最大長方形の面積

和田 英一
IIJ技術研究所　wada@u-tokyo.ac.jp

■ 連載の始めに

この4月から「プログラム・プロムナード」という連載を始めることになった.

本年1月に開催したプログラミング・シンポジウム報告集の前書きにも書いたのだが，最近プログラムを書く機会が少なくなっているということを耳にする．またACM国際大学対抗プログラミングコンテスト，日本でのアジア地区予選では，アジア各地から参加する海外の大学に首位を奪われ続けている．これを放置しておくと，日本のプログラミングパワーはますます沈下するのではないかと懸念される.

古きよき昔はせっせとプログラムを書いたものである．そもそも各地にあった計算機はそれぞれが個性的であったから，プログラムを彼方から持ってくるということは考えもしなかった．他のセンターに欲しいプログラムがあれば，そのプログラムを調べ，自分用に書き直さなければならなかった．左様な面倒なことをするより，自ら考え，作ってしまう方が早かったし，楽しかった.

しかし事情は一転している．最近の学生はプログラムの移植ばかりに熱心だと嘆く教授がいる．同じアーキテクチャの計算機が孫悟空の分身のように散らばっているから，どこかに使えそうなプログラムが存在する機会，確率は激増している．移植しなければ損だという気運も分からぬではない.

一方，自分で書かないのはなぜか.

- プログラムを書くのは面倒だ.
- とにかく書き方が分からぬ.
- 周囲にプログラムを書こうという雰囲気がない.
- 就職してもプログラム技能は活かされそうもない.

まだあろうか.

だが，プログラミングは情報処理の基本である．情報処理学会誌では，当然のこととして，これまでプログラミングの奨励は特にはしてこなかった．今回，これらの反省としてプログラミングも取り上げたいというのが，連載の趣旨である.

いうまでもないが，プログラムを書くことには利点も多い.

- 面白いアルゴリズムが見出せる.
- すっきりしたシステムが設計できる.
- 自らの思考の正しさが納得できる.
- 計画通りにものが動くのを見るのは楽しい.

プログラミングはものつくりの第一歩であり，こういう喜びを経験できない人は気の毒である.

この連載では，以前Communications of the ACM誌に連載されていた，プログラミングパールス☆1を範にとり，いろいろな問題に対し，プログラムはいかに考えて書くべきかを解説することにしたい．その問題は，前述のACM国際大学対抗プログラミングコンテストから借りてこよう．これらの問題は限られた時間のなかで，与えられた複数のデータを入力し，期待された出力を得るプログラムがいくつ書けるかをみるためのもので，拙速プログラミングを奨励するようでもあり，必ずしも適切ではないが，コンテストとしてはやむを得まい．今回の連載には，学生諸君にACMプログラミングコンテストは少しも恐くないということを知ってもらい，大勢の挑戦者が現れて欲しいという願望もこめられている．

このコンテストについてここで簡単に説明する[1), 2)]．ACM国際大学対抗プログラミングコンテストは毎年開催される．チームは3名の学部学生で構成する．世界大会は3月頃に各地の予選を通過したチームを集めて開催されるが，日本国内で話題になるのは国内で開催されるアジア地区予選である．アジア地区予選といってもアジアの数カ所で開催され，最近の4年間はその1カ所が日本国内にある（1998年東京，1999年京都，2000年つくば，2001年函館）．アジア地区予選に参加できるのは各地でそれぞれ30チーム程度であり，このうち数チームは海外から遠征してくるので，国内からは20チームくらいしか参加できない．そのため国内からの参加チームを選抜する国内予選がまず行われる．したがってコンテストは

- 国内予選
- アジア地区予選

と2回ある．国内予選の問題は5題，アジア地区予選は8題となっているので，毎年13問題が新たに作成される．それを少しずつ連載で解説しよう．時にはこのコンテスト以外の問題もあるかもしれぬ（直前の函館大会の様子はhttp://www.fun.ac.jp/icpc/を見て欲しい．問題もそこから得られる）．

