정수론 : (⭐을 붙인 부분만 읽으셔도 됩니다)

작성자	박 박영서
태그	Euclidean algorithm · Extended Euclidean algorithm · Greatest Common Divisor
생성일	@2023년 1월 29일 오후 10:08

☝️ 서울대학교 권오남 교수님의 [정수론] 강의를 편집한 내용입니다. (링크 클릭 후 가입 시 수강 가능함)
문제 발생 시 삭제될 수 있습니다.

[1] Divisibility, Congruence

(1-1) 정의

▼ Divisibility (가분성. 나눌 수 있음)

a, b가 정수일 때, a가 b를 나눈다는 것은 a * k = b가 되는 정수 k가 존재한다는 것이다

$$a|b \leftrightarrow \exists k, s.t. b = a \times k$$

▼ Congruence (합동)

a, b가 정수이고 n이 0보다 큰 정수일 때, a가 법 n에 대해 b와 합동이라는 것은 n이 a-b를 나눈다는 것이다.

$$a \equiv b (mod\, n) \leftrightarrow n|(a-b)$$

(1-2) 정리

아래에서 a, b, c는 모두 정수이고 n은 양의 정수.

▼ Theorem 1.1 ⭐

$$(i)\ a|b\text{이고 } a|c\text{이면, } a|(b+c)\text{이다}$$
$$(ii)\ a|b\text{이고 } a|c\text{이면, } a|(b-c)\text{이다. 즉 } b \equiv c (mod\, a)\text{이다.}$$

▼ 증명

$$\begin{aligned}
&(i)\\
&\quad a|b\text{이므로 어떤 정수} k_1\text{에 대해 } b = ak_1\text{이다.}\\
&\quad a|c\text{이므로 어떤 정수} k_2\text{에 대해 } c = ak_2\text{이다.}\\
&\quad b + c = (ak_1 + ak_2) = a(k_1 + k_2)\text{이고 } (k_1 + k_2)\text{ 또한 정수이므로, } a|(b+c)\text{이다.}\square\\
&(ii)\\
&\quad (i)\text{ 과 마찬가지로 증명할 수 있다}
\end{aligned}$$

▼ Theorem 1.2. ⭐

$$(i)\ a|b,\ a|c\text{이면 } a|(bc)\text{이다}$$
$$(ii)\ a|b\text{ 이면 } a|(bc)\text{이다}$$

증명은 생략

▼ Theorem 1.3.

n이 0보다 큰 정수라고 하면 다음이 성립한다.

$(i)\, a \equiv a(mod\, n)$
$(ii)\, a \equiv b(mod\, n)$이면, $b \equiv a(mod\, n)$
$(iii)\, a \equiv b(mod\, n)$이고 $b \equiv c(mod\, n)$이면 $a \equiv c(mod\, n)$

▼ 증명

(i)
$a \equiv a(mod\, n) \leftrightarrow\ n|(a-a) = n|0.$
모든 정수 n이 0을 나누므로 이는 성립함$\Box$
(ii)
$a \equiv b(mod\, n) \leftrightarrow\ n|(a-b)$이므로
$(a-b) = nk$가 되는 어떤 정수 k가 존재.
양변에 -1을 곱하면 $(b-a) = -nk = (-k)n$이 되고,
$-k$ 또한 정수이므로 $n|(b-a) \leftrightarrow\ b \equiv a(mod\, n)$
따라서 $a \equiv b(mod\, n)$이면 $b \equiv a(mod\, n).$
(iii)
$a \equiv b(mod\, n)$이므로 $n|(a-b)$이고, 따라서 $(a-b) = nk_1$이
되는 정수 k가 존재.
$b \equiv c(mod\, n)$이므로 $n|(b-c) \leftrightarrow \exists k_2 \in \mathbb{Z}, s.t., (b-c) = nk_2.$
$(a-b)+(b-c) = (a-c) = (nk_1 + nk_2)$
$(a-c) = n(k_1+k_2), (k_1+k_2) \in \mathbb{Z}$이므로 $n|(a-c) \leftrightarrow a \equiv c(mod\, n)\Box$

▼ Theorem 1.4.

$(i)\, a \equiv b(mod\, n)$이고 $c \equiv (mod\, n)$이면 $(a \pm c) \equiv d(b \pm d)(mod\, n)$이다.
$(ii)\, a \equiv b(mod\, n),\ c \equiv d(mod\, n)$이면 $ac \equiv bd(mod\, n)$이다.
$(iii) k > 0 \in \mathbb{Z}$에대해, $a \equiv b(mod\, n)$이면 $a^k \equiv b^k(mod\, n)$이다.

