Program Promenade

連載

Enigmaへの挑戦[1)]

和田 英一（IIJ 技術研究所）
wada@u-tokyo.ac.jp

■Enigmaの仕掛け

ACM国際大学対抗プログラミングコンテストの過去の問題は随所にアーカイブされている．その1つがスイス連邦工科大学ETH[2)]である．そこへ行きNorthwestern European Regionals 2001を眺めていたら，1918年にArthur Scherbiusが発明，第2次世界大戦の頃，ドイツが使い，Alan TuringらがBletchley Park（右の写真．実際に解読していたのはもっと貧しい建物）で解読に従事していた暗号機Enigmaが話題になっていたので，今回はその問題をやってみることにした．めずらしく入力と出力のファイルもWebから取れる．

下の写真がEnigmaで，Bletchely Parkに展示されているものである（写真はNII三浦謙一による）．Enigma暗号機を模式的に描くと**図-1**のようになる．キーボード状のものが2つあり，下がキーボード，上が対応するランプボードである．キーボードで何かの文字を押すと変換された文字のランプが点灯する仕掛けである．ドイツ語では"Y"より"Z"を多用するので，ドイツのタイプライタは"Y"と"Z"を反対に配置する．またこれは3段キーボードなので，キー配列は上から下へ1/3ずつずれている（4段キーボードは1/2, 1/4, 1/2とずれている）．さらに下に（本物のEnigmaでは手前の面に）プラグボードがあり，任意の文字対間をプラグで接続すると文字同士の交換ができる．配線がなければ文字は変換されない．

上方のローターダイアルと書いてある辺りに3枚のローターがある．図の縞模様はローターを手動で回転するためのダイアルである．縞模様の左にローターの位置を文字で確認するための窓がある．ローター自身は両面それぞれに円周状に26個の接点があり，片面の各接点は逆面の接点とランダムに接続されている．ローターというわけは1文字変換すると右ローターが1文字分回転し，右ローターが1回転すると中ローターが1文字分回転，中ローターが1回転すると左ローターが1文字分回転するからである．大体は自動車の距離計と同様と思ってよい（厳密にいうと違う）．距離に相当する文字が窓に現れる．

このほかに反射ローターがある．これは片面だけに接点があり，その2字を対にして接続されている．したがってこの変換は自己対称になっている．つまり"a"が"o"に変換されるなら，"o"は"a"に変換される．"a"と"o"の接点が接続してあるわけだ．この反射ローターの構造のため，本物のEnigmaではある文字は必ず他の文字へ変換されるという弱点があり，読み出し位置を決める重要な手がかりを解読者に与えて

いた．

図-2は結線図を示す．キーボードでいずれかのキー（たとえば"a"）を押すと，その下のスイッチが閉じ，電流がプラグボードIPへくる．プラグで接続してある相手のターミナルから右ローター（π_0）へ行き，中（π_1），左のローター（π_2）を通り，最後の反射ローター（π_R）を通り，今度は左，中，右のローターを逆に通過，再びプラグボードを通り，ランプボードのある文字（今は"j"）が点灯する．Enigmaを使うにはプラグボードを指定通りに配線し，ローター（と反射ローター）を指定通りに挿入し，各ローターを開始位置まで指定通りに手動で進める．その後変換すべき文字列をキーボードから打ち込み，点灯した文字を書き取るのである．

