数論アルゴリズム入門

横田　壽 [1]

[1] 芝浦工業大学

September 12, 2017

基数

- n を $(d_{k-1}d_{k-2}\ldots d_1d_0)_b$ のように表すとき，b を**基数**という．また，d_i はディジット (桁) といい，0 から $b-1$ の間の整数である．この表記法は

$$n = d_{k-1}b^{k-1} + d_{k-2}b^{k-2} + \cdots + d_1b + d_0$$

を意味している．

例題

$(11001001)_2$ を求めよ．

解 $(11001001)_2 = 1 * 2^7 + 1 * 2^6 + 1 * 2^3 + 1 * 2^0 = 201$

例題

$b = 26$ のとき，数字の 0〜25 がそれぞれ A〜Z を表すとする．このとき，$(\mathrm{BAD})_{26}$ を求めよ．また，$(\mathrm{B.AD})_{26}$ を求めよ．

解 $(\mathrm{BAD})_{26} = 1 * 26^2 + 0 * 26 + 3 = 679$

例題

7 進法で 160 と 199 の乗算を行え.

解 $160 = (316)_7, 199 = (403)_7$ より,

$$
\begin{array}{cccccc}
 & & & 3 & 1 & 6 \\
 & & & 4 & 0 & 3 \\
\hline
 & & 1 & 2 & 5 & 4 \\
1 & 6 & 0 & 3 & 0 & \\
\hline
1 & 6 & 1 & 5 & 5 & 4
\end{array}
$$

例題

$(11001001)_2$ を $(100111)_2$ で割れ, また (HAPPY) を (SAD) で割れ.

解答 $(1001001)_2 \div (100111)_2 = 101 \frac{110}{100111}$

例題

10^6 を 2 進数，7 進数, 26 進数に変換せよ．

解答 $10^6 \div 2 = 0$ より，最下位桁の数字が求まる．またその商 500000 を 2 で割ると最後から 2 番目の桁の数が求まる．以下同様に繰り返すと，$10^6 = (11110100001001000000)_2$
NOTE $b^{k-1} \leq n < b^k$ を満たす整数 n は，b を基数として k 桁を持つ．対数の定義より，

$$\text{桁の数} = [\log_b n] + 1 = [\frac{\log n}{\log b}] + 1$$

NOTE 数 n を 2 進数で表すとき，桁数をビット長という．

例題

次の演算を行え．

$$\begin{array}{cccccccc} & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ + & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}$$

解 この演算を行うには，次のステップを 7 回繰り返す．
1. 上下のビット数を調べる．また上のビットの上に繰り上がりがあるか調べる．
2. 両方のビットが 0 で繰り上がりもない場合，0 を記入し次の桁へ移動
3. 両方のビットが 0 で繰り上がりがある場合，もしくは片方のビットが 0 で他方が 1，かつ繰り上がりがない場合は，1 を記入し次の桁へ移動
4. 片方のビットが 0 で他方が 1，かつ繰り上がりがある場合，もしくは両方のビットが 1 で繰り上がりがない場合は，0 を記入し次の桁に繰り上がりをしてから移動する．
5. 両方のビットが 1 で繰り上がりがある場合，1 を記入し次の桁に繰り上がりをしてから移動する．

この手続きを 1 回行うことをビット演算という．これより，2 つの k ビットの数の加算には k ビット演算が要求される．

O 記法

- 関数 $f : \mathbb{Z}_+ \to \mathbb{Z}_+$ を整関数という.
- $f(n)$ と $g(n)$ が $n \geq n_0$ で $f(n) < Cg(n)$ となる定数 C が存在するとき，$f(n) = O(g(n))$ と表す.
- $g(n)$ として用いる関数は，$f(n)$ より簡単な関数で，$f(n)$ よりあまり早く増加しないを用いる.

例題

$2n^2 + 3n + 4$ を O 記法で表せ.

解 $2n^2 + 3n + 4 = O(n^2)$

例題

$n\sqrt{n} + n^3 + 2n$ を O 記法で表せ.

解 $n\sqrt{n} + n^3 + 2n = O(n^3)$

例題

$\lim_{n\to\infty} \frac{f(n)}{g(n)} =$ 定数 ならば，$f(n) = O(g(n))$ が成り立つことを示せ.

- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ のとき，$f(n) = \circ(g(n))$ と表す.
- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$ のとき，$f(n) \asymp g(n)$ と表す.
- $\lim_{n\to\infty} \frac{n\sqrt{n}+n^3+2n}{n^3} = 1$ より、$n\sqrt{n} + n^3 + 2n = O(n^3)$.
- 素数定理 $\pi(n) \asymp \frac{n}{\log n}$. $\pi(n)$ は n 以下の素数の個数を表す.

Ω 記法

- $f = \Omega(g)$ は $g = O(f)$ を意味する.
- $f = \Theta(g)$ は $f = O(g)$ と $f = \Omega(g)$ を意味する.
- これらの記号は数式の中より数式の内側で用いられることが多い.
- 関数が $n^{O(\log\log n)}$ であるということは，$n \geq n_0$ に対して，関数が $\leq n^{C\log\log n}$ となる定数 C が存在することを意味する.

