数学実習 2011 年度第 4 回
担当：中口悦史

今週の講義メモ：整数の問題，アルゴリズム入門

整数と演算

数の集合

自然数　$1, 2, 3, \ldots$
1 は自然数，n が自然数なら $n+1$ も自然数，という数学的帰納法で定義される。

整数　$0, \pm1, \pm2, \pm3, \ldots$
任意の整数 n は，適当な 2 個の自然数 m, ℓ を使って，$n = m - \ell$ と表される。

整数の演算

加算　整数の和も整数。自然数の和は自然数。

乗算　整数の積も整数。自然数の積は自然数。

減算　整数の差も整数。自然数の差は自然数とは限らない。

除算　整数 a と，0 でない整数 b に対して，商 $a \text{ div } b = (|a/b| \text{ を超えない最大の整数}) \times (a \text{ の符号}) \times (b \text{ の符号})$
　注意：　a あるいは b が負の整数の場合は，Pascal の `div` と `mod` が一般的な定義と異なる点がある。

剰余　整数 a と，0 でない整数 b に対して，剰余 $a \mod b = a - b \times (a \text{ div } b)$

整除可能性，約数と倍数　以下，正または非負の整数の場合のみ扱う。

整除可能性　非負整数 a と正の整数 b について $a \mod b = 0$ であるとき，b は a を割り切る，a は b で割り切れるあるいは整除可能といい，$b \setminus a$ と書く。a は b で割り切れない，つまり $a \mod b \neq 0$ であるときは $b \not\setminus a$ と書く。
b は a を割り切るとき，b は a の約数である，a は b の倍数であるともいう。

- 任意の非負整数 a に対して $1 \setminus a$, $a \setminus a$。
- a, b が正の整数のとき，$b \setminus a$ かつ $a \setminus b$ なら，$a = b$。

素数と合成数　1 とそれ自身以外に約数を持たない正の整数を，素数という。1 とそれ自身以外に 1 個以上の約数を持つ正の整数を，合成数という。

- 1 は素数でも合成数でもない。
- 1 より大きな整数は，素数か合成数かのどちらかである。
- 1 より大きな整数 n は，2 以上 $\sqrt{n}$ 以下のいずれの整数でも割り切れなければ，素数である。（**要確認**）
- 合成数は，素数の積として表される。その表し方は，素数を小さい順に並べて書く方法に限れば，一通りに限る。（算術の基本定理）

公約数　非負整数 a と b の共通の約数，すなわち $d \setminus a$ かつ $d \setminus b$ であるような正の整数 d を，a と b の公約数という。

互いに素　非負整数 a と b の公約数が 1 以外にないとき，a と b は互いに素であるといい，$a \perp b$ と表す。

最大公約数　非負整数 a と b の公約数の中で最大のものを，a と b の最大公約数という。これを $\gcd(a,b)$ と書く。
$\gcd(a,b) = \max\{d;\ d$ は正の整数, $d \setminus a,\ d \setminus b\}$ (注意：gcd という関数は Pascal には用意されていません。)

- 最小の公約数は必ず 1。
- a と b が互いに素，つまり $a \perp b$ のとき，$\gcd(a,b) = 1$。
- $(a \text{ div } \gcd(a,b)) \perp (b \text{ div } \gcd(a,b))$。
- $c \setminus a$ かつ $c \setminus b$ なら $c \setminus \gcd(a,b)$，つまり任意の公約数は最大公約数の約数。

公倍数　正の整数 a と b の共通の倍数，すなわち $a \setminus m$ かつ $b \setminus m$ であるような非負整数 m を，a と b の公倍数という。

最小公倍数　正の整数 a と b の公倍数の中で最小のものを，a と b の最小公倍数という。これを $\mathrm{lcm}(a,b)$ と書く。
$\mathrm{lcm}(a,b) = \min\{m;\ m$ は正の整数, $a \setminus m,\ b \setminus m\}$