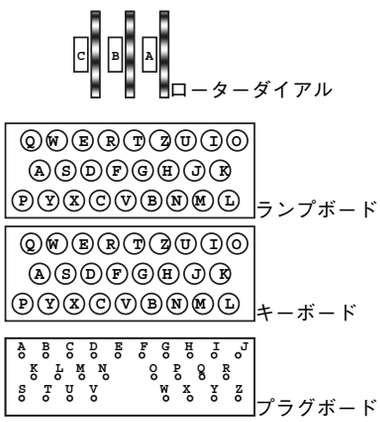

図-1 模式図

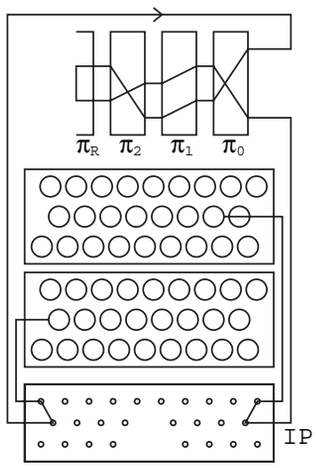

図-2 結線図

暗号は平文の文字を1字ずつ別の文字に変換して作る．一定の変換規則だと文字の頻度情報などから鍵なしでも解読できる．それでEnigmaではこの規則をローターの回転により1文字ごとに変えようとした．**図-3**に変換の様子を示す（暗号では大文字だけ使うが，本稿では問題と同様に小文字を使う）．アルファベットの順列を変換文字列（または変換表）という．変換文字列がたとえば"quickbrown..."なら，これは"a"を"q"に，"b"を"u"に，"c"を"i"に，"d"を"c"に...と変換する規則を表す．

```
      abcd efgh ijkl mnop qrst uvwx yz
IP    abcd efgh ijkl mnop qrst uvwx yz
π0    wfbt izru vcqe jpok shxg mady rl
π1    hmrg nqpk jcai vwlu ebfz syxt do
π2    drua hlbf zvgm wckx piqy sont je
πR    owtv skyp jifm luah rqec ndbz gx
π2-1  dgna zhke ryof lwvq sbux cjmp ti
π1-1  krjy qsda liho bezg fcux pmnw vt
π0-1  vcjw lbtr empz ugon kyqd hias xf
IP-1  abcd efgh ijkl mnop qrst uvwx yz
```

```
      abcd efgh ijkl mnop qrst uvwx yz
IP    abcd efgh ijkl mnop qrst uvwx yz
π0    eash ymtu bpdi onjr gwfl zcxq kv
π1    hmrg nqpk jcai vwlu ebfz syxt do
π2    drua hlbf zvgm wckx piqy sont je
πR    owtv skyp jifm luah rqec ndbz gx
π2-1  dgna zhke ryof lwvq sbux cjmp ti
π1-1  krjy qsda liho bezg fcux pmnw vt
π0-1  bivk asqd loyt fnmj xpcg hzrw eu
IP-1  abcd efgh ijkl mnop qrst uvwx yz
```

図-3 変換の進行状況

最上段のアルファベットは見出しである．その下に変換文字列が9行あり，上からIP, π_0, π_1, π_2, π_R, π_2^{-1}, π_1^{-1}, π_0^{-1}, IP^{-1}とする（π_0, π_1, π_2, IPの逆変換をそれぞれπ_0^{-1}, π_1^{-1}, π_2^{-1}, IP^{-1}と書く．そのつもりでπ_0とπ_0^{-1}を見ると，π_0では"a"が"w"に変換され，π_0^{-1}では"w"が"a"に変換されている）．

IP, IP^{-1}は最上段の見出し行と同じだから，この例では変換していない（つまりプラグボードは配線していない）．まず"g"を変換してみる．左の文字列群で最上行と2行目の"g"が丸で囲ってあるのは"g"をIPで変換したら"g"になることを示す．"g"の下の3行目の"n"の丸はπ_0は"g"を"n"に変換することを示す．そこで次は"n"の下のπ_1の行を見ると"w"に丸があり，"w"に変換した．これを繰り返すと，最後に"m"が得られる．すなわち"g"はこの変換文字列群で"m"に変換した．ここでローターが1文字分前進する．

右の文字列群は次の文字"r"を変換するもので，π_0とπ_0^{-1}が違っている．前のπ_0, π_0^{-1}を1文字左へ回転し，アルファベット順で1つ前の文字になっている．これで変換すると最後に"a"が得られる．2回の暗号化で平文"gr"

は暗号"ma"になった．この文字列群は逆変換にも使える．左の文字列群で暗号"m"を変換すると平文"g"に，右の文字列群で"a"を変換すると"r"になる．試してみて欲しい．Enigmaの特徴の1つは暗号化にも平文化にも使えることである．

■Enigma問題の説明

Northwestern European Regionals 2001問題Eでは各暗号の組に対し，π_0, π_1, π_2, π_R, IP（プラグボード）の変換表と各ローターの最初の位置，（最大80文字までの）平文，暗号文が与えられている．ローターの回転は文字列の左回転シフトの関係にある．コンテストの問題では反射ローターも回転する．また問題の反射ローターには"gbc.."で始まるものがある，つまり"b"を"b"に，"c"を"c"に変換する．後述のプログラムのサンプル2で"a"が"a"に変換され，プログラムミスかと思ったが，反射ローターの仕様のせいであった．さらにローターの繰り上がり方も本物と違い，開始位置から1回転すると，次のローターが1文字進む．