課題

1. ε が任意の正の数のとき，$\log n = o(n^{\varepsilon})$ と表せることを示せ.
2. 全次数が高々 n の x, y, z の単項式の数を最良の評価となる式で与えよ.
3. 最初の n 個の正整数の平方の和を評価せよ.

- 2. $x^i y^j z^k$ において、$0 \leq i, j, k \leq n$.
- 3. $1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} = O(n^3)$

数の長さ

- 整数の長さ (length) はその整数を 2 進数で表したときのビットの数を意味する.

$$\mathrm{length}(n) = 1 + [\log_2 n]$$

であり，これは $O(\log n)$ で評価できることに注意

例題

長さが k の整数 m は，$2^{k-1} \leq m < 2^k$ を満たすことを示せ.

- 解 長さが k より，整数 m は

$$2^{k-1} \leq m \leq 2^{k-1} + 2^{k-2} + \cdots + 2 + 1 = 2^k - 1 < 2^k$$

を満たす.

例題

次の操作によって求められる長さを求めよ．
それぞれの長さが高々 k の n 個の正整数を

- *1.* 加え合わせる
- *2.* 掛け合わせる

- 1 解 $\dot{2}$ つの数の和の長さは，大きい方の数の長さか大きい数の長さ＋１であることに注意する．それぞれの長さが高々 k ということより，それぞれの数の大きさは 2^k より小さく，2^{k-1} 以上であることがわかる．よって，n 個の数の和の大きさは $n2^k$ より小さいことから，その長さは $O(k + \log(n))$ で評価できる．

- 2 解. 長さ k の数 m の大きさは $2^{k-1} \leq m < 2^k$ を満たす．そこで、m_1 が長さ k をもち，m_2 が長さ k をもつとすると，2 つの不等式

$$2^{k-1} \leq m_1 < 2^k$$

$$2^{k-1} \leq m_2 < 2^k$$

を掛けて，$2^{2k-2} \leq m_1 m_2 < 2^{2k}$ を得る．これから $\mathrm{length}(m_1 m_2)$ は m_1 と m_2 の長さの和 $2k$，または m_1 と m_2 の長さの和-1 に等しい．これより，n 個の積は

$$2^{nk-n} \leq \prod m_i < 2^{nk}$$

したがって，長さは高々 nk である．

例題

$n!$ の長さを求めよ．

解答 $n!$ を構成している n 個の数はすべて $\mathrm{length}(n)$ より短い．そこで、$n! = \Pi m_i$ とすると，$m_i < 2^{\mathrm{length}(n)}$ を満たす．したがって，

$$n! = \Pi m_i < 2^{n\mathrm{length}(n)}$$

これより，$n!$ の長さは $n(\mathrm{length}(n))$ 以下である．したがって，

$$\mathrm{length}(n!) \leq n\mathrm{length}(n) = O(n \log n)$$

例題

次の各々の場合，指示した数の長さを評価せよ．答えは O 記法で表せ．

- *1.* それぞれの長さが高々 k である n 個の数の和
- *2.* $n^4 + 25n^2 + 40$
- *3.* n の k 次多項式 $a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$．ただし，$k$ と a_i は定数
- *4.* ビット数が k 以下の全ての素数の積
- *5.* $(n^2)!$
- *6.* n 番目の *Fibonacci* 数

解答

- 1. それぞれの長さが高々 k ということは、それぞれの数は 2^k より小さい．これらを n 個加えて作った数 m は $n2^k$ より小さいことになる．
- 2. $m = n^4 + 25n^2 + 40$ とすると、$m = O(n^4)$ と表せる．したがって、m の長さは、$\mathrm{length}(m) = O(\log n^4) = O(\log n)$.
- 3. $m = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$ とおくと、$m = O(n^k)$．よって、$\mathrm{length}(m) = O(\log n)$
- 4. ビット数が k 以下の素数 m_i の大きさは、$m_i \leq 2^k$ である．また、大きさ n までの素数の個数は、$\frac{n}{\log n}$ である．したがって、ビット数が k 以下のすべての素数の積 m は $m \leq \prod_{i=1}^{2^k/\log 2^k} 2^k \leq (2^k)^{2^k/\log 2^k}$．したがって、$m$ の長さは、

$$\mathrm{length}(m) = O(\frac{2^k}{\log 2^k} \log 2^k) = O(2^k)$$

- 5. $m = n!$ とおくと、$\mathrm{length}(m) = O(n \log n)$ より、

$$\mathrm{length}(\mathrm{n}^2!) = \mathrm{O}(\mathrm{n}^2 \log(\mathrm{n}^2) = O(n^2 \log n)$$