コンテストの公式言語はC，C++，Pascal，Javaとなっているが，この連載では主としてCを使う．ただ，アルゴリズムを明確にすべく，ANSI-C規格に厳密には従わないかもしれぬ．その他Lispを使うこともあろう．プロトタイピングにはLispが便利だからだ．周知のようにLispには方言が多く，使うLispも本来は決めておきたいが，方言というほどでSchemeを除いては相互にあまり違いはなく，現段階ではどのLispということは決めずに出発したい．

■ 問題: 最大長方形の面積

2001年度国内予選の問題Aは，5×5の桝目のそれぞれに1か0が書いてあり，その中から1だけでできている最大の長方形の面積を答える．

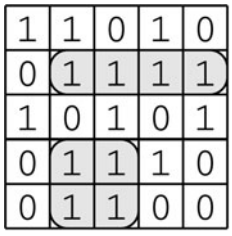


図-1

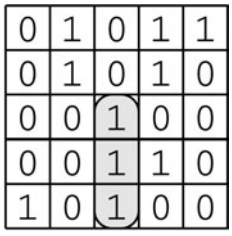


図-2

図-1なら4，**図-2**なら3が正解となる．
力づくで書くと

☆1 Jon Bentley: Programming Pearls（Addison-Wesley 1986, 野下浩平 訳:「プログラム設計の着想」近代科学社）．

```
#include <stdio.h>
main()
{int a[5][5];
 int i,j,k,l,x,y,z,w=0;
 for(i=0;i<5;i++)
  for(j=0;j<5;j++) scanf("%d",&a[i][j]);
 for(i=0;i<5;i++)
 {for(j=0;j<5;j++) printf("%2d",a[i][j]);
  printf("\n");}
 for(i=0;i<5;i++)
  for(j=0;j<5;j++)
   for(k=i;k<5;k++)
    for(l=j;l<5;l++)
    {z=0;
     for(x=i;x<=k;x++)
      for(y=j;y<=l;y++)
       if(a[x][y]!=0) z++;
       if(z == (k-i+1)*(l-j+1))
        if(z>w)w=z;}
printf("%d\n",w);}
```

となろう．これはforループが6重になっているので，$O(n^6)$ アルゴリズムという．5×5のデータ処理の時間に対し，10×10の処理時間は64倍になるわけだ．

■ ダイナミックプログラミングによるもの

通常，この種の問題はまずダイナミックプログラミングで解けないかと考える．つまり（2次元の問題なら）左上から右下へ個々の要素を順次に辿り，上隣と左隣の解の情報から，注目中の要素の解の情報を構成する．いいかえれば，ダイナミックプログラミングは配列の各要素につき，隣まで解けていたとして，その結果から注目している要素について解く．この方法については学会誌2002年1月号で阿久津君がDNA配列の比較の話題で述べている[3]が，DNA以前にそのアルゴリズムをテキストエディタで使っていた例[4]がある．

多少天降り的だが，図-1に対しては，**図-3**のような図を作りながら計算が進行するとする．

(11)	(21)		(11)	
	(12)	(21)	{12 31}	(41)
(11)		(12)		(12)
	(11)	{13 21}	(31)	
	(12)	{14 22}		

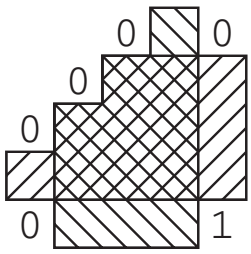
図－3

左上隅の(1 1)は，ここにできる最大の長方形は，x（左）方向に長さ1，y（上）方向にも長さ1であることを示す．その右の要素の(2 1)は，最大の長方形は，x方向に長さ2，y方向に長さ1である；さらに下へいくと(1 2)で，これはx方向が1，y方向が2の長方形ができるの意である．この要素から右へ2進むと(1 2)(3 1)とあり，図-1と見比べると，確かに上に長さ2の長方形(1 2)と，左に長さ3のもの(3 1)ができるらしい．この最後の例のように，要素によっては複数の可能性を持つものがあり，その後の処理にどの長方形が利用されるか未定なため，すべての可能性の情報を持ち歩くことになる．したがってこの情報はリストの形で保持する．

