整数論の基礎

暗号方式の多くは、整数論を基盤として設計されています。情報システムでは無限大の数を扱うことはできないので、法計算という計算を定義して、限られた範囲内の整数で計算ができるようにします。ここでは、法計算について基本的な事項を学びます。

合同式

- 合同の定義（a,bは整数（負でもよい）、nは正の整数とする）
  $a=b \pmod n$
  は、(a-b)がnで割り切れるという意味であると定義する。
- （例）
  - 3=10(mod 7) ⇔ (3-10)=-7 が 7で割り切れる
  - 13=8(mod 5) ⇔ (13-8)=5 が 5で割り切れる
  - -1 = 11 (mod 12) ⇔ (-1-11)=-12 が 12で割り切れる

（注）a=b (mod n)の意味として、aはbをnで割った余りである、と説明されることがある。しかし、この説明では、上の2番目や３番目の例はうまく説明できない。

同値関係とは

- 前頁の合同の定義は、数学的な同値関係を満たしている

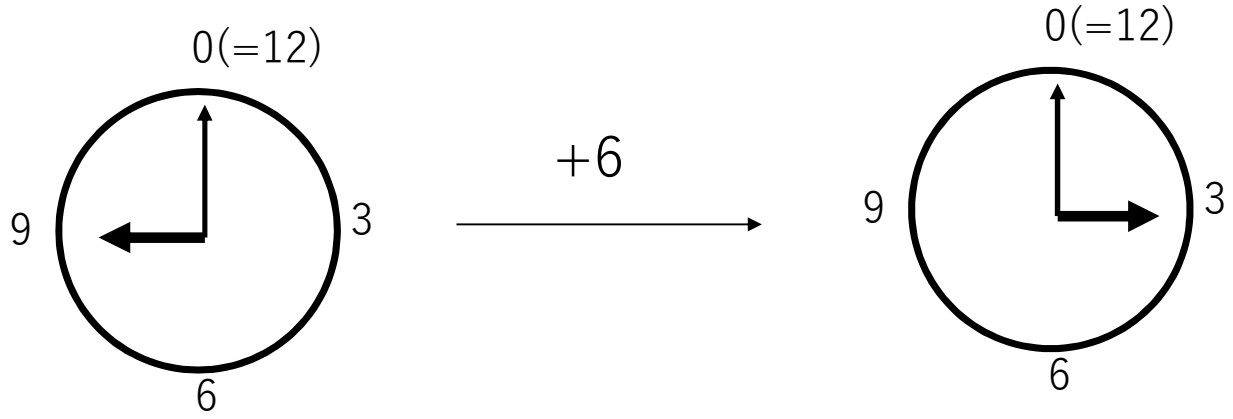
1. 反射律：　任意の$a \in Z$について
$$a = a\ (mod\ n)$$
2. 対称律：
$$a = b\ (mod\ n)\text{ならば}b = a\ (mod\ n)$$
3. 推移律：
$$a = b\ (mod\ n)\text{かつ}b = c\ (mod\ n)\text{ならば}$$
$$a = c\ (mod\ n)$$

- 上記の各性質が成り立つことを前頁の合同式の定義から導いてみよう

合同式の身近な例

- 合同式の身近な例として、時計算（時間の計算）がある。
- 時計の針は12時間で1周する　⇔　mod 12 の合同式

  (例)
  - 9時の6時間後は何時？　9+6=15、15=3(mod12)
  - 2時の3時間前は何時？　2-3=-1、-1=11(mod 12)



記号の読み方と定義

- 合同式 a=b (mod n) は、aイコールb モッドnと読めばよい。modはmoduloの略なので、モジュロと読むこともある。
- 数nのことを、modulus（日本語では、法）と呼ぶ。従って、a=b (mod n)を「aとbはnを法として合同である」と説明することがある。
- 文献によっては、=(イコール)の記号の代わりに、≡を使うこともある。
- aがnで割り切れるとき、n|aと記号 |を使って表現する。（例）3|12、5|(-20)。nとaの左右を間違えないように。

合同式の性質（１）

- [定理１]　a=a’ (mod n)、b=b’(mod n)とする。このとき、
  - a+b = a’+b’ (mod n)
  - a b = a’ b’ (mod n)

  が成り立つ。

  （証明）各自考えてみよ（合同式の定義に戻って考えるとよい）

  （例）n=31とする。a=36, a’=5, b=48, b’=17とする。定理１より、
  36+48=5+17 (mod 31)、また、36*48=5*17 (mod 31)

上記の例で、36+48を計算するより5+17を計算する方が楽であろう。つまり、足し算や掛け算をしてから余りの計算をしても、先に余りの計算をしてから足し算や掛け算をしても同じ結果が得られることを意味している。