▼ 증명

$(i,+$증명$)a \equiv b(mod\, n) \leftrightarrow n|(a-b).\, (a-b) = nk_1$이 되는 $k_1 \in \mathbb{Z}$가 존재
$c \equiv d(mod\, n) \leftrightarrow n|(c-d).\, (c-d) = nk_2$가 되는 $k_2 \in \mathbb{Z}$가 존재
$(a-b)+(c-d) = nk_1 + nk_2 = n(k_1+k_2)$
$(a+c)-(b+d) = n(k_1+k_2)$
$n|((a+c)-(b+d)) \leftrightarrow (a+c) \equiv (b+d)(mod\, n)\Box$
$(-$도 마찬가지로 증명 가능$)$
(ii)
$\exists k_1, k_2 \in \mathbb{Z} s.t.,\ a-b = nk_1, c-d = nk_2.$
$a = nk_1 + b, c = nk_2 + d$
$ac = (nk_1+b)(nk_2+d) = (nk_1nk_2 + nk_1d + nk_2b + bd)$
$= n(nk_1nk_2 + k_1d + k2_b) + bd$
$ac - bd = n(nk_1nk_2 + k_1d + k2_b)$
$n|(ac-bd) \leftrightarrow ac \equiv bd(mod\, n)\Box$
(iii)
$a \equiv b(mod\, n)$에 대해 (ii)를 활용해서 증명 가능함.

▼ Theorem 1.5.

자연수 n과 m이 주어졌을 때, m을 n으로 나눈 몫과 나머지는 유일하게 존재한다. (정수에까지 확장 가능함.)

$(i)n$과 m이 모두 자연수일 때, $m = nq + r$가 되는 $r \in \mathbb{N}(0 \le r < n)$가 존재한다.$(existence)$
(ii)또한 $m = nq_1 + r_1 = nq_2 + r_2$를 만족하는 $q_1, q_2, r_1, r_2 \in \mathbb{Z}(0 \le r_1, r_2 < n)$이 존재한다면 $q_1 = q_2, r_1 = r_2$

증명은 생략

▼ Theorem 1.6.

$$a \equiv b(mod\, n) \iff a\text{와 } b\text{가 } n\text{으로 나누었을 때 같은 나머지를 가짐.}$$
$$\text{즉, } a \equiv b(mod\, n) \iff a = nq_1 + r_1,\ b = nq_2 + r_2\text{이면 } r_1 = r_2(\text{단 } 0 \le r_1, r_2 < n)$$

▼ 증명

$$(\rightarrow)$$
$$a \equiv b(mod\, n) \iff \exists k \in \mathbb{Z}, s.t.(a-b) = nk$$
$$b = a - nk = (nq_1 + r_1) - nk = n(q_1 - k) + r_1 = nq_2 + r_2.$$
$$\therefore r_1 = r_2$$
$$(\leftarrow)$$
$$a = nq_1 + r_1, b = nq_2 + r_2\text{이므로 } a - b = nq_1 - nq_2 = n(q_1 - q_2)$$
$$\therefore n|(a-b),\ a \equiv b(mod\, n)$$

---

[2] Greatest Common Divisor

(2-1) 정의

▼ Common Divisor(공약수) ⭐

정수 a, b의 공약수 d는 $d|a, d|b$인 정수이다.

▼ Greatest Common Divisor(최대공약수) ⭐

정수 a, b의 최대공약수 d는 공약수 중 가장 큰 공약수이다. d = (a, b)로 표기

(2-2) 정리

▼ Theorem 2.1. ⭐

$$a, n, b, r, k\text{가 정수라고 하자.}$$
$$\text{만약} a = nb + r\text{이고 } k|a, k|b\text{이면, } k|r\text{이다.}$$

▼ 증명

$$k|a,\ k|b\text{이므로 } a = kc_1, b = kc_2\text{라고 하자.}$$
$$a = nb + r\text{이므로 } kc_1 = nkc_2 + r\text{이고,}$$
$$r = kc_1 - nkc_2 = k(c_1 - nc_2)\text{이다.}$$
$$(c_1 - nc_2) = c^*\text{라고 할 때, } c^* \in \mathbb{Z}\text{이므로} k|r\text{이다.}\square$$

▼ Theorem 2.2. ⭐

$$a, b, n, r_1\text{가 정수라고 하자.}$$
$$\text{만약} a = nb + r_1\text{이면} (a, b) = (b, r_1)\text{이다.}$$