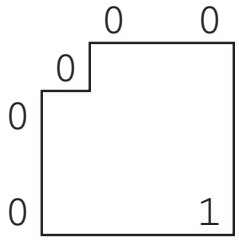
上隣と左隣の情報から自分の情報をどう作るかがこのアルゴリズムの中心だが，まず

• 注目点が0ならなにもしない．
• 注目点が1なら上隣と左隣の情報を見る．
  - 両隣がともに0なら自分だけで長方形を作るより仕方ないので(1 1)とする．
  - 上隣が0で左隣が1なら，上にはもう発展できないが，左に伸びられるから，
    （左方に続く1の個数 1）
    とする．
  - 左隣が0で上隣が1なら，左にはもう発展できないが，上に伸びられるから，
    （1 上方に続く1の個数）
    とする．
  - 左隣も上隣も1なら，両隣の候補のリストを参考にして新しい候補のリストを作る．それには図**-4**を見て考えよう．

いま右下の1と書いてある要素を処理しているとする．左隣は((1 5)(2 4)(3 3))上隣は((3 3)(4 2)(5 1))である．左隣は横に伸びたのでxの値をそれぞれ1増やす；しかし上方は0で押さえられるから，yの値は上方に1の続く個数，ここでは4を最大値にする．したがって((2 4)(3 4)(4 3))になる．同様に上隣のはyの値を1増やし，xの値は左方に1の続く個数，ここでは4を最大値にするから((3 4)(4 3)(4 2))となる． この両方をマージすると注目点の長方形候補のリスト((2 4)(3 4)(4 3)(4 2))ができる．



図－4



図－5

しかし，(2 4)は(3 4)に含まれるからこれは消す．(4 2)も同様．したがって((3 4)(4 3))ができる（図**-5**）．

このようにして，右下まで辿り，途中長方形の面積の最大値（これは(x y)のxとyの値の積）を更新していく．Lisp（UtiLisp）で書いたプログラムは次の通り．

```
(setq width 5 height 5)              ;配列のサイズ指定
(setq a (vector (* width height)))   ;図-1，図-2の配列に対応

(defun geta (i j)                    ;a[i,j]の0．1に従いnil，tを返す
  (and (<= 0 i) (<= 0 j) (= (vref a (+ (* i width) j)) 1)))
(defun getb (i j)                    ;b[i,j]のリストを返す
       (vref b (+ (* i width) j)))
(defun putb (i j v)                  ;b[i,j]=v (vはリスト)
  (vset b (+ (* i width) j) v)
  (mapcar v '(lambda (b) (setq max (max max (* (car b) (cadr b)))))))
                                     ;maxを更新する
(defun left (i j)                    ;左方に続く1の個数
 (1+ (apply 'max (mapcar (getb i (1- j)) 'car))))

(defun up (i j)                      ;上方に続く1の個数
 (1+ (apply 'max (mapcar (getb (1- i) j) 'cadr))))

(defun selec (b)                     ;重複するものを省く
 (cond ((null (cdr b)) b)
```

```
        ((some (cdr b)
            '(lambda (x) (and (<= (caar b) (car x)) (<= (cadar b) (cadr x)))))
         (selec (cdr b)))
        (t (cons (car b) (selec (cdr b))))))

(defun newb (bu bl u l)              ;新しいリストを作る
 (selec (append (mapcar bu '(lambda (x) (list (min l (car x)) (1+ (cadr x)))))
                (mapcar bl '(lambda (x) (list (1+ (car x)) (min u (cadr x))))))))

(defun area nil                      ;主プログラム
 (lets ((max 0) (b (vector (* width height))))  ;図-3の配列に対応
  (do ((i 0 (1+ i)))
      ((= i height))
      (do ((j 0 (1+ j)))
          ((= j width))
          (cond ((null (geta i j)))              ;a[i,j]とその上，左により仕事振分け
                (t (cond ((and (null (geta (1- i) j)) (null (geta i (1- j))))
                          (putb i j '((1 1))))
                         ((null (geta (1- i) j))
                          (putb i j (list (list (left i j) 1))))
                         ((null (geta i (1- j)))
                          (putb i j (list (list 1 (up i j)))))
                         (t (putb i j
                                  (newb (getb (1- i) j) (getb i (1- j))
                                   (up i j) (left i j)))))))))
  max))

(fill-vector a '(1 1 0 1 0  0 1 1 1 1  1 0 1 0 1  0 1 1 1 0  0 1 1 0 0))
           ;図-1 のデータを配列 a に置く
(area)     ;面積の計算 以下同様に4組のデータについて実行.
(fill-vector a '(0 1 0 1 1  0 1 0 1 0  0 0 1 0 0  0 0 1 1 0  1 0 1 0 0)); 図-2
(area)
(fill-vector a '(0 0 0 1 0  0 0 1 1 1  0 1 1 1 1  1 1 1 1 1  0 1 1 1 1)); 図-4
(area)
(fill-vector a '(0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0 0))
(area)
(fill-vector a '(1 1 1 1 1  1 1 1 1 1  1 1 1 1 1  1 1 1 1 1  1 1 1 1 1))
(area)
```