- 6. Fibonacci 数とは、$F_n = F_{n-1} + F_{n-2}, F_0 = 0, F_1 = 1$ で定義される数である．上式を変形して、

$$F_n - \alpha F_{n-1} = \beta(F_{n-1} - F_{n-2})$$

の形にする．ここで、$\alpha + \beta = 1, \alpha\beta = 1$ に注意すると、$\alpha = \frac{1+\sqrt{5}}{2}, \beta = \frac{1-\sqrt{5}}{2}$．また、$F_n - \alpha F_{n-1}$ は公比 β の等比数列となるので、

$$F_n - \alpha F_{n-1} = \beta^{n-2}(F_2 - \alpha F_1) = \beta^{n-1}$$

同様にして、

$$F_n - \beta F_{n-1} = \alpha^{n-1}$$

この 2 式から F_{n-1} を消去し、F_n について解くと、

$$F_n = \frac{\alpha^n - \beta^n}{\alpha - \beta} = \frac{1}{\sqrt{5}} \left\{ (\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n \right\}$$

課題

- *7.* n 番目の *Fibonacci* 数の長さと漸近的に等しいような関数 $g(n)$ を見出せ.
- *8. Stiring* の公式を使い，$n!$ の長さと漸近的に等しいような関数 $g(n)$ を見出せ.
- *9.* 文字 $A, B, C \ldots, Z$ が *26* 進文字として使われているとする．このとき

$$(DOG)_{26}^{(CAT)_{26}}$$

の 2 進数での長さは，ほぼ次のどれに等しいか.

50 150 500 1500 5000 15000 50000 150000 500000 1500000

ビット演算

- 数 n を 2 進数で表すとき，$\mathrm{length}(n)$ をビット長という．

例題

次の演算を行え．

$$\begin{array}{cccccccc} & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ + & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}$$

- 一桁ごとの演算をビット演算という．
  2 つの数を足すのに必要な時間（ビット演算の数) は 2 つの数の長さの最大値に等しい．

$$\mathrm{Times}(k\text{ビット} + \ell\text{ビット}) = \max(k, \ell)$$

- 2 つの数 m と n で表すときは，

$$\mathrm{Times}(m + n) = \max(\log m, \log n)$$

掛け算のビット演算

k ビットの整数に ℓ ビットの整数をかけることを考える．

$$
\begin{array}{cccccccccc}
 & & & & 1 & 1 & 1 & 0 & 1 \\
\times & & & & & 1 & 1 & 0 & 1 \\
\hline
 & & & & 1 & 1 & 1 & 0 & 1 \\
 & & & 1 & 1 & 1 & 0 & 1 & 0 & \\
 & & 1 & 1 & 1 & 0 & 1 & & & \\
\hline
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0
\end{array}
$$

高々ℓ 個の行を得る．ここで，各行はある数だけ左にシフトしただけである．シフトはビット演算ではなく管理上の手続きとして数えられるので，時間評価では無視されることに注意する．その結果，各行の足し算は k 個のビット演算しか要しない．したがって，ビット演算の総数は

(ℓ個の加算) $\times$ (加算あたり k 個のビット演算) $= k\ell$ より少ない．

$$\mathrm{Times}(m \times n) = O(\log m \log n)$$

割り算のビット演算

- $(11001001)_2$ を $(100111)_2$ で割ることを考える．

$$
\begin{array}{cccccccccc}
 & & & & & & & 1 & 0 & 1 \\
\cline{8-10}
1 & 0 & 0 & 1 & 1 & 1 &) & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
 & & & & & & & 1 & 0 & 0 & 1 & 1 & 1 & & \\
\cline{8-14}
 & & & & & & & & & 1 & 0 & 1 & 1 & 0 & 1 \\
 & & & & & & & & & 1 & 0 & 0 & 1 & 1 & 1 \\
\cline{8-14}
 & & & & & & & & & & & & 1 & 1 & 0
\end{array}
$$

- $k > \ell$ とするとき，k ビット整数を ℓ ビット整数で割った時の時間評価を求める．
  商と余りを得るには最大で $k - \ell + 1$ 回の減算が必要である．
  そして各減算は，ℓ ビット演算を必要とする．したがって，
  時間評価は $(k - \ell + 1)\ell \leq k\ell$

$$k \text{ ビットを } \ell \text{ビットで割るのにかかる時間} \leq k\ell$$

アルゴリズム

- 計算をするために明示的な段階的手続は，**アルゴリズム**とよばれる．
- 徐算は乗算と同じように解析でき，k ビット整数を ℓ ビット整数で割るとき，商と余りを得るには $O(\ell(k-\ell+1))$ 個のビット演算を要する．

例題

- k ビット整数 n を基数 10 の表現に変換するのに要する時間を評価せよ．

- 解答
  k ビット整数を 10 進表現に変換することを考える．まず，k ビット整数を n とし，n を $10 = (1010)_2$ で割ると、余りは $0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001$ のどれかとなる．次に、商を $(1010)_2$ で割る．これを繰り返すと、$O(\frac{\log n}{\log 10}) = O(k)$ 回の割り算が必要となる．次に、1 回の割り算において k ビットを 4 ビットで割るので、$O(4k)$ 個のビット演算をようする．したがって，ビット演算数の合計は $O(k^2) = O(\log^2 n)$ となる．

例題

- *1.* k ビット整数 n を b 進数表現に変換するのに必要な時間を評価する.
- *2.* $n!$ を計算するのに必要な時間を評価せよ.