▼ 증명

$C = \{c|(c|a)\&(c|b)\}, D = \{d|(d|b)\&(d|r)\}$라고 하자.
$(\subseteq) Theorem 2.1.$에 따라 C의 임의의 원소 c는 d의 원소이므로, $C \subseteq D$이다.
$(\supseteq) D$의 임의의 원소 d에 대해, $d|b$이고 $d|r$이다. $Theorem 1.2$에 따라 $d|b$이므로 $d|bn_1$이고
$Theorem 1.1$에 따라 $d|bn_1$이고 $d|r$이면 $d|bn_1 + r$, 즉 $d|a$이다.
D의 임의의 원소 $d \in C$이므로, $D \subseteq C$이다.
$(\therefore) C \subseteq D, D \subseteq C$이므로 $C = D$이고, 따라서 C의 최댓값 $= D$의 최댓값이다.
즉 a, b의 최대공약수는 b, r의 최대공약수이다. $\square$

(2-3) Euclidean Algorithm ⭐⭐

Theorem2.2에 따라 정수 a, b의 최대공약수 (a,b)를 구하는 알고리즘이다.

- ▼ Q. 그냥 소인수분해 해서 구하면 안되나요?

  A. 작은 수면 할 수도 있겠지만, 소인수를 구하는 다항식 시간 알고리즘이 아직은(?) 발견되지 못했습니다.

  [소인수분해] ← 이 문서를 참고.
  작은 수면 모르겠지만 큰 수에 대해서는 소인수 분해를 적용하기가 힘든데, 유클리드 알고리즘을 이용하면 확실히 두 수의 최대공약수를 구할 수 있음!

- ▼ 유클리드 알고리즘 (설명)

$$
\begin{gathered}
a = bn_1 + r_1 \text{이라고 하자. } (a,b) = (b, r_1)\text{이다.} \\
b = q_2 r_1 + r_2 \text{라고 하자. } (b, r_1) = (r_1, r_2)\text{이다.} \\
r_1 = q_3 r_2 + r_3 \text{이라고 하자. } (r_1, r_2) = (r_2, r_3)\text{이다.} \\
\dots \\
\text{위 작업을 } r_{n-1} = q_{n+1} r_n + r_{n+1} \text{에 대해 } r_{n+1} \text{가 } 0 \text{이 될 때까지 반복한다.} \\
\text{이때 얻을 수 있는 } 0 \text{이 아닌 가장 작은 정수 } r_n \text{이 바로 } (a,b)\text{이다.}
\end{gathered}
$$

- ▼ 위 알고리즘을 그대로 c++ 코드로 쓰면 다음과 같다

```
int gcd(int a, int b){
    int dividend = a;
    int divisor = b;
    int quotient;
    int remainder;

    while (a && b){
        quotient = dividend / divisor;
        remainder = dividend % divisor;
        printf("%d = %d * %d + %d\n", dividend, quotient, divisor, remainder);
        if (remainder == 0) return divisor;
        dividend = divisor;
        divisor = remainder;
    }
    return -1; //입력 a, b 중 하나라도 0인 경우 -1을 리턴해서 연산이 제대로 되지 않았음을 표시했습니다.
}
```

```
int gcd(int a, int b){//재귀를 이용한 코드
    if (b == 0) return a;
    return gcd(b, a%b);
}
```

---

[3] Linear Diophantine Equations & Bezout's Identity (증명은 나중에…)

(3-1) 정리(1)

- ▼ Theorem 3.1.

$$
\begin{gathered}
a, b\text{가 정수라고 하자.} \\
a, b\text{가 서로소이다. } \iff \exists x, y \in \mathbb{Z}, s.t., ax + by = 1
\end{gathered}
$$

▼ 증명

▼ Theorem 3.2. ⭐

$$0\text{이 아닌 두 정수 } a, b\text{에 대해, 다음을 만족하는 두 정수 } x, y\text{가 존재한다.}$$
$$ax + by = (a, b)$$

▼ Theorem 3.3. (Euclid's Lemma)

$$\text{정수 } a, b, c\text{에 대해, 만약 } a|bc\text{이고, } a\text{와 } b\text{가 서로소이면 } a|c\text{이다.}$$

▼ Theorem 3.4.

$$a, b, n\text{이 정수라 하자.}$$
$$a|n,\ b|n,\ (a, b) = 1\text{이면 } ab|n\text{이다.}$$

▼ Theorem 3.5.

$$(a, n) = 1,\ (b, n) = 1\text{이면 } (ab, n) = 1\text{이다.}$$

▼ Theorem 3.6.

$$a, b, c, n\text{이 정수이고, } n > 0\text{이라 하자.}$$
$$ac \equiv bc (mod\, n)\text{이고 } (c, n) = 1\text{이면 } a \equiv b (mod\, n)\text{이다.}$$

(3-2) 정리(2)