関数selecはリストbを貰い，リストの先頭の(x y)と後方の(x' y')でx≤x' ∧ y≤y'のものがあれば，先頭の要素を捨てる．

■ナノピコ教室流

プログラミングコンテストに出題する審判団も気がつかなかったようだが，この問題はbitのナノピコ教室に出題されていた[5),6)]．出題と解答は農工大（当時はもちろん東大の院生）の斎藤隆文君．

そこに辻さんという方の$O(n^2)$のプログラムが示してあり，なかなかよくできていると思うので，（元はBASICだが）Cに書き直して説明したい．

データが$n \times n$の配列のとき，プログラムを簡単にするため，周囲を0で囲む．そのため，配列boardは$(n+2) \times (n+2)$に用意する．また作業用に長さ$n+2$の1次元配列a,bも使う．

```
#include<stdio.h>
#define n 5                                          /* 盤面の大きさ */
main()
{int n1=n+1;
 int n2=n+2;
 int board[n2][n2];                                  /*(n+2)*(n+2) の board*/
 int a[n2],b[n2];                                    /* 補助配列 a, b*/
 int x,x1,xa,xb,y,ya,s,s0,r1,r2,r3,r4;               /* その他の変数 */
 for(x=0;x<=n1;x++)
 {a[x]=0;b[x]=0;board[x][n1]=0;board[0][x]=0;board[n1][x]=0;}    /* 作業場所初期化 */
 for(y=1;y<=n;y++)
  for(x=1;x<=n;x++)
   scanf("%d",&board[x][y]);                         /* データ入力 */
 for(y=1;y<=n1;y++)
 {for(x=1;x<=n1;x++)printf(" %d",board[x][y]);printf("\n");}     /* データ出力 */
 a[0]=n1;s0=0;
 for(y=1;y<=n;y++)                                   /*y のループ */
 {xa=0;xb=0;
  for(x=0;x<=n;x++)                                  /*x のループ */
  {x1=x+1;
   if(board[x1][y]==0)
   {a[x1]=y;b[x1]=0;xb=x1;}
   else if(b[x1]<xb)b[x1]=xb;
   if(board[x][y+1]==0)xa=x;                         /* 注目点 */
   if(a[xa]<a[x1])                                   /* 長方形を見つけた */
   {while(a[xa]<a[x1])
    {ya=a[xa];xa=b[xa];s=(y-ya)*(x-xa);              /* 長方形が出来たので面積計算 */
     if(s>s0){s0=s;r1=xa;r2=x;r3=ya;r4=y;}}          /* 面積の最大値と位置を記憶 */
    xa=x1;}}}
 printf("max= %d %d %d %d %d\n",s0,r1+1,r2,r3+1,r4);}    /* 最大面積と位置の出力 */
```

