62 x 52 = 3120
4. 3120과 서로소인 17을 e 선택 (1 < e < 3120)
5. d = 2753 을 얻음

   (확장된 유클리드 알고리즘 이용하여 가능)

즉, 공개키: (3233, 17) 개인키: (3233, 2753)

만약 m = 65 라면,

암호문 c = 65^{17} (mod 3233) = 2790 을 얻음

암호문 2790으로부터 원문 복원은

2790^{2753} (mod 3233) = 65 = m

RSA **알고리즘**

1. 서로 다른 소수 p, q를 선택한다.
2. n = p·q
3. Φ(n)=Φ(p) Φ(q)=(p-1)(q=1) 을 계산
4. gcd(e, Φ(n))=1 (1 < e < Φ(n))인 e를 선택한다.
5. d ≡ e^{-1} (mod Φ(n)) 을 계산. 즉, d·e ≡ 1 (mod Φ(n) 인 d를 찾음

(n, e)를 공개키로,

(n, d)를 개인키로 저장함

c = m^e (mod n)

m = c^d (mod n)

RSA 알고리즘

RSA **알고리즘 정확성 증명**

- 이 알고리즘의 정확성은 페르마의 소정리(정리 12)에 근거한다.
- $m^{ed} \equiv m \pmod{pq}$을 보이고자 한다.
  (여기서 p, q는 서로 다른 소수이며 $ed \equiv 1 \pmod{\Phi(pq)}$)
- $\Phi(pq) = (p-1)(q-1)$ 이므로 $ed - 1 = h(p-1)(q-1)$ $(h > 0)$ 로 쓸 수 있다.
- $m^{ed} \equiv m \pmod{pq}$ 가 참임을 보이기 위해 $m^{ed} \equiv m \pmod{p}$이 참이고
  $m^{ed} \equiv m \pmod{q}$이 참임을 각각 보이고자 한다. (정리 5(f) 참조)
- $m^{ed} \equiv m \pmod{p}$ 이 참임을 보이자. 이는 두 경우로 나누어 생각:
  경우 (1) $m \equiv 0 \pmod{p}$,   경우 (2) $m \not\equiv 0 \pmod{p}$
- 경우 (1): m은 p의 배수이고, $m^{ed} \equiv 0 \equiv m \pmod{p}$ 이 되어 참이다.
- 경우 (2): $m^{ed} = m^{(ed-1)} \cdot m = m^{h(p-1)(q-1)} \cdot m = (m^{p-1})^{h(q-1)} \cdot m \equiv (1)^{h(q-1)} \cdot m \equiv m \pmod{p}$
- (m^{p-1}) 를 1 로 바꾸기 위해 페르마의 소정리를 사용했다.
- 같은 방법으로 $m^{ed} \equiv m \pmod{q}$ 가 참임을 보일 수 있다.
- $m^{ed} \equiv m \pmod{p}$, $m^{ed} \equiv m \pmod{q} \rightarrow m^{ed} \equiv m \pmod{pq}$ (정리 5(f))
- 즉, $(m^e)^d \equiv m \pmod{n}$