たとえば

```
wfbtiznuvcqejpokshxgmadyrl          ;; 右ローター π0
hmrgnqpkjcaivwluebfzsyxtdo          ;; 中ローター π1
druahlbfzvgmwckxpiqysontje          ;; 左ローター π2
owtvskypjifmluahrqecndbzgx          ;; 反射ローター πR
?bcdefghijklmnopqrstuvwxyz          ;; プラグボードIP
aaaa                                ;;開始位置 左からπ0, π1…
manyorganizationsrelyoncom??ters    ;;平文
grsuztldsznkwnerdpfbovvqnobkyiqn    ;;暗号文
```

が与えられ，それに対し解読した平文を答えるのである．平文が与えられているのに平文を答えるというのは不思議だが，平文は部分しか分からないことになっている．つまり上の例の"??"である．さらに変換表や開始位置も虫食いの可能性がある．ただし虫食い（"?"）の数は1組に3文字までとなっている．

上の例のように変換表IPの1文字の"?"は自明である．アルファベットで欠けている文字を補えばよい．2文字の"?"は2通りの可能性を調べる．変換表に"?"が3文字あれば，3文字の順列6通りを試みることになる．

面倒なのは開始位置の"?"で，これはローターごとに独立であり，26通りをテストしなければならない．

いずれにしても暗号文と平文（の一部）が与えられているから決められる．

■変換表が既知のプログラム

変換表，開始位置はすべて分かっているとして，問題にあるような暗号化，解読をするプログラムを書いてみる．平文，暗号文も80字以内となっているから，実際にはたかだか最初の2個のローターしか回転せず，3個目からの回転のチェックは不要．(rotate rot)はローターrotを1文字分回転する．ローターの文字列を1文字回転シフトしnewrotとする．次に各文字をアルファベット順で1文字前のものに変える．"a"(=97)は"z"(=122)にする．(rot n r)は文字列rをn回，回転する．

enigの引数は見ての通り．まず初期位置initを"a"を0とする数値リストにし，各変換表を指定通りに回転する．その後暗号文の各文字につき，図-3のように9回の変換を行い，最後に文字に戻す．毎度右ローターとカウンタiを進め，カウンタが26の倍数になったら中ローターを1歩進める（以下のプログラムではπ_0, π_1, π_2, π_R, IPをそれぞれpi0, pi1, pi2, pir, ipで表す）．

中心は変換部分で(conv p c)は整数cを変換表pで変換する（(conv "quick..." 3) => 2）．それには(string-ref p c)でpのc番目の文字をとり，asciiintで整数にする．逆変換(invc p c)では文字列pでcは何文字目かを知りたい（(invc "quick..." 2) => 3）から，(string->list p)でまず文字列を文字のリストにし，(intascii c)でcを文字にしてmemberで探す．すると探す文字を先頭にした部分リストが返るから，その長さを26（文字列の長さ）から引く．

Program Promenade Program Promenade Program Promenade

```
(define (asciiint ch) (- (char->integer ch) 97));asciiから0起点の自然数へ a→0, z→25
(define (intascii n) (integer->char (+ n 97)))  ;上の逆変換 0→a, 25→z

(define (rotate rot)            ;ローターrotを1文字分回転
  (let ((newrot (string-append (substring rot 1 26) (substring rot 0 1))))
    (list->string (map integer->char
      (map (lambda (c) (if (= c 97) 122 (- c 1)))
        (map char->integer (string->list newrot)))))))

(define (rot n r) (if (= n 0) r (rot (- n 1) (rotate r))))      ;rotateをn回実行

(define (enig pi0 pi1 pi2 pir ip init plain crypt)      ;cryptを解読
  (define (conv p c) (asciiint (string-ref p c)))       ;変換と逆変換ルーチン
  (define (invc p c) (- 26 (length (member (intascii c) (string->list p)))))
  (set! init (map asciiint (string->list init)))        ;0起点の数値リストへ
  (set! pi0 (rot (car init) pi0))       ;右ローターpi0を初期位置まで回転
  (set! pi1 (rot (cadr init) pi1))      ;中ローターpi1を初期位置まで回転
  (set! pi2 (rot (caddr init) pi2))     ;左ローターpi2を初期位置まで回転
  (set! pir (rot (cadddr init) pir))    ;反射ローターpirを初期位置まで回転
  (let ((i 0))                          ;カウンタを初期化
    (list->string       ;cryptを変換した文字のリストから文字列にする
      (map (lambda (x)
        (let ((c (asciiint x)))         ;文字xを整数cにしてから
          (set! c (conv ip c))  (set! c (conv pi0 c))   ;ip,pi0で変換
          (set! c (conv pi1 c)) (set! c (conv pi2 c))   ;pi1,pi2で変換
          (set! c (conv pir c))                         ;pirで変換
          (set! c (invc pi2 c)) (set! c (invc pi1 c))   ;pi2,pi1で逆変換
          (set! c (invc pi0 c)) (set! c (invc ip c))    ;pi0,ipで逆変換
          (set! pi0 (rotate pi0)) (set! i (+ i 1))      ;カウンタを進め
          (if (= (remainder i 26) 0) (set! pi1 (rotate pi1)))   ;繰り上り?
          (intascii c)))        ;整数から文字に戻す
       (string->list crypt)))))) ;文字列cryptを文字のリストに
```