図-4のデータでやってみよう．配列boardは**図-6**のようになる．周囲の0は省いた．太線の壁はあとで説明する．

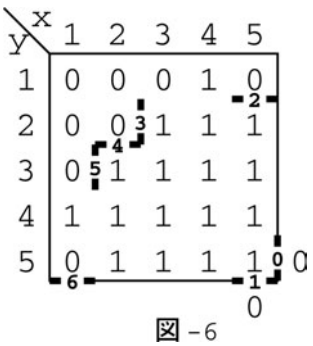


図-6

こうしておいてyを1からn，xを0からnまで繰り返す（したがってy=0の行は不要であった）．これでy=5，x=5でプログラムの注目点まで来た時，配列a，bは**図-7**のようになっており，x1=6，xa=5である．

```
                x x1
                ↓ ↓
 i    0 1 2 3 4 5 6
a[i] 6 5 2 1 0 1 5
b[i] 0 0 1 2 3 2 0
```

図-7

配列a，bの内容を調べてみよう．

a[i](0≤i≤x+1) には (i，y) から上で最も下の0のy座標が入っている．ただし，a[0]は特別で最初からn+1にしてある．

また b[i](0≤i≤x+1)はa[i]から左に眺め，初めてa[i]を超えるa[j]のjが入っている．つまりiの左で1の柱がどんと低くなるのはどこかという情報である．調べてみると，b[5]=2なのはa[5]=1の左で初めてこの1を超えるaはa[2]=2であり，b[4]=3はa[3]=0の左で初めてこの0を超えるaはa[3]=1だという意味で，確かにそうなっている．これはx=3，y=2にある1は横幅3の長方形の左の辺の候補になり得るし，x=2，y=3にある1も横幅4の長方形の左辺の候補になり得るということである．

xaはy+1の行の，i≤xで最も右の 0のx座標である．y+1は今は最下行，どこも0なので，xa=x=5.