- 最大の公約数は存在しない。(敢えて書くなら ∞)
- a と b が互いに素，つまり $a \perp b$ のとき，$\mathrm{lcm}(a,b) = ab$。
- $(\mathrm{lcm}(a,b) \text{ div } a) \perp (\mathrm{lcm}(a,b) \text{ div } b)$。
- $a \setminus m$ かつ $b \setminus m$ なら $\mathrm{lcm}(a,b) \setminus m$，つまり任意の公倍数は最小公倍数の倍数。
- $\mathrm{lcm}(a,b)\gcd(a,b) = ab$。
- $\mathrm{lcm}(a,b) = \gcd(a,b) \times (a \text{ div } \gcd(a,b)) \times (b \text{ div } \gcd(a,b))$。

ユークリッドの互除法　最大公約数に関する公式

- $a > 0$ のとき，$\gcd(a,0) = a$。($\gcd(0,0)$ は未定義)
- $a \geqq b > 0$ のとき，$\gcd(a,b) = \gcd(b, a \mod b)$。

に基づいて，2 個の非負整数 a と b の最大公約数 $\gcd(a,b)$ を求めるアルゴリズム。

1. $a < b$ なら，a の値と b の値を入れ替える： $r := a;\ a; = b;\ b := r$
2. 以下の処理を $b = 0$ となるまで繰返す。
   (a) $r := a \mod b;\ a := b;\ b := r$ { 公式 $\gcd(a,b) = \gcd(b,r)$}
3. a に保存されている値が，求める最大公約数。終了。

エラトステネスのふるい　自然数 n 以下の素数の一覧を表示するアルゴリズム。

1. すべての $i := 2, \ldots, n$ について $a_i := 0,\ b_i := 0$ とする。 { 初期化 }
2. 以下の処理を $i := 2$ から n まで繰返す。
   (a) $a_i = 0$ なら以下の処理をする。 {i は合成数ではない }
      i. $b_i := 1$ {i は素数 }
      ii. $j := 2i$
      iii. 以下の処理を $j \leqq n$ となっている間繰返す。
         A. $a_j := 1;\ j := j + i$ {j は合成数 }
3. $b_2, \ldots, b_n$ のうち $b_i = 1$ となっている i はすべて素数。そのような i をすべて出力して終了。

二項係数 ${}_nC_r$

定義　${}_nC_r = \dfrac{n!}{r!(n-r)!} = \dfrac{(n-r+1)\times(n-r+2)\times\cdots\times n}{1\times 2\times\cdots\times r} = \dfrac{(r+1)\times(r+2)\times\cdots\times n}{1\times 2\times\cdots\times(n-r)}$
ただし $0! = 1$。$r < 0$ または $n < r$ のときは ${}_nC_r = 0$ と約束しておく。

公式

- $n \geqq 0$ のとき，${}_nC_0 = {}_nC_n = 1$
- $n \geqq r \geqq 0$ のとき，${}_nC_{n-r} = {}_nC_r$ （対称性）
- $n \geqq 2, n-1 \geqq r \geqq 1$ のとき，${}_nC_r = {}_{n-1}C_r + {}_{n-1}C_{r-1}$ （加法定理）
- $n \geqq 1, n \geqq r \geqq 1$ のとき，$r\ {}_nC_r = n\ {}_{n-1}C_{r-1}$
- $n \geqq 1, n-1 \geqq r \geqq 0$ のとき，$(n-r)\ {}_nC_r = n\ {}_{n-1}C_r$
- $n \geqq s \geqq r \geqq 0$ のとき，${}_nC_r\ {}_{n-r}C_{s-r} = {}_nC_s\ {}_sC_r$
- $n \geqq r \geqq 0$ のとき，${}_nC_r = \sum_{s=0}^{n-1} {}_sC_{r-1} = \sum_{s=0}^{n-1} {}_{n-r-1+s}C_s$ （和公式）
- $n \geqq 0$ のとき，$\sum_{r=0}^{n} {}_nC_r = 2^n$, $\sum_{r=0}^{n} (-1)^r\ {}_nC_r = 0$ （二項定理より）