▼ Theorem 3.7. ⭐

$$a, b\text{가 } 0\text{이 아닌 정수이고 } c\text{가 정수이면 다음을 만족하는 정수 } x, y\text{가 존재한다.}$$
$$ax + by = c \iff (a, b)|c$$

▼ Theorem 3.8.

$$a, b, c, x_0, y_0\text{이 } ax_0 + by_0 = c\text{인 정수라 하자}$$
$$x = x_0 + b/(a, b),\ y = y_0 - a/(a, b)\text{인 정수 } x, y \text{ 또한 } ax + by = c\text{를 만족한다.}$$

▼ Theorem 3.9.

$$ax_0 + by_0 = c\text{라면, 임의의 정수 } k\text{에 대해,}$$
$$\text{정수 } x = x_0 + kb/(a, b),\ y = y_0 - ka/(a, b) \text{ 또한 } ax + by = c\text{를 만족한다.}$$
$$\text{또한, } ax + by = c\text{의 모든 해들은 위와 같은 형태로 나타난다.}$$

(3-3) 확장 유클리드 알고리즘 ⭐

Theorem 3.7.에서 $ax + by = c$이면 $(a, b)|c$임을 보였다.
확장 유클리드 알고리즘은 (a, b)를 구하면서 동시에 $ax + by = (a, b)$가 되는 정수 x, y를 구하는 알고리즘이다.

유클리드 알고리즘을 다시 살펴보면 다음과 같다.

$$\begin{gathered} a = q_1 b + r_1 \\ b = q_2 r_1 + r_2 \\ r_1 = q_3 r_2 + r_3 \\ r_2 = q_4 r_3 + r_4 \\ \dots \\ r_{n-1} = q_{n+1} r_n + r_{n+1}\ (r_{n+1} = 0, r_n = (a, b)) \end{gathered}$$

$$
\begin{aligned}
r_1 =& a - q_1 b \\
r_2 =& b - q_2 r_1 \\
=& b - q_2(a - q_1 b) \\
=& b - q_2 a + q_1 q_2 b \\
=& -q_2 a + (1 + q_1 q_2)b \\
=& \dots
\end{aligned}
$$

이렇게 유클리드 알고리즘에서 보이는 모든 나머지들은 a와 b의 일차결합($ax+by$의 꼴)으로 나타낼 수 있다.

이를 일반화해 다음과 같이 표현하자.

$$r_i = s_i a + t_i b$$

여기서 a 를 r_{-1}, b를 r_0이라고 하면 다음과 같이 표현할 수 있다.

$$
\begin{gathered}
r_{-1} = 1 \times a + 0 \times b \\
r_0 = 0 \times a + 1 \times b \\
\text{즉, } s_{-1} = 1, s_0 = 0, t_{-1} = 0, t_0 = 1
\end{gathered}
$$

유클리드 알고리즘에서 $r_{n-1} = q_{n+1} r_n + r_{n+1}$임을 얻었으니, 우리는 다음과 같은 점화식을 얻을 수 있다.

$$
\begin{aligned}
r_{i+1} = s_{i+1}a + t_{i+1}b =& r_{n-1} - q_{n+1} r_n \\
=& s_{i-1}a + t_{i-1}b - q_{n+1}(s_i a + t_i b) \\
=& (s_{i-1} - q_{n+1} s_i)a + (t_{i-1} - q_{n+1} t_i)b \\
\therefore s_{i+1} = s_{i-1} - q_{n+1} s_i&, t_{i+1} = t_{i-1} - q_{n+1} t_i
\end{aligned}
$$

우리가 알고 싶은 건 $r_n = ax + by$가 되는 x, y값(즉 s_n, t_n)이고, 이는 위의 점화식을 이용해 구할 수 있다!

▼ CODE

```
/*
ax + by = (a,b)가 되는
{{x, y}, (a,b)}를 찾는 함수
*/
pair<pair<int,int>, int> extendedGCD(int a, int b){
    int dividend = a, divisor = b;
    int quotient, remainder;
    int s_prev = 1, s_cur = 0, s_next;
    int t_prev = 0, t_cur = 1, t_next;

    while (a && b){
        quotient = dividend / divisor;
        remainder = dividend % divisor;
        if (remainder == 0) return {{s_cur, t_cur}, divisor};
        s_next = s_prev - quotient * s_cur;
        t_next = t_prev - quotient * t_cur;
        s_prev = s_cur, s_cur = s_next;
        t_prev = t_cur, t_cur = t_next;
        dividend = divisor;
        divisor = remainder;
    }
    return {{-1, -1}, -1}; //잘못된 입력의 경우
}
```

끝!