問題文にあったサンプルで使ってみる.

```
(enig "wfbtiznuvcqejpokshxgmadyrl" "hmrgnqpkjcaivwluebfzsyxtdo"
      "druahlbfzvgmwckxpiqysontje" "owtvskypjifmluahrqecndbzgx"
      "abcdefghijklmnopqrstuvwxyz" "aaaa"
      "manyorganizationsrelyoncom??ters"
      "grsuztldsznkwnerdpfbovvqnobkyiqn")
=> "manyorganizationsrelyoncomputers"           ;サンプル入力1

(enig "oqzunvhtxwryfebicmjpklsgda" "zupogrskynxtwdfqvbliejcmha"
      "kzvlyjuodmscewxtfbphriqgna" "gbcnylaztwkfmdspqvoiurjxeh"
      "rfyhkxbuvplgtqmdiewjosznca" "demo"
      "???" "ave")
=> "akf"                                        ;サンプル入力2
```

■虫食い対策

変換表の文字列の中に"?"が含まれていたらどうするか．前述のようにアルファベットで現れない文字の全順列を"?"に入れた候補を作り，それを順に使って暗号を解読し，平文の既知の部分と一致するまでmapで繰り返す．一致したらそこでループをやめて外へ脱出する．Schemeではループの脱出にcall-with-current-continuationを使う．

候補のリストを用意するのが以下のexpandである．これは下請けにqpos（"?"の出現位置のリストを返す）とlack（存在しない文字のリストを返す）を使う．下には順列のリストを作るためのpermも定義してある．目的のexpandは最後にある（"#\?"は"?"の文字定数）．

```
(define alph (string->list "abcdefghijklmnopqrstuvwxyz"));アルファベットの並び

(define (qpos s)        ;string sの中の?の位置のリストを返す
  (define (qs n s)
    (cond ((null? s) s) ;sが空になれば終わり 空リストを返す
          ((equal? (car s) #\?) (cons n (qs (+ n 1) (cdr s))))  ;先頭が?ならnをcons
          (else (qs (+ n 1) (cdr s)))))  ;?でなければ先を探す
  (qs 0 (string->list s)))      ;文字列を文字のリストにする

(define (lack s)        ;sの中にない文字のリストを返す
  (define (filter p l)  ;リストlからpで真になるもののリストを返す
    (cond ((null? l) l)
          ((p (car l)) (cons (car l) (filter p (cdr l))))
          (else (filter p (cdr l)))))
  (filter (lambda (c) (not (member c (string->list s)))) alph))

(define (perm l)        ;lの順列のリストを作る.
  (define (remove item seq)     ;並びseqからitemを削除したリストを返す
    (cond ((null? seq) seq)
          ((equal? item (car seq)) (cdr seq))
          (else (cons (car seq) (remove item (cdr seq))))))
  (if (null? l) (list l)
      (apply append (map (lambda (x) (map (lambda (p) (cons x p))
        (perm (remove x l)))) l))))

(define (expand s)      ;?の位置に欠落文字の順列を挿入し
  (define (ins ps cs s) ;可能性のあるローターの組を生成
    (if (null? ps) s
        (begin (string-set! s (car ps) (car cs))
               (ins (cdr ps) (cdr cs)s))))
  (let* ((q (qpos s)) (l (lack s)) (ss (perm l)))
    (map (lambda (x) (ins q x (substring s 0 (string-length s)))) ss)))
```