  - 1. n を ℓ ビット整数 b で割る.
    割り算は $[\log_b n] + 1 = O(\log_2 n / \log_2 b) = O(k/\ell)$ 回必要で,
    1 回の割り算で減算を $O(\ell k)$ 回行う必要がある. したがって,
    ビット演算数の合計は $O(k^2) = O(\log^2 n)$ となる.

- 2. $2 \cdot 3 \cdots k-1$ と k をかけるのに必要なビット演算を考える．$(k-1)!$ を k にかけるには，$(k-1)!$ の値を求めるために，$k-2$ 回の掛け算を必要とする．つまり最悪 n 回の掛け算を必要とする．次に，$(k-1)!$ を 2 進数で表すのに，$\mathrm{length}(n!) = O(n \log n)$ の演算が必要である．そこで，$(k-1)!$ と k をかけるには $O((k-1)!k)$ 回．つまり，$O(n \log n \times \log n) = O(n \log^2 n)$ の演算が必要となる．これを $k = 1$ から $k = n$ まで行うので，演算数の合計は $O(n^2 \log^2 n)$ となる．

課題

- O 記法をつかって，3^n を 2 進で計算するのに必要なビット演算の数を n の簡単な関数で評価せよ．
- 同じことを n^n について行え．
- N^n を計算するのに必要なビット演算の数を n と N の簡単な関数で評価せよ．
- 次の 2 進数

$$1011011001011^{1000111}$$

の正確な値を計算するのに必要なビット演算数は，ほぼ次のどれに等しいか．

$$100, 1000, 10000, 100000, 10^6, 10^{10}, 10^{25}, 10^{75}$$

Euclid

例題

2つの整数 $a > b > 0$ が与えられるとき，a と b の最大公約数 d を計算することと，方程式 $au + bv = d$ を整数 u と v について解くことは，ともに時間 $O(\log a \log b)$ でできる．

- 解答 ユークリッドアルゴリズムより

$$\begin{aligned} a &= q_0 b + r_1 & 0 < r_1 < b \\ b &= q_1 r_1 + r_2 & 0 < r_2 < r_1 \\ &\vdots \\ r_{l-1} &= q_l r_l + r_{l+1} & 0 < r_{l+1} < r_l \\ r_l &= q_{l+1} r_{l+1} \end{aligned}$$

これより，$d = r_{l+1}$ を得る．

最大公約数

逆向きに戻ると，

$$\begin{aligned} d &= r_{\ell+1} = r_{\ell-1} - q_\ell r_\ell \\ &= v_\ell r_\ell + u_{\ell-1} r_{\ell-1} && v_\ell = q_\ell, u_{\ell-1} = 1 \\ &= v_{\ell-1} r_{\ell-1} + u_{\ell-2} r_{\ell-2} && v_{l-1} = u_{\ell-1} - q_{\ell-1} v_\ell, u_{\ell-2} v_\ell \\ &\vdots \\ &= v_1 r_1 + u_0 b \\ &= vb + ua \end{aligned}$$

除算 $a = q_0 b + r_1$ のビット演算数が高々$\mathrm{length}(b) \cdot \mathrm{length}(q_0)$ であることを用いる．すると，すべての除算の総時間は

$$O(\log b(\log q_0 + \log q_1 + \cdots \log q_{\ell+1})) = O((\log b)(\log \Pi q_i))$$

である．しかし，$\Pi q_i \leq a$ であるから，限界は $O(\log b \log a)$ である

合同式

定義

3つの整数 a, b および m が与えられているときに，差 $a - b$ が m で割り切れるならば,「m を法として，a は b と合同である」といい，$a \equiv b \bmod m$ と表す.

課題

$$ax \equiv 1 \pmod{m}$$

は $|a| < m$ および $\gcd(a, m) = 1$ として，x について時間 $O((\log m)^2)$ で解けることを示せ.

定理

数 a が $ab \equiv 1 \bmod m$ となる b を持つことと，$\gcd(a, m) = 1$ となることは同値である．

$ab \equiv 1 \bmod m$ となる b が存在し、$\gcd(a, m) = d$ とすると、
$ab - 1 = mk$. ここで、$d|a, a|m$ より、$d = 1$.
$\gcd(a, m) = 1$ とし、$a, 2a, 3a, \ldots, ma$ を考える．ここで、
$i \neq j, 1 \leq i < j \leq m$ ならば　$ja \neq ka \pmod m$ であることを示す．
もしかりに、$ja \equiv ka \pmod m$ とすると、$(j - k)a = ms$. ここで、
$(a, m) = 1$ より、$m|i - j$ となり矛盾である．

例題

例題 $160^{-1} \bmod 841$ を求めよ．

$160x = 1 \bmod 841$ となる x を求める．そこで、ユークリッドの互除法を用いて $\gcd(160, 841)$ を求め、$1 = 160x + 841y$ を解く．

Fermat の小定理

定理

素数 p を考える．任意の整数 a に対して，$a^p \equiv a \bmod p$ が成り立つ．また，$(a, p) = 1$ のとき，$a^{p-1} \equiv 1 \bmod p$ が成り立つ．

$\{a, 2a, 3a, \ldots, (p-1)a\}$ は全体として、$\{1, 2, 3, \ldots, p-1\}$ と合同である．すでに示したように、$ja \neq ka \pmod p$. そこで、すべてを掛け合わせると

$$(p-1)!a^{p-1} \equiv (p-1)! \pmod p$$

したがって、$a^{p-1} \equiv 1 \pmod p$.

