改訂新版　序

本書は『C 言語による最新アルゴリズム事典』(1991 年初版）の組版・装丁を新しくし，若干の修正をしたものである。項目に増減はない。

本書初版の「序」では次のように書いた：

> 本書は，たとえば
>
> - 数の列を大きさの順にできるだけ速く並べ替えたい
> - 行列の固有値・固有ベクトルを求めたい
> - “SEND + MORE = MONEY” のようなパズルを解きたい
>
> といったような，さまざまな種類の問題について，コンピュータ向きの解法（アルゴリズム）を集めたものである。
>
> たとえば大きさの順に整列する方法について調べるには，「整列」(巻末の索引から「並べ替え」や「ソーティング」でも探せる）を引くと，整列の概要と個々の方法の名前がわかる。その中からたとえば「クイックソート」を引き直すと，クイックソートの説明と C 言語のプログラム（再帰版，非再帰版）が見つかる。
>
> また，たとえば「番人」の項目を読んで具体例をさらに調べたければ，巻末の索引で「番人」を引くと，番人を使ったプログラムのページがすべてわかる。

幸い，評判は良く，石田晴久先生の『コンピュータの名著・古典 100 冊』(インプレス，2003 年）の百選にも選んでいただいた。また，その後の『Java によるアルゴリズム事典』(奥村ほか，技術評論社，2003 年）の基となった。

おかげさまで，27 年を経た今に至るまで，刷を重ねることができた。支えてくださった読者に感謝したい。

しかし，グラフィックス関連の項目が PC-9800 シリーズ（初版当時の日本での標準パソコン）を仮定したコードになっていたことについては，さすがにまずい状態であった。この改訂新版ではまず，これらの部分を，機種依存しないコードで置き換えた。機種依存しないグラフィックスというのは，C 言語では難しいと思われるかもしれないが，ビットマップは BMP 形式，ベクトルグラフィックは SVG 形式や EPS 形式で書き出すだけなら，簡単にできる。BMP，SVG は，Windows，Mac，Linux のどれでも，標準の画像ビューアまたは Web ブラウザで開くことができる。

これ以外の部分は触らないつもりであったが，読み返してみると，さすがに時代の流れを感じる記述がいくつか目についたので，若干の加筆を行った。

あとは，現在の習慣に従って，`int main()` を `int main(void)` にしたり，`EXIT_SUCCESS` を `0` にしたりした。

本書掲載ソースコードおよびサポート情報は `https://github.com/okumuralab/algo-c` から得られる。

本書初版は TEX から写研の写植機に出力して印刷した。その際，東京書籍印刷株式会社（現：株式会社リーブルテック）の小林 肇 さんにたいへんお世話になった。もうその写植機は動いていないので，改訂新版は現在の標準技術を使って TEX から PDF 出力（電子版），および PDF 経由で CTP 出力（紙版）した。フォントは，石井中明朝・中ゴシックがヒラギノに，Computer Modern が Palatino や Inconsolata になった。改訂にあたって，技術評論社の須藤真己（まさき）さんには始終お世話になった。

2018 年 3 月 28 日
奥村 晴彦

凡　例

- 項目は「あ」—「ん」「A」—「Z」の順に並べた。「バブルソート」は「はふるそおと」の位置にある。海外の固有名詞は英字で綴り，代表的な読みをかっこ書きした。
- 本文中で † を冠した語は項目語である。
- 「⇒ 何々」は「「何々」を参照せよ」の意である。
- `a[0..n-1]` は `a[0]` から `a[n-1]` までの意である。
- "\"（バックスラッシュ）はフォントによっては "¥" と表示される場合がある。
- "␣" は（半角）空白を見やすく書いたものである。
- $(1101)_2$ は 2 進法で 1101 と表される数である（⇒ 基数の変換）。
- たとえば実行時間が $O(n^2)$ であるとは，実行時間がたかだか n^2 に比例する程度であることを意味する（⇒ O 記法）。
- $\lfloor x \rfloor$ は x を小さい方に丸めた整数（C 言語の `floor(x)`）である。また，$\lceil x \rceil$ は x を大きい方に丸めた整数（C 言語の `ceil(x)`）である（⇒ 床・天井）。
  例: $\lfloor 3.14 \rfloor = 3$，$\lceil 3.14 \rceil = 4$。
- "$x \bmod y$" は「x を y で割った余り」の意である（⇒ 整数の除算）。
  例: 8 を 3 で割ると 2 余るので，$8 \bmod 3 = 2$。
- $\max(\dots)$ は最大値，$\min(\dots)$ は最小値である。
  例: $\max(35, 97, 12) = 97$，$\min(35, 97, 12) = 12$。
- 行列についての記法は ⇒ 行列。