以上の準備の後，"?b?d?fghijklmnopqrstuvwxyz"を虫食い変換表としてexpandしてみる．

```
(qpos "?b?d?fghijklmnopqrstuvwxyz") => (0 2 4)  ; ?の位置は0, 2, 4

(lack "?b?d?fghijklmnopqrstuvwxyz") => (#\a #\c #\e)    ;存在しない文字はa, c, e

(expand "?b?d?fghijklmnopqrstuvwxyz") =>
   ("abcdefghijklmnopqrstuvwxyz" "abedcfghijklmnopqrstuvwxyz"
    "cbadefghijklmnopqrstuvwxyz" "cbedafghijklmnopqrstuvwxyz"
    "ebadcfghijklmnopqrstuvwxyz" "ebcdafghijklmnopqrstuvwxyz")
```

もちろん"?"がないか1個の変換表では候補の文字列は1つしかできない．

一方開始位置の方は"?"をアルファベットすべてで置き換えた候補のリストが必要である．いまアルファベットは"abcd"だけとし（(definealph (string->list "abcd"))として），"?x?y"から候補のリストを作るようなexpandinitを書く．つまり

```
(expandinit "?x?y") =>
  ("axay" "axby" "axcy" "axdy" "bxay" "bxby" "bxcy" "bxdy"
   "cxay" "cxby" "cxcy" "cxdy" "dxay" "dxby" "dxcy" "dxdy")
```

それには((a b) (c d))から((a c) (a d) (b c) (b d))を作るようなproductも定義する．

```
(define (product l)             ;要素のリストの組合せを作成
  (if (null? l) (list l)
    (apply append (map (lambda (x)
```

Program Promenade Program Promenade Program Promenade

```
      (map (lambda (y) (cons x y)) (product (cdr l)))) (car l)))))

(define (expandinit init)        ;初期位置の可能な組を生成
  (define (ins l)
    (cond ((null? l) l)
          ((equal? (car l) #\?) (cons alph (ins (cdr l))))
          (else (cons (list (car l)) (ins (cdr l))))))
  (map list->string (product (ins (string->list init)))))
```

この準備の後，前述のプログラムに候補を次々と供給する駆動プログラム（enigma）を書く．

```
(define (string-match plain decrypt)     ;平文の?以外はdecryptと一致?
  (define (match p c)                    ;1文字について比較
    (cond ((null? c) #t)                 ;文字列が終った→真を返す
          ((or (equal? (car p) (car c)) (equal? (car p) #\?))
           (match (cdr p) (cdr c)))      ;一致しているか平文が?なら次の文字へ
          (else #f)))                    ;不一致→偽を返す
  (match (string->list plain) (string->list decrypt)))   ;文字列を文字の並びへ

(define (enigma pi0 pi1 pi2 pir ip init plain crypt)
(call-with-current-continuation (lambda (exit)           ;脱出準備
(map (lambda (pi0)       ;右ローターの可能性の1つをpi0とする
(map (lambda (pi1)       ;中ローターの可能性の1つをpi1とする
(map (lambda (pi2)       ;以下同様
(map (lambda (pir)
(map (lambda (ip)
(map (lambda (init)
(let ((text (enig pi0 pi1 pi2 pir ip init plain crypt))) ;enigで変換 結果をtext
(if (string-match plain decrypt) (exit text)))) ;マッチすれば脱出
  (expandinit init)))            ;初期位置の可能な組を作る
  (expand ip)))                  ;ipの可能な組を作る
  (expand pir)))                 ;以下同様
  (expand pi2)))
  (expand pi1)))
  (expand pi0)))))

(define (readstr . s)   ; (readstr) (readstr s)と使い，文字列を読む
  (symbol->string (if (null? s) (read) (read (car s)))))

(define (solve)          ;1組の問題についてデータを読み enigmaを呼ぶ
  (let* ((pi0 (readstr s)) (pi1 (readstr s)) (pi2 (readstr s))
         (pir (readstr s)) (ip (readstr s)) (init (readstr s))
         (plain (readstr s)) (crypt (readstr s)))
    (newline) (display (enigma pi0 pi1 pi2 pir ip init plain crypt))))

(define s (open-input-file "5.tin.txt"));入力ファイル
(define n (read s))                     ;データの組の数
(do ((i 0 (+ i 1))) ((= i n)) (solve))  ;問題を解く
(newline) (close-input-port s)          ;ポートsを閉じる
```