例題

例題 7 を基数として，$2^{1000000}$ の最後の桁の値を求めよ．

$2^{7-1} \equiv 1 \pmod 7$ を用いる．

繰り返し 2 乗法を用いた法べキ乗計算

定理

m は k ビットの自然数，N は ℓ ビットの自然数，そして $|b| < m$ とする．m を法とする b^N の最小非負剰余を，時間 $O(k^2\ell)$ で見出す方法を示せ．

N を 2 進数で表す．

問

$n^5 - n$ は常に 30 で割り切れることを証明せよ．

$n^5 - n = n(n^4 - 1) = n(n^2 - 1)(n^2 + 1) = n(n+1)(n-1)(n^2+1)$.
30 で割り切れるとは $2, 3, 5$ で割り切れることを示せばよい．連続する 2 つの整数の積は 2 で割り切れる．連続する 3 つの数の積は 3 で割り切れる．そこで、$n^5 - n$ は 5 で割り切れることを示す．

多項式時間から指数時間まで

多項式時間

2 進数で表された全長が高々 k の整数について，そのアルゴリズムを実行するのに必要なビット演算数が $O(k^d)$ となる整数 d が存在する．

- 通常の算術演算 $+, -, \times, \div$ は，多項式時間アルゴリズム

指数時間

2 進数で表された全長が高々 k の整数について，そのアルゴリズムを実行するのに必要なビット演算数が $O(e^{ck})$ の形の時間評価をもつ．ただし，c は定数である．ここで，k は整数の 2 進数としての全長である．

多項式時間から指数時間まで

- 整数 n を素因数分解するとき，$2, 3, 4, 5, 6 \ldots$ で割っていくと，このアルゴリズムは時間 $O(n^{1/2+\varepsilon})$ かかる．これより，$e^{ck} = n^{1/2+\varepsilon}$ とおくと，$k \approx \log_2 n$ となり，指数時間となる．
- 多項式時間と指数時間の間の領域で分類する有用な方法を紹介する．

定義

γ を 0 と 1 の間の実数，$c > 0$ とする．このとき，

$$L_n(\gamma; c) = O(e^{c(\log n)^{\gamma}(\log\log n)^{1-\gamma}})$$

と定める．

- $L_n(1; c) = O(e^{c \log n}) = O(n^c)$,
  $L_n(0; c) = O(e^{c \log\log n}) = O((\log n)^c)$

素因数分解

問

8091 を素因数分解せよ.

解：(Carl Pomerance)

$$\begin{aligned} 8051 &= 8100 - 49 = 90^2 - 7^2 \\ &= (90-7)(90+7) = 83 \cdot 97 \end{aligned}$$

これより，n の素因数分解には

$$x^2 - n = y^2$$

となる x, y を探せばよい．つまり，$n = 8051$ の時には，$x = 90, 90 \pm 1, 90 \pm 2, \ldots$ として，$Q(x) = x^2 - n$ が平方数になるものを探す．この方法を **Fermat** の平方差法という．

2次ふるい法

M. Kraitchik は Fermat 法を改良し，$n|x^2 - y^2 = (x+y)(x-y)$ を満たす n の因数を見つける方法を考え出した．もし，$n \nmid |x+y$ または $n \nmid x-y$ ならば，$\gcd(x-y.n) > 1$ となる．ここで，ユークリッドアルゴリズムを用いて $\gcd(n, x-y)$ を求めると，n の自明でない因数が求まる．

$n = 3937$ の場合について考える．

$$
\begin{aligned}
63^2 - n &= 32 &&= 2^5 \\
64^2 - n &= 159 &&= 3 \cdot 53 \\
65^2 - n &= 288 &&= 2^5 \cdot 3^2 \\
66^2 - n &= 419 &&= 419 \\
67^2 - n &= 552 &&= 2^3 \cdot 3 \cdot 23
\end{aligned}
$$

これより，$(63 \cdot 65)^2 \equiv (2^5 \cdot 3)^2 \bmod n$ を得るので，$\gcd(63 \cdot 65 - 2^5 \cdot 3, 3937)$ をユークリッドアルゴリズムで求めると 31 となる．したがって，

$$3937 = 31 \cdot 127$$

プロジェクト

- 因数分解の方法はここで紹介したもの以外に，連分数法と数体アルゴリズムが知られている．
- 連分数法による因数分解について調べ発表を行う．

クラス P,NP, および NP 完全

- クラス P（Polynomial) に属するとは，ある問題が多項式時間のアルゴリズムを持つときである．
- クラス NP(Non-deterministic Polynomial) に属するとは，ある決定問題が非決定性多項式時間のアルゴリズムを持つときである．
- 未解決問題　クラス P = クラス NP
- クラス NP 完全（NP-complete) 問題とは，NP に属する問題のうち，解くのが最も難しいものをいう．NP のすべての問題から多項式時間帰着可能な問題