Pascal の三角形（または Tartaglia の三角形）　二項係数の九九の表

```
nCr|    0    1    2    3    4    5    6    7    8    9   10   11   12   13
---------------------------------------------------------------------------
  0|    1
  1|    1    1
  2|    1    2    1
  3|    1    3    3    1
  4|    1    4    6    4    1
  5|    1    5   10   10    5    1
  6|    1    6   15   20   15    6    1
  7|    1    7   21   35   35   21    7    1
  8|    1    8   28   56   70   56   28    8    1
  9|    1    9   36   84  126  126   84   36    9    1
 10|    1   10   45  120  210  252  210  120   45   10    1
 11|    1   11   55  165  330  462  462  330  165   55   11    1
 12|    1   12   66  220  495  792  924  792  495  220   66   12    1
 13|    1   13   78  286  715 1287 1716 1716 1287  715  286   78   13    1
```

位取り記法と基数変換

位取り記法　自然数 r を一つ決めて，任意の数 n を r ごとに桁挙げして表記する方法。r のことを基数といい，基数 r のときの表記を r 進数表記または r 進法という。通常は 10 進法。

$n = k_\ell r^\ell + k_{\ell-1} r^{\ell-1} + \cdots + k_1 r^1 + k_0 r^0$ と表せるとき，数 n を $k_\ell k_{\ell-1} \ldots k_1 k_0$，あるいは，$r$ 進法であることを明記して $(k_\ell k_{\ell-1} \ldots k_1 k_0)_r$ と書く。10 進法のときは明記しないことが多い。

例。　123 は 10 進法で書かれていて，$123 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$ ということ。これを例えば 3 進法で書くと $123 = 1 \times 3^4 + 1 \times 3^3 + 1 \times 3^2 + 2 \times 3^1 + 0 \times 3^0 = (11120)_3$ となる。

基数変換　基数を変えて位取り記法をする，例えば 10 進法の数 n を 10 以外の r 進法に書き換えるためには，各桁の数を求めなければならない。剰余算で計算できる。

1. $n_0 := n$, $\ell := 0$ とおく。
2. 以下の計算を $n_{\ell+1} = 0$ になるまで繰返す。
   (a) $k_\ell := n_\ell \mod r$; $n_{\ell+1} := (n_\ell - k_\ell) \operatorname{div} r$; $\ell := \ell + 1$
3. $k_\ell k_{\ell-1} \ldots k_1 k_0$ と並べて出力して終了。

アルゴリズム入門

計算手順とプログラムとアルゴリズム　与えられた問題をコンピュータを用いて解くにはどうするか。

1. 問題の (数学的) 解法を考える。
   厳密に解く方法があれば OK。なければ，近似的に解く方法を考案する。
2. 数学的解法から，数値的解法を考案する。
   極限操作があれば，収束列を途中で打ち切る。無限回反復があれば，有限回反復で打ち切る。
3. 数値的解法からアルゴリズムを構成する。
   アルゴリズム　＝　与えられた問題を解くための，機械的操作からなる有限の手続き (計算手順)
   機械的操作：コンピュータあるいはプログラミング言語が持つ基本的な演算
4. アルゴリズムからプログラムを作成する。
   Pascal などのプログラミング言語で実装する。
5. プログラムを実行する。

代表的なアルゴリズム

最大値探索　n 個のデータ $a_1, \ldots, a_n$ の最大値を求める。

1. $b := a_1$
2. 以下の処理を $i := 2$ から n まで繰返す。
   (a) $a_i > b$ なら $b := a_i$ とする。
3. b に保存されている値が求める最大値。終了。

並べ替え (バブルソート)　n 個のデータ $a_1, \ldots, a_n$ を昇順 (小さい順) に並べ替える。

1. 以下の処理を $i := 1$ から n まで繰返す。
   (a) 以下の処理を $j := n$ から $i + 1$ まで逆順に繰返す。
      i. $a_j < a_{j-1}$ なら，
         a_j と a_{j-1} を入れ替える：$b := a_{j-1}$; $a_{j-1} := a_j$; $a_j := b$
2. 終了。

計算量の概念　アルゴリズムの良さを評価する指標

そのアルゴリズムの実行に要する演算回数 (時間量) と記憶容量 (領域量)。特にそれらを，計算対象の典型的な数 (計算対象の値，データ数など) の関数として表現したもの。

- 演算回数
  - 四則演算，特に乗算と除算の回数
  - 条件文の通過回数（比較演算＋実行文の選択）
  - 繰返し文の反復回数
- 記憶容量
  - 変数の数，配列の要素数
  - データ保管用だけでなく作業用の領域も含む