もちろん出力データと一致している．

■本物のEnigmaとそのシミュレータ

問題のEnigmaの記述は本物のEnigmaとはかなり違う．もちろん本物のEnigmaにも多くの種類があるので一概にこうとは言えない．まずローターが（IからVの）5枚（または8枚）用意してある．また反射ローターは（bとcの）2枚ある．この中から挿入の順（左，中，右）に使用するローターと反射ローターを選ぶ．

前述のように本物のEnigmaでは反射ローターは回転しない．また問題のように，いずれの開始位置で始めても，

1周すると繰り上がりが出るのではなく，ローターごとに繰り上げを発生する位置（ノッチ位置）が決まっている．また面倒なことに，中のローターには構造に起因するダブルステッピング問題があった．

さらにローターはキーボードの押下げ力でまず回転し，回転後のローター位置で変換する．

図-4に示すように，ローターの回転は下からバネで押し上げられているT字バーによる．各ローターは右側には切り欠けが26個のラチェットがあり，そのローターに対応するT字バーがラチェットに噛んでいれば，キーボード押下げで動くT字バーにより，そのローターは1文字分回転する．一番右ローターは常にそのT字バーが噛んでいて，毎度回転する．各ローターの左側には1カ所だけ欠けたノッチがあり，ノッチがT字バーの上にきたとき，左隣りのローターに対応するT字バーはラチェットに噛み，左のローターを回転させる．したがって前回の回転の結果，一番右のローターのノッチが中ローターに対応するT字バーの上にきたとすると，次には中のローターも回転する．その結果中ローターのノッチが左ローターのT字バーの上にくると，このT字バーにより，中と左のローターが回転する．これをダブルステッピングという．10進法に例えると，088→089→090→101→102と進む．

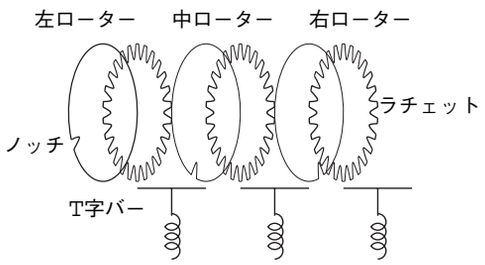

図-4 ダブルステッピングの説明

以下のシミュレータではノッチ位置を見るため，ローターのほかにインディケータを用意する．ローターの文字列の後の1文字はそのローターのノッチ位置を示す．ローター定義の先頭の () は0番（反射ローターではa）に対応するダミー．

```
(define rots '(() ("ekmflgdqvzntowyhxuspaibrcj" "q")    ;ローター1から5を定義
  ("ajdksiruxblhwtmcqgznpyfvoe" "e") ("bdfhjlcprtxvznyeiwgakmusqo" "v")
  ("esovpzjayquirhxlnftgkdcmwb" "j") ("vzbrgityupsdnhlxawmjqofeck" "z")))
(define refs '(() "yruhqsldpxngokmiebfzcwvjat" "fvpjiaoyedrzxwgctkuqsbnmhl"))
                                                    ;反射ローターb, cを定義
(define (turn ind) (set-cdr! (list-tail ind 25) (list (car ind))) (cdr ind))
(define (tn n i) (if (= n 0) i (tn (- n 1) (turn i))))  ;インディケータ回転

;前に書いたプログラムからrotate, rot, asciiint, intascii, alph, readstrを挿入

(define (enigsym)       ;シミュレータ本体
(let ((ip "abcdefghijklmnopqrstuvwxyz") (pi2 '()) (pi1 '()) (pi0 '())
      (pir '()) (ind0 alf) (ind1 alf) (ind2 alf) (notch0 '()) (notch1 '()))

  (define (advance)     ;ノッチ位置を見てローターを回転
    (define (adv0) (set! pi0 (rotate pi0)) (set! ind0 (turn ind0)))
    (define (adv1) (set! pi1 (rotate pi1)) (set! ind1 (turn ind1)))
    (define (adv2) (set! pi2 (rotate pi2)) (set! ind2 (turn ind2)))
    (cond ((equal? (car ind0) notch0) (adv0) (adv1))        ;pi0がノッチ位置
          ((equal? (car ind1) notch1) (adv0) (adv1) (adv2)) ;pi1がノッチ位置
          (else (adv0))))                                   ;その他

  (define (exchange p)  ;pは(a b)のようなもの ppは(#\a #\b) ipに使う
    (let ((pp (map (lambda (x) (string-ref (symbol->string x) 0)) p)))
    (string-set! ip (asciiint (car pp)) (cadr pp))
    (string-set! ip (asciiint (cadr pp)) (car pp))))

;conv, invcを挿入
```