暗号

- Alice 重要な文書を送る人
- Bob 重要な文書を受け取る人
- Alice が重要な文書をインターネットを介して Bob に送る．このとき，Alice の文書は 26 進数の数字に変換された後 10 進数で表される．したがって，送るのは数字の羅列である．
- Alice が送った数字の羅列を傍受したものは，26 進数表示のアルファベットに戻すことで Alice の送った文書の中身を知ることができる．
- ここでどうすれば途中で盗聴されても中身を知られないで済むかを考える．

秘諾方法

- ステガノグラフィー
  - データ隠蔽技術の一つ．在原業平「からころもきつつなれにしつましあればはるばるきぬるたびをしぞおもう」
  - 画像に著作権者などを埋め込む電子透かし (digital watermark)
- コード　単語やフレーズを事前に決めておいた言葉・記号で置き換える．
- サイファ　通信文を意味とは関係なく，事前に決めたアルゴリズムにしたがって，文字やビットごとに置換や転置をおこなうことで，読めない文に変換する．

共有鍵暗号

- Alice と Bob はこっそり会って，お互いしか知らない秘密鍵 (secret key) となる数字を決める．
- Alice から Bob に重要な文書を送るとき，平文（plaintext) をこの秘密鍵を使って暗号文 (ciphertext) に暗号化する，
- 文書を受け取った Bob は秘密鍵を用いて，暗号文を平文に戻す．
- この方法の問題点をあげ，改良方法について考えよ．

公開鍵暗号

- Diffie と Hellman は１９７５年に，鍵を２つ用意することを考えた．１つは，公開鍵とよばれ誰でも見ることができるもの，もう一つは秘密鍵である．ここで，重要なことは秘密鍵があれば公開鍵が作れるが，公開鍵があっても秘密鍵は容易に作れないことである．
- Bob は秘密鍵から作った公開鍵をインターネット上に公開しておく．
- Alice は Bob の公開鍵を用いて，平文を暗号文に変換し送る．
- 暗号文を受け取った Bob は秘密鍵を使って復号化し，暗号文を平文に直す．
- Eve が盗聴に成功したとしても，公開鍵から秘密鍵は簡単に作れないので，秘密情報は守れる．

RSA 暗号

- 2 つの大きな素数を用意する．ここでは例なので素数を $p = 43, q = 53$ とする．
- ここで，$n = 43 \times 53 = 2279$ を公開鍵とする．
- (p-1) と (q-1) の積を計算する．L = 42*52 = 2184
- L=2184 と互いに素で L より小さい任意の素数 e を選ぶ．ここでは $e = 101$ を選び公開鍵とする．
- 次に，$e \times d \equiv 1 \bmod L$ となる正の整数 d を計算する．これが秘密鍵となる．
- d は 2184 を法とする $e = 101$ の逆元より，Euclid の互除法を用いると，$d = 173$ となる．

暗号化・復号化

- 平文 cat を暗号化することを考える.
- cat を 26 進表示で表すと 2,0,19 これを 10 進数に直すと $2 \times 26^2 + 0 + 19 \times 26^0 = 1371$
- 1371 を公開鍵 101 と 2279 を使って暗号化すると,

$$1371^{101} \equiv 1520 \pmod{2279}$$

  となり，1520 が暗号文となる.
- 次に 1520 を復号化するには秘密鍵 d = 173 を用いて, $1520^{173} \mod 2279$ を計算すればよい.

期末課題

- 平文・暗号文に用いるアルファベットを 40 文字とする．$A-Z$ は $0-25$ に対応し，空白 $=26$, . $=27$, ? $=28$, \$ $=29$ とし，数字 $0-9$ は $30-39$ に対応する．また，平文は 2 文字組で，暗号文は 3 文字組で表す ($k=2, l=3$ とし，n は $40^2 < n < 40^3$ の範囲とする）．
- RSA 暗号プログラムを改良し，“SEND \$7500” というメッセージを，暗号化鍵 $(n, e) = (2047, 179)$ を持つ相手に送信せよ．
- n を因数分解し，復号化鍵 (n, d) を求めて，この暗号を破れ．
- n を因数分解しなくても，手早く復号化鍵を求めることができることを説明せよ．