あ

値の交換 exchange of values

変数 a，b の値を交換するには，

```
a = b;  b = a;  /* 駄目! */
```

では駄目である。余分な変数 temp を使って

```
temp = a;  a = b;  b = temp;
```

とする。あるいは †ビットごとの排他的論理和を使って

```
b ^= a;  a ^= b;  b ^= a;
```

としてもよい。これでよい理由はビットごとの排他的論理和の性質 $a \oplus a = 0$ および結合法則 $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ から導ける。

余分な変数を使わない値交換のアルゴリズムは，ほかにも

```
b = a - b;  a -= b;  b += a;
```

や，実数で b $\neq 0$ ならさらに素直でない方法

```
b = a / b;  a /= b;  b *= a;
```

が考えられる。

Ruby，Python，Julia など一部の言語では a,b = b,a のような書き方ができる。

swap.c

swap(&x, &y); のように変数名に & を付けて呼び出すと int 型の変数 x，y の値を交換する。

```c
void swap(int *x, int *y)
{
    int temp;

    temp = *x;  *x = *y;  *y = temp;
}
```

誤り検出符号 error detecting code

クレジットカードの番号のような数字の列を扱うとき，誤記・誤読を防ぐ一つの方法は，もともとたとえば素数 97 で割って 1 余る数だけを番号に使うことである。こうすれ

あ

ば，誤記・誤読は 96/97 の確率で検出できる（97 の倍数だけ使う方式では末位の 0 の欠落が検出できない [4]）。

実際のクレジット番号で使われている Luhn（ルーン）のアルゴリズムは，最下位桁から数えて偶数番目（10 の位，1000 の位，…）の桁だけ 2 倍し，すべての桁の値の合計が 10 の倍数になるように，最下位桁（チェックディジット）を調整する。例えば偶数番目の桁が 9 であれば，2 倍して 18 になるが，$1+8=9$ として扱う（あるいは同じことであるが 9 を超えたら 9 を引いて $18-9=9$ にする）。より好ましい性質を持つものに Damm（ダム）のアルゴリズム [5] がある。

その他の誤り検出の方法については ⇒ †ISBN，†チェックサム，†CRC。ファイルの改竄（ざん）防止には暗号学的ハッシュ関数（⇒ †ハッシュ法）を使う。

luhn.c

クレジットカード番号をチェックする Luhn のアルゴリズム。

```c
#include <stdio.h>
#include <string.h>

int main(void)
{
    char *s = "5555555555554444";                    /* カード番号（例）*/
    int i, d, w = 1, t = 0;

    for (i = strlen(s) - 1; i >= 0; i--) {
        d = w * (s[i] - '0');
        if (d > 9) d -= 9;
        t += d;
        w = 3 - w;
    }
    if (t % 10 == 0) printf("有効\n"); else printf("無効\n");
    return 0;
}
```


[1] Benjamin Arazi. *A Commonsense Approach to the Theory of Error Correcting Codes*. MIT Press, 1988. 数学的な本ではないが非常に読みやすい．

[2] Richard W. Hamming. *Coding and Information Theory*. Prentice Hall, second edition 1986. 初等的な数学だけを使い，誤り訂正を含めた符号化の理論全般をやさしく解説した名著．

[3] W. Wesley Peterson and E. J. Weldon, Jr. *Error-Correcting Codes*. MIT Press, second edition 1972. 本格的な教科書．巻末に原始多項式の表がある．

[4] W. Wesley Peterson. *Communications of the ACM*, 34(12): 110–113, December 1991.

[5] H. Michael Damm. *Discrete Mathematics*, 307(6): 715–729 (2007).

アルゴリズム algorithm

一般に，問題を解くための処方。特に，コンピュータ向きの解法。算法と訳す。アルゴリズムを特定のプログラム言語で書いたものがプログラムである。9 世紀アラビアの数学者の名前 al-Khwārizmī（アル・フワリズミ，アル・コワリズミ）が語源。

暗号 cryptosystem

通信文が漏れても第三者には意味が分からないようにする仕組み。

送り手は平文（plaintext）を暗号化（動詞 encrypt，名詞 encryption）して暗号文（ciphertext）にする。受け手は暗号文を復号（動詞 decrypt，名詞 decryption）して平文に戻す。第三者が暗号の仕組みを解析することを解読（動詞 cryptanalyze，名詞 cryptanalysis）という。

アルファベットを順繰りにずらすだけの暗号を Caesar（シーザー）暗号（Caesar cipher）という。たとえば IBM を −1 文字ずらせば HAL に，+7 文字ずらせば PIT になる。Caesar は 3 文字ずらした。復号には何文字ずらしたかの情報が必要である。このような情報を暗号の鍵（key）という。