Program Promenade Program Promenade Program Promenade

```
  (define (convert)              ;入力文字列を変換　"."で始まる行で終了
    (newline) (display "enter text: ") (define text (readstr))
    (if (string>? text ".")
      (begin (map (lambda (c)
        (advance)                                   ;変換前にローターを回転する
        (set! c (conv ip c))  (set! c (conv pi0 c))
        (set! c (conv pi1 c)) (set! c (conv pi2 c))
        (set! c (conv pir c))
        (set! c (invc pi2 c)) (set! c (invc pi1 c))
        (set! c (invc pi0 c)) (set! c (invc ip c))
        (display (intascii c))) (map asciiint (string->list text)))
        (convert))))

(newline) (display "left rotor: ") (define left (read)) ;左ローターを入力
(display "middle rotor: ") (define middle (read))       ;中ローターを入力
(display "right rotor: ") (define right (read))         ;右ローターを入力
(display "reflector: ") (define refl (readstr))         ;反射ローターを入力
(display "initial positions from left to right: ") (define init (readstr))
(display "list of plugged letter pairs: ") (define plugged (read))
(display (list left middle right refl init plugged))    ;確認のため入力を表示
(set! pi2 (car (list-ref rots left)))   ;ローター群からpi2を選ぶ
(set! pi1 (car (list-ref rots middle))) ;ローター群からpi1を選ぶ
(set! pi0 (car (list-ref rots right)))  ;ローター群からpi0を選ぶ
(set! pir (list-ref refs (asciiint (string-ref refl 0))))   ;反射ローター群からpirを選ぶ
(set! notch0 (string-ref (cadr (list-ref rots right)) 0))   ;ノッチ位置設定
(set! notch1 (string-ref (cadr (list-ref rots middle)) 0))  ;ノッチ位置設定
(map exchange plugged)  ;プラグボードの文字対からipを作る
(set! init (map asciiint (string->list init)))  ;初期位置を数値化
(newline) (display pi0) (newline) (display pi1) ;ローター類を表示
(newline) (display pi2) (newline) (display pir)
(newline) (display ip) (newline) (display notch0)
(newline) (display notch1) (newline) (display init)
(set! ind2 (tn (car init) ind2)) (set! pi2 (rot (car init) pi2))    ;初期回転
(set! ind1 (tn (cadr init) ind1)) (set! pi1 (rot (cadr init) pi1))  ;初期回転
(set! ind0 (tn (caddr init) ind0)) (set! pi0 (rot (caddr init) pi0));初期回転
(convert))                                      ;変換ルーチンを起動
'ok)

(enigsym)       ;シミュレータを起動
```

これは以下のように使う．(:までが入力要求，;より右は説明)

```
left rotor: 1                                   ;1番ローターを左へ
middle rotor: 2                                 ;2番ローターを中へ
right rotor: 3                                  ;3番ローターを右へ
reflector: b                                    ;bを反射ローターへ
initial positions from left to right: abw       ;初期位置を左からabw
list of plugged letter pairs: ((a b) (c d))     ;ab，cdをプラグボード接続
```

でaaaaaaaaaaaaaaaaを入力するとmlwicywkxqkroolcが得られ，同じ条件でmlwicywkxqkroolcを入力するとaaaaaaaaaaaaaaaaが得られる．

サーチエンジンで探すとEnigmaの解説やシミュレータがいくつも見つかる．いずれもダブルステッピングを正確にシミュレートする．実行結果もこのシミュレータと同じであった．

参考文献

1）Harris, R.（後藤安彦訳）: 暗号機エニグマへの挑戦，新潮文庫 ハ -32-1.
2）http://www.acm.inf.ethz.ch/ProblemSetArchive.html

（平成16年9月2日受付）