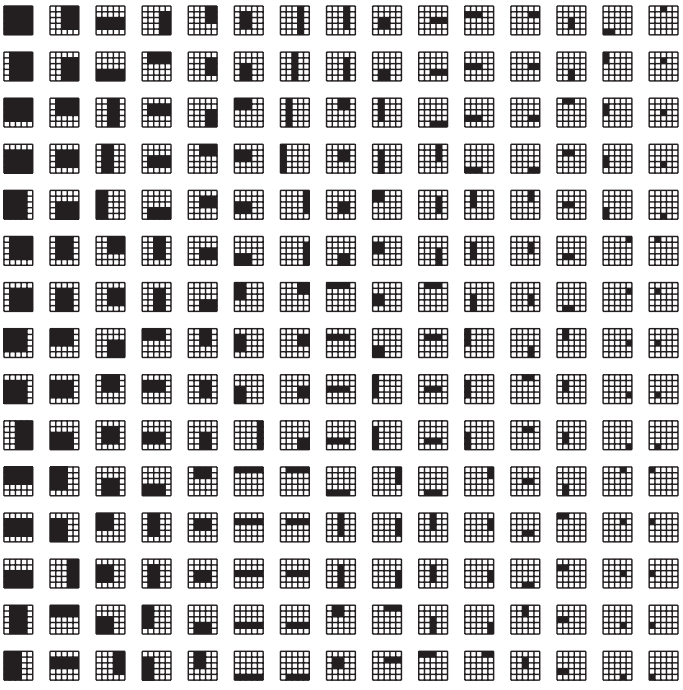
プログラムは上述の関係が成立するようにできている．

さて，「長方形を見つけた」のところだが，a[xa]<a[x1]はx1の左に1の長方形の壁が聳えているということである．図-6 でいうと縦壁0を見つけたのだ．a[xa]はx=xaの列の上の0の位置を示すから，横壁2も知れた．さらにb[xa]はxaの左でaの値が初めてa[xa]を超えた，つまり低い位置に0がある列の場所を示すから，縦壁3が分かる．これで長方形が1つ求まった．縦壁3の隣に横壁4があり，これは次の長方形の候補になり得る．そこでこの列を新たにxaとし，a[xa]<a[x1]を確認し，b[xa]から縦壁5を知り，次の長方形が求まる．その次のxaは1だが，この列の横壁は6で，a[xa]=a[x1]で低過ぎるから，xa=b[xa]によるbのリスト辿りはここで終了する．リストを辿るのにxaを使ってしまったので，whileループが済むと，xaを元に戻す．これは実によくできたプログラムである．

■もっと簡単にできないか

とりあえず解答するにはダイナミックプログラミングもよいが，5×5と大きさが決まっているなら別の解法はないかと考えるのは当たり前である．鶏を割くのに牛刀を用いるようなものだった．

1と0が5×5に並んでいるなら，25ビットの情報である．それなら1語に収まる．一方長方形のパターンはいくつあるか考えてみると，横方向は5の長さのもの1個，4の長さのもの2個，..., 1の長さのもの5個だから，15種類．縦方向も同じだから，225個のパターンを用意し，入力した問題のパターンで，ちょうど長方形のパターンの部分がすべて1のものを，面積の広い方から探すというのはうまくいきそうである．パターンを表示すると図-8のようになる．



図－8

図-9は最上段が読み込んだデータ；2段目と4段目が用意したテストパターンである．まず読み込んだデータとテストパターンのビットごと論理積をとり，パターンの1でない部分を切り落とす．次にパターンとビットごと排他的論理和をとり，パターンの部分がすべて1であったか調べる．その部分に0のものがあれば，この演算で1になり，最後の零のテストに引っ掛かるのである．

```
1010111011010 10
0000011111000 00  ∧
-----------------
0000011011000 00
0000011111000 00  ⊕
-----------------
0000000100000 00
```

図-9

Cのプログラムを示す．配列pには225個のパターンが面積の順に入る．配列qには対応する面積の値が入る．pのパターンは面積の順に入れなければならないが，ソートするまでもない．各面積の個数が分かっているので，その先頭番地を示す配列rに従って挿入する．rは次のように考えて作る．最後の部分和がrに逆順に入る．