Caesar 暗号と似たものに，次のような †ビットごとの排他的論理和による暗号化がある。k はあらかじめ定めた鍵である。

```
while ((c = getchar()) != EOF) putchar(c ^ k);
```

これで文書は一見でたらめなバイト列と化する。ビットごとの排他的論理和は 2 回行うと元に戻る（$c \oplus k \oplus k = c$）。したがって，暗号文を同じプログラムにもう一度通せば平文に戻る。残念ながら，これでは 256 通りの鍵をすべて試すだけで解読できる。鍵を数バイト長にして順繰りに使っても，解読はあまり難しくならない。解読をもう少し難しくするには，1 文字ごとに k を †乱数で変え，本当の鍵は乱数の初期化だけに使うことが考えられる。

ちなみに，引き算も $k-(k-c)=c$ のように 2 回すると元に戻る。この方が，文字コードがたとえば 0x20 から 0x7E までに限られている場合にも

```
while ((c = getchar()) != EOF)
    putchar((k + 0x7E - c) % 0x5F + 0x20);
```

とできるので便利かもしれない。

残念ながら，このような簡単な暗号では，解読は容易である。本格的な暗号として，本書初版では DES（Data Encryption Standard）や FEAL [4] を紹介したが，現在の推奨は，AES（Advanced Encryption Standard）や，NTT と三菱電機が共同開発した Camellia などである。ただ，これらとて永久に安全というわけではない。また，これらの暗号はす

べて，暗号化の鍵と復号の鍵が共通な共通鍵暗号である。これに対して，暗号化の鍵と復号の鍵が異なる公開鍵暗号は，暗号化の鍵だけ公開しておけば，だれでも暗号化できるが，復号できるのは復号の鍵を持っている者だけである。

文献 [1–4] は初版時のものである。追加した文献 [5–7] を参照されたい。オープンソースの暗号ツールとしては GnuPG（コマンド gpg）が有名である。

`crypt.c`

簡単な暗号化・復号プログラム。たとえば

```
crypt foo bar 12345
```

とすれば foo というファイルを bar というファイルに鍵 12345 で変換する。鍵は整数値で，省略すると 1 になる。こうしてできたファイル bar を同じ鍵 12345 で再度このプログラムにかけると元の foo が復元できる。

0 以上 255 以下の乱数と各文字とのビットごとの排他的論理和を出力しているだけである。RAND_MAX + 1 が 256 の倍数なら行 16 と行 18 は不要である。

再度注意するが，この程度の暗号では，十分な強度は期待できない。アルゴリズムを秘密にするのではなく，評価の定まったアルゴリズムと複雑で長い鍵を使い，鍵の管理をしっかりすることが大切である。