例．A) 最大値検索は，データ数 n のとき，長さ n の配列を使い，n 回の比較演算で終わるので，$O(n)$。
B) バブルソートは，データ数 n のとき，長さ n の配列を使い，$\frac{1}{2}n(n+1)$ 回の比較演算が必要なので，$O(n^2)$。
C) エラトステネスのふるいは，対象とする自然数 n のとき，n 回程度の演算で終わるので，$O(n)$。
D) ユークリッドの互除法は，対象とする自然数 n, m のとき，自然数 $\log \max\{n, m\}$ の程度の回数の反復計算で終わることが知られているので，$O(\log \max\{n, m\})$。

今週のサンプルプログラム

(4a) [Renshu4a.pas] 整数の積と商と剰余の九九の表を表示

```
program Renshu4a( input, output );
var n, m, d: integer;
    p: array[1..9,1..9] of integer;
begin
  for n:=1 to 9 do begin {積の九九を配列に保存}
    for m:=1 to 9 do begin
      p[n,m] := n * m; {積}
    end
  end;
  write( 'n*m|' ); {積の九九}
  for m:=1 to 9 do write( '  ', m );
  writeln;
  write( '----' );
  for m:=1 to 9 do write( '---' ); {罫線}
  writeln;
  for n:=1 to 9 do begin
    write( '  ', n, '|' ); {第 n 行}
    for m:=1 to 9 do begin
      if (abs(p[n,m])<10) then write( ' ' ); {桁合わせ}
      write( ' ', p[n,m] )
    end;
    writeln
  end;
  writeln;

  write( 'n div m|' ); {商の九九}
  for m:=1 to 9 do write( '  ', m );
  writeln;
  write( '--------' );
  for m:=1 to 9 do write( '---' ); {罫線}
  writeln;
  for n:=1 to 9 do begin
    write( '      ', n, '|' ); {第 n 行}
    for m:=1 to 9 do begin
      d := n div m; {商}
      write( '  ', d )
    end;
    writeln
  end;
  writeln;

  write( 'n mod m|' ); {剰余の九九}
  for m:=1 to 9 do write( '  ', m );
```

```
  writeln;
  write( '--------' );
  for m:=1 to 9 do write( '---' ); {罫線}
  writeln;
  for n:=1 to 9 do begin
    write( '      ', n, '|' ); {第 n 行}
    for m:=1 to 9 do
      write( '  ', n mod m ); {剰余}
    writeln
  end
end.
```

実行結果：

```
n*m|  1  2  3  4  5  6  7  8  9
------------------------------
  1|  1  2  3  4  5  6  7  8  9
  2|  2  4  6  8 10 12 14 16 18
  3|  3  6  9 12 15 18 21 24 27
  4|  4  8 12 16 20 24 28 32 36
  5|  5 10 15 20 25 30 35 40 45
  6|  6 12 18 24 30 36 42 48 54
  7|  7 14 21 28 35 42 49 56 63
  8|  8 16 24 32 40 48 56 64 72
  9|  9 18 27 36 45 54 63 72 81

n div m|  1  2  3  4  5  6  7  8  9
-----------------------------------
      1|  1  0  0  0  0  0  0  0  0
      2|  2  1  0  0  0  0  0  0  0
      3|  3  1  1  0  0  0  0  0  0
      4|  4  2  1  1  0  0  0  0  0
      5|  5  2  1  1  1  0  0  0  0
      6|  6  3  2  1  1  1  0  0  0
      7|  7  3  2  1  1  1  1  0  0
      8|  8  4  2  2  1  1  1  1  0
      9|  9  4  3  2  1  1  1  1  1

n mod m|  1  2  3  4  5  6  7  8  9
-----------------------------------
      1|  0  1  1  1  1  1  1  1  1
      2|  0  0  2  2  2  2  2  2  2
      3|  0  1  0  3  3  3  3  3  3
      4|  0  0  1  0  4  4  4  4  4
      5|  0  1  2  1  0  5  5  5  5
      6|  0  0  0  2  1  0  6  6  6
      7|  0  1  1  3  2  1  0  7  7
      8|  0  0  2  0  3  2  1  0  8
      9|  0  1  0  1  4  3  2  1  0
```