合同式の性質（２）

- [定理２] a,b,c:整数、n:正整数とする。
  $a\,c = b\,c \pmod n$
  であるとき、cとnの最大公約数が1であるならば、
  $a = b \pmod n$
  である。
- （例）
  (1) $3=33 \pmod{10}$について、3と10の最大公約数は1であるので、両辺を3で割ると、$1=11 \pmod{10}$となる。
  (2) $4=14 \pmod{10}$であるが、2と10の最大公約数は1でないので、両辺を2で割った、2と7は10を法として合同でない。

合同式で、両辺を同じ数で割るときは、割る数と法の最大公約数が１であることを確認しないといけない

記号の定義（最大公約数）

- aとbの最大公約数を
  gcd(a,b) または、(a,b)
  と表現する
  （(a,b)はベクトルみたいだが、誤解が生じない限りよく使われる）
- 特に、(a,b)=1であるとき、aとbは互いに素である、と言う
- 最大公約数は、a,bが小さい数ならばa,bを素因数分解すると求められる。
  （例）36と48の最大公約数は、
  $36=2\times2\times3\times3$
  $48=2\times2\times2\times2\times3$
  なので、$(36,48)=2\times2\times3=12$となる。

素因数分解による方法はa、bが大きな数になると使えない
（暗号では数百桁の大きさの数を扱う必要があり、この方法は非現実的）

ユークリッドの互除法

- 高速に最大公約数g=(a,b)を求めるアルゴリズム(a>bとする)
  - b|aのとき、明らかにg=bである
  - そうでない時、aをbで割って余りrを求めると（a=bq+r）、g=(b,r) である（なぜそうなるか考えてみよう）
  - つまり、aとbの最大公約数gは、bとrの最大公約数に等しい。r<bであるので、上記を繰り返し行うと、余りは単調減少し、いつかは割り切れる（そのときの除数が最大公約数である）
- ユークリッドの互除法の計算量（繰り返し回数）は、bのビット数（桁数）の定数倍になる。
  →bが1000ビットの数でも高々数千回の繰り返しでOK
  （コンピュータでは一瞬で計算可能）

ユークリッドの互除法の例

- a=1785, b=1122とする。最大公約数g=(a,b)を求める。

1785=1122×1+663　（1785を1122で割って余り663）

1122=663×1+459　（除数1122を余り663で割って余り459）

663=459×1+204

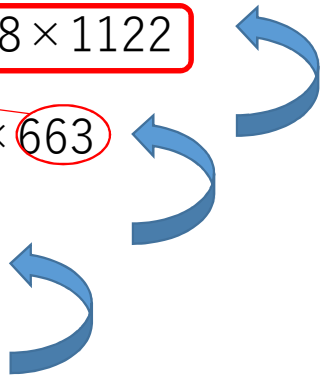
459=204×2+51

204=51×4　（割り切れたので終了、g=51である）

拡張ユークリッドの互除法

- g=(a,b)であるとき、整数s,tを使ってg=sa+tbと表すことができる。
  拡張ユークリッドの互除法は、gと共にs,tを求めるアルゴリズム。
  ユークリッドの互除法の過程を逆にたどると求められる。

1785=1122×1+663	51=3×1122-5×(1785-1122×1)=-5×1785+8×1122
1122=663×1+459	51=-2×663+3×(1122-663×1)=3×1122-5×663
663=459×1+204	51=459-(663-459×1)×2=-2×663+3×459
459=204×2+51	51=459-204×2
204=51×4	以上より、s=-5、t=8が求められる。



拡張ユークリッドアルゴリズム

- 前頁の方法はコンピュータで計算するときは適切でない（途中の計算結果を全て記憶しておかないといけないから）
- 逐次的に計算する方法を考える。i回目の余りをr_iとし、$r_i = s_i a + t_i b$とおく。$r_{i-2} = r_{i-1} q_i + r_i$であるから、

$$\begin{aligned} r_i &= r_{i-2} - r_{i-1} q_i \\ &= (s_{i-2} a + t_{i-2} b) - q_i (s_{i-1} a + t_{i-1} b) \\ &= (s_{i-2} - q_i s_{i-1}) a + (t_{i-2} - q_i t_{i-1}) b \end{aligned}$$

- 上式より、漸化式

$$\begin{aligned} s_i &= s_{i-2} - q_i s_{i-1} \\ t_i &= t_{i-2} - q_i t_{i-1} \end{aligned}$$

- 初期値は、$s_{-2} = 1, s_{-1} = 0, t_{-2} = 0, t_{-1} = 1$
- 各自で、拡張ユークリッドアルゴリズムを実装してみるとよい