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 int main(int argc, char *argv[])
 4 {
 5     int c, r;
 6     FILE *infile, *outfile;
 7
 8     if (argc < 3 || argc > 4 ||
 9      (infile  = fopen(argv[1], "rb")) == NULL ||
10      (outfile = fopen(argv[2], "wb")) == NULL) {
11         fputs("使用法: crypt infile outfile [key]\n", stderr);
12         return 1;
13     }
14     if (argc == 4) srand(atoi(argv[3]));
15     while ((c = getc(infile)) != EOF) {
16         do {
17             r = rand() / ((RAND_MAX + 1U) / 256);
18         } while (r >= 256);
19         putc(c ^ r, outfile);
20     }
21     return 0;
22 }
```

[1] Jennifer Seberry and Josef Pieprzyk. *Cryptography: An Introduction to Computer Security*. Prentice Hall, 1989.

[2] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1988. 228–236 ページ．第 2 版（1992 年）の邦訳：丹慶 勝市，佐藤 俊郎，奥村 晴彦，小林 誠 訳．『Numerical Recipes in C 日本語版』．技術評論社，1993.

[3] Al Stevens. DES Revisited and the Shaft. *Dr. Dobb's Journal*, November 1990, 149–153, 164–166.

[4] 辻井 重男，笠原 正雄 編．『暗号と情報セキュリティ』．昭晃堂，1990．DES の仕様概要，FEAL の仕様が載っている．

[5] Bruce Schneier. *Applied Cryptography*. Wiley, 1993. 名著だがやや古くなった．2015 年に再刊されたが内容は同じ．

[6] 結城 浩．『暗号技術入門 第 3 版』．SB クリエイティブ，2015.

[7] 光成 滋生．『クラウドを支えるこれからの暗号技術』．秀和システム，2015．https://herumi.github.io/ango/

安定な結婚の問題 stable marriage problem

N 人の男性と N 人の女性が集団見合いをし，おのおの異性を好みの順に順位づけした。この順位表をもとにして安定な縁結びの仕方を決めるのがこの問題である。仮に男性 M_1 と女性 F_1 が結婚し，男性 M_2 と女性 F_2 が結婚したのに，じつは M_1 は F_1 より F_2 を好み，F_2 は M_2 より M_1 を好んだならば，将来問題が起こりかねない。このようなことがないのがここでいう安定な結婚である。

この問題には次のような非常に簡単な解法がある。まず男性 1 が彼のリストでトップの女性に申し込む（実際にではなく，アルゴリズム上での話である）。彼女はまだ誰からも声がかかっていないので，とりあえずオーケーする。次に，男性 2 が彼のリストでトップの女性に求婚するが，彼女がすでにほかの男性にオーケーを出していた場合，もし彼女が新しい求婚者の方を好めば，古い男性を振って新しい求婚者にオーケーを出す。振られた男性はすぐさま彼のリストで次のランクの女性に申し込む。以下同様に続ける。全員が納得できる縁結びが $O(N^2)$ ステップで完成する。

このアルゴリズム（Gale–Shapley のアルゴリズム）で知られる Shapley は 2012 年のノーベル経済学賞を受賞した。

実際の見合いでこの種のアルゴリズムが使われたことがあるかどうかは寡聞にして知らないが，人と職場を結びつけるなどの状況では実際に使われているようである。

大学の選抜をこの種のアルゴリズムで行えばどうであろうか。各受験生は大学名を入りたい順に並べた表を入試センターに提出する。各大学も受験生を採りたい順に並べてセンターに提出する。センターは大型計算機を使って安定な大学・受験生の組合せを決める。こうすれば，読みを誤って定員割れを起こす大学や，高望みしすぎて悔やむ受験生が減るのではなかろうか。

あ

marriage.c

女 N 人，男 N 人がそれぞれ異性 N 人を好みの順に列挙した $2N \times N$ の表を入力すると，安定な結婚の解を一つ出力する。男女とも $1, \ldots, N$ の番号（整数）で表す。$N = 3$ ならたとえば次の縦線の左側のように入力する。右側はその意味である。

2 3 1	女$_1$ にとって 男$_2$ > 男$_3$ > 男$_1$
2 1 3	女$_2$ にとって 男$_2$ > 男$_1$ > 男$_3$
3 2 1	女$_3$ にとって 男$_3$ > 男$_2$ > 男$_1$
1 3 2	男$_1$ にとって 女$_1$ > 女$_3$ > 女$_2$
2 3 1	男$_2$ にとって 女$_2$ > 女$_3$ > 女$_1$
3 1 2	男$_3$ にとって 女$_3$ > 女$_1$ > 女$_2$

```
#include <stdio.h>
#include <stdlib.h>
#define N  3                                                    /* 各性の人数 */
int boy[N+1], girl[N+1][N+1], position[N+1], rank[N+1][N+1];

int main(void)
{
    int b, g, r, s, t;

    for (g = 1; g <= N; g++) {                                  /* 各女性の好み */
        for (r = 1; r <= N; r++) {
            scanf("%d", &b);  rank[g][b] = r;
        }
        boy[g] = 0;  rank[g][0] = N + 1;                        /* †番人 */
    }
    for (b = 1; b <= N; b++) {                                  /* 各男性の好み */
        for (r = 1; r <= N; r++) scanf("%d", &girl[b][r]);
        position[b] = 0;
    }
    for (b = 1; b <= N; b++) {
        s = b;
        while (s != 0) {
            g = girl[s][++position[s]];
            if (rank[g][s] < rank[g][boy[g]]) {
                t = boy[g];  boy[g] = s;  s = t;
            }
        }
    }
    for (g = 1; g <= N; g++) printf("女 %d - 男 %d\n", g, boy[g]);
    return 0;
}
```


[1] Dan Gusfield and Robert W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, 1989.

[2] Donald E. Knuth. *Mariages stables et leurs relations avec d'autres problèmes combinatoires: Introduction à l'analyse mathématique des algorithmes*. Les Presses de l'Université de Montréal, 1976.

[3] Robert Sedgewick. *Algorithms*. Addison–Wesley, second edition 1988. 499–504 ページ．元は Pascal 版であったが，その後 C 版，C++ 版，Java 版も出た．邦訳あり．原著最新版（fourth edition 2011）は Kevin Wayne との共著．

[4] Niklaus Wirth．片山 卓也 訳．『アルゴリズム＋データ構造＝プログラム』．マイクロソフトウェア／日本コンピュータ協会，1979．168–176 ページ．