面積	構成要素	個数		部分和
25	5×5	1×1	1	1
20	$4\times5, 5\times4$	$2\times1+1\times2$	4	5
16	4×4	2×2	4	9
15	$3\times5, 5\times3$	$3\times1+1\times3$	6	15
12	$3\times4, 4\times3$	$3\times2+2\times3$	12	27
10	$2\times5, 5\times2$	$4\times1+1\times4$	8	35
9	3×3	3×3	9	44
8	$2\times4, 4\times2$	$4\times2+2\times4$	16	60
6	$2\times3, 3\times2$	$4\times3+3\times4$	24	84
5	$1\times5, 5\times2$	$5\times1+1\times5$	10	94
4	$1\times4, 2\times2, 4\times1$	$5\times2+4\times4+2\times5$	36	130
3	$1\times3, 3\times1$	$5\times3+3\times5$	30	160
2	$1\times2, 2\times1$	$5\times4+4\times5$	40	200
1	1×1	5×5	25	225

```
#include<stdio.h>
main()
{int p[226],q[226],
  r[]={200,160,130,94,84,60,0,44,35,27,0,15,0,0,9,5,0,0,0,1,0,0,0,0};
 int x0,x1,y0,y1,i,j,k,l,m;
 for(i=0;i<226;i++)p[i]=0;
 q[225]=0;                          /* 読み込んだデータに 1 が皆無の時の用意 */
 for(x0=0;x0<5;x0++)                /*x の下端 */
  for(x1=x0+1;x1<6;x1++)            /*x の上端 */
   for(y0=0;y0<5;y0++)              /*y の下端 */
    for(y1=y0+1;y1<6;y1++)          /*y の上端 */
    {k=0;
     for(i=0;i<5;i++)for(j=0;j<5;j++)
      k=k*2+((x0<=i && i<x1 && y0<=j && j<y1)?1:0);
     l=(x1-x0)*(y1-y0);
     m=r[l-1];
     while(p[m]!=0)m++;             /* 空き地を探す */
```

```
    p[m]=k;
    q[m]=l;}                    /* ここまでp，qの初期設定 */
 scanf("%d",&l);                /* データの個数を読み込む */
 for(j=0;j<l;j++)
 {k=0;
  for(i=0;i<25;i++)
  {scanf("%d",&m);              /* 読み込んだデータを整数に変換 */
   k=k*2+m;}
  printf("m= %x ",k);           /* 読み込んだデータを印字 */
  for(i=0;i<226;i++)            /* 先頭からマッチしたものを探す */
  {if(((k & p[i]) ^ p[i])==0)
   {printf("area= %d\n",q[i]);
    break;}}}}
```

データファイルの例

```
5
1 1 0 1 0  0 1 1 1 1  1 0 1 0 1  0 1 1 1 0  0 1 1 0 0
0 1 0 1 1  0 1 0 1 0  0 0 1 0 0  0 0 1 1 0  1 0 1 0 0
0 0 0 1 0  0 0 1 1 1  0 1 1 1 1  1 1 1 1 1  0 1 1 1 1
0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0 0
1 1 1 1 1  1 1 1 1 1  1 1 1 1 1  1 1 1 1 1  1 1 1 1 1
```

pとqは実際には226個分とってある．これはデータが全部0の時を別扱いにせず，最後で捕まえるためのいわゆる番兵みたいなものである．

始めの2つのプログラムとこのプログラムの最大な違いは，このプログラムでは，最初にかなりの準備をしておくかわりに，実行速度が早いことである．これはどういう問題にでもいえそうだが，データの組が多ければ，そのそれぞれの組が短時間で処理できるように，時間をかけてもあらかじめ準備をするのは能率的だということである．プログラミングコンテストでは，多くのデータの組を処理する場合があるので，こういう解法も合理的といえよう．

いまから30年近く前に，オランダから（学会誌2002年2月号に「計算科学の終焉？」を寄稿した）Dijkstra氏が来日した．その折，彼が力説した「プログラミングに必要な条件」は，

1. 母国語のマスター，
2. Sense of Humor，
3. スタミナ

であった．Sense of Humorはむずかしいが，問題解決の発想を切り替える度量なのであろう．

参考文献
1）筧　捷彦: ACM国際大学対抗プログラミングコンテスト'98-'99アジア地区予選東京大会顛末記（前編）, bit, Vol.31, No.6, pp.62-67（June 1999）.
2）石畑　清: ACM国際大学対抗プログラミングコンテスト'98-'99アジア地区予選東京大会顛末記（後編）, bit, Vol.31, No.7, pp.50-57（July 1999）.
3）阿久津達也: ゲノム情報科学における情報科学的諸問題, 情報処理, Vol.43, No.1, pp.3-15（Jan. 2002）.
4）Gosling, J.: A Redisplay Algorithm, Sigplan Notices, Vol.16, No.6, pp.123-129（June 1981）.
5）斎藤隆文: 最大面積の空き領域, ナノピコ教室, bit, Vol.15, No.4, p.50（Apr. 1983）.
6）斎藤隆文: 最大面積の空き領域, ナノピコ教室解答, bit, Vol.15, No.7, pp.94-97（July 1983）.

（平成14年2月21日受付）