離散対数暗号

- RSA 暗号は，おおきな 2 つの素数を生成しそれらを掛け合わすことで n を生成することが，n から 2 つの素数をもとめるよりもずっと簡単であるという事実を基礎とすることを前節で述べた．" 落とし戸" や" 一方向" といった性質をもつものが，数論の他の基本的な計算処理にも存在する．そこで，ここでは有限体におけるベキ乗計算を考える．
- 定義　G を有限群，b と b のベキ乗である y を G の要素とすると，b に対する y の**離散対数**とは，$b^x = y$ であるような任意の整数 x のことである．
- 例　$G = \boldsymbol{F}_{19}^* = (\boldsymbol{Z}/19\boldsymbol{Z})^*$ とし，b を生成元 2 とすると，底 2 に対する 7 の離散対数は 6 である．
  2 が生成元とは，2,4,8,16,13,7,14,9,18,17,15,11,3,6,12,5,10,1 のように，$\boldsymbol{F}_{19}$ のすべての元を生成するからである．そして，$2^6 = 64 = 7 (\mathrm{mod} 19)$

離散対数暗号

- 例 2 $X^2 - X - 1$ の根 α を $\boldsymbol{F}_9^*$ で考えると，底 α に対する -1 の離散対数は 4 である．
  まず，$\boldsymbol{F}_9$ は $\boldsymbol{F}_{3^2}$ より，$p = 3$ は $\boldsymbol{F}_9$ の標数である．つまり，$1 + 1 + 1 = 0$ になる．次に，α を $X^2 - X - 1$ の根とすると，

$$\begin{aligned} \alpha^2 &= \alpha + 1 \\ \alpha^3 &= \alpha^2 + \alpha = 2\alpha + 1 = -\alpha + 1 \\ \alpha^4 &= -\alpha^2 + \alpha = -1 \end{aligned}$$

  したがって，$\alpha^4 = -1$ となり，α に対する -1 の離散対数は 4 である．
- 離散対数問題とは，b^x は高速に計算できるが，$b^x = y$ の b と y が与えられているとき，どのように x を高速に求めるかである．

Diffie-Hellman 鍵交換システム

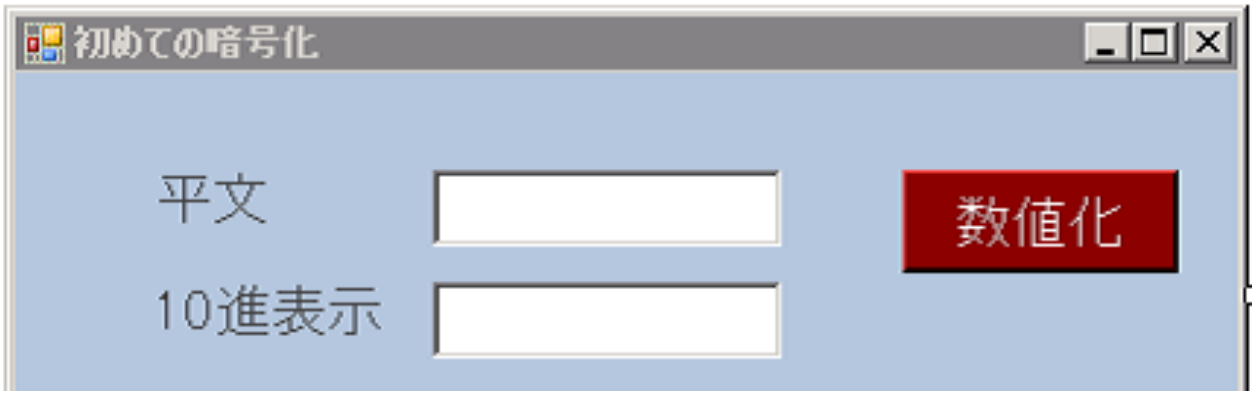
- 大きな有限体 $\boldsymbol{F}_q$ の中からランダムに要素を生成する Diffie-Hellman 法を紹介する.
- q は公開され，すべての人が，鍵がどの有限体に属しているのか知っていると仮定する．$\boldsymbol{F}_q$ の要素 g を固定し，これも公開する.
- A(Aida) と B(Bernardo) が $\boldsymbol{F}_q^*$ の乱数である鍵に合意し，それを用いて平文を暗号化するものとする.
- Aida は 1 から $q-1$ の範囲で秘密の乱数 a を選び，$g^a \in \boldsymbol{F}_q$ を計算し公開する.
- Bernardo も同じように乱数 b を選び，$g^b \in \boldsymbol{F}_q$ を計算し公開する.

Diffie-Hellman 予想

- g^a と g^b だけしか知らないなら，g^{ab} を求めるのは計算上困難である．
- 離散対数が計算可能ならば，g^a と g^b より，a, b が求まる．したがって，g^{ab} も求まり，Diffie-Hellman 予想は成り立たない．
- この逆も成り立つと予想する人もいるが，これはまだ未解決問題である．つまり，a と b を求めずに，g^a と g^b から直接 g^{ab} を求める方法をだれも思いつかないということである．

Visual C# 入門

- どんな手段でもよいから Visual C#が使える状態にする.
- Visual C#を起動し，Form アプリケーションを開く.
- 次のフォームをツールボックスから label, textbox, button を使って作成する.



ASCII コード

- textBox1 に ABC と記述し、ボタンを押したら textbox2 に ABC の ASCII コードを表示せを表示するプログラムを作成せよ.
  1. textBox1 に記述した文字を取り出すには、string str = textBox1.Text;
  2. str の一文字ずつ ASCII コードに変換するには (int)str[j]
  3. str の一文字ずつ ASCII コードに変換し、textBox2 に表示するには、str の長さ len を str.Length; で計算し、ループを用いて一文字ずつ変換する.
  4. for(int j=0;j<len;j++) textBox2.Text += (int)str[j];

文字および文字列の扱い

- textBox1 に SEND $5000 と記述し、ボタンを押したら textbox2 に組文字数 2 の同義数を表示するプログラムを作成せよ.
  1. ここでは、アルファベット、記号、数字などが混在しているので、正確に見分けるためのコマンドが必要である.
  2. char.Isletter( ) で文字かどうかを判断でき, 先程の練習問題より 65 を引けば条件を満たす. char.IsLower( ) で小文字、char.IsUpper( ) で大文字
  3. char.IsNumber( ) または char.IsDigit( ) で数字かどうかを判断でき、ASCII コードから 18 を引けばよい.
  4. 2 文字組で処理をするので、全体の文字数% 2 でそれぞれの組の文字を同義数に変換する.
  5. 文字 S は 18,E は 4 として扱うので、SE の同義数は 18*40+4 で求まる.

ファイルの操作

- textBox3 にファイルから読み込んだ公開鍵を表示するプログラムを作成せよ.
  1. ファイルを操作するために、ツールボックスから openFileDialog1 を取ってくる.
  2. textBox3 の横にファイルボタンを用意する.
  3. 開いたファイルの中身を読み込むために、string fileText を用意する.
  4. if(openFileDialog1.ShowDialog( ) == DialogResult.OK) fileText = path.GetFullPath(openFileDialog1.FileName);
  5. StreamReader reader = new StreamReader(fileText); textBox3.Text = reader.ReadToEnd();
  6. reader.close();

暗号化

- 同義数 1371 を公開鍵 101 と 2279 を使って暗号化すると，

$$1371^{101} \equiv 1520 \pmod{2279}$$

  となり，1520 が暗号文の同義数となる．
- int r = 1; int equiv = 1371; int pubkey = 2279;
- r = r*(int.Parse(equiv));
- r = r%pubkey

button3_Click

- 暗号化鍵を読み込む
- 同義数を読み込む
- 同義数ごとに暗号数に変換
  1. string[] st = textBox2.Text.Split(',');
  2. int len = st.Length;
  3. for(int j = 0; j < len; j++)
  4. int r = 1;
  5. for(int i = 0; i < power; i++)
  6. r = r*(int.Parse(同義数)) % 公開鍵

暗号数から暗号文

- 暗号数 1520 を文に直すには、平文から同義数を求める操作と逆の操作を行う.
- 3 文字組の場合
  1. for(int i=0; i < len; i++)
  2. for (int j=0; j < 3; j++)
  3. 1520 / Math.Pow(40,2-j) により、i=0 のとき $1520/40^2$ の値は 0 となり A
  4. r % Math.Pow(40,2-j) により、r は 1520
  5. 次のループで 1520/40 の値は 38

button4_Click

- string[] s = textBox4.Text.Split(' ');
- int len = s.Length;
- for(int i = 0; i < len ; i++)
- for (int j=0; j < 3; j++)
- cr = int.Parse(s[i]) / Math.Pow(40,2-j);
- rem = int.Parse(s[i]) % Math.Pow(40,2-j);
- s[i] = rem.ToString();

RSA 暗号の基本的考え方

- $a^{179} \mod 2047 = 1906$
- これより a を求める.
- Euler の公式より $a^{\phi(n)} \equiv 1 \mod n$
- $179 * d \mod \phi(n) = 1$ となる d が求まれば、
- $a = a^{179*d} = 1906^d \mod 2047$ により a が求まる.

$179 * d \mod (\phi(n)) = 1$

- $179 * d \mod (1936) = 1$ を解く.
- 拡張ユークリッドの互除法を用いる.
- while(179*d % 1936 != 1) で d を求める.

$1906^d \mod 2047$

- 上で 1906 の逆元が求まったので、それを用いて
- int r = 1;
- for(int i=0; i < d; i++)
- $r = 1906 * r\%2047;$
- これで $1906^d \mod 2047$ が求まる.
- 同じことをそれぞれの暗号数で繰り返す.

button5_Click

- prime[0] = 23;
- prime[1] = 89;
- int phin = (prime[0]-1)*(prime[1]-1);
- string[] s = textBox4.Text.Split(' ');
- int len = s.Length;
- 逆元を求める.

button5_Click

- int inv = inverse(pubkey[0],pubkey[1]);
- for(int j=0; j < len; j++)
- int r = 1;
- for(int i=0; i < inv; i++)
- r = r*int.Parse(s[j]) % pubkey[0];

複合化

- 暗号文を直接複合化するか、暗号数を用いて複合化する.
- ここでは暗号数を用いて複合化を行う.
- すでに button5_Click で解読数ができているので、
- 解読数を文に直せば完成である.