1998 年資訊奧林匹亞試題及參考解答(二)

何榮桂* 林順喜* 李忠謀* 孫春在**
*國立臺灣師範大學 資訊教育系
**國立交通大學 資訊科學系

一、第一天第三題：宴會燈

爲了使 IOI'98 盛大晚宴增添光彩，我們備有 N 盞燈泡，編號從 1 到 N。這些燈泡接到四個控制按鈕：

按鈕 1—當這個按鈕按下時，所有的燈泡改變它的狀態；也就是說，原來若是亮的就變成暗的；原來若是暗的就變成亮的。

按鈕 2—改變所有編號爲奇數的燈泡的狀態。

按鈕 3—改變所有編號爲偶數的燈泡的狀態。

按鈕 4—改變所有編號爲 3K+1 的燈泡的狀態($K \geq 0$)，亦即編號爲 1、4、7、…的燈泡。

有一個計數變數 C，用來記錄按鈕被按下的總次數。

當宴會開始時，所有燈泡均爲亮的。而計數變數 C 則設定爲 0。

任務(Task)

給定計數變數 C 之值，以及一部份燈泡最後的狀態資訊，請撰寫一程式來計算合乎上述條件之不同燈泡組態數。

輸入資料(Input Data)

檔名爲 PARTY.IN 的檔案含有四橫列，用以描述燈泡的個數 N、按鈕按下的次數 C、及部份燈泡最後的狀態。

第一橫列含有 N 之值。

第二橫列則爲計數變數 C 最後之值。

第三橫列——列出最後狀態必須爲亮著的燈泡編號，之間用一空白符號隔開，末尾用一整數-1 標示。

第四橫列——列出最後狀態必須爲暗著的燈泡編號，之間用一空白符號隔開，末尾用一整數-1 標示。

輸入範例(Sample Input)：

10
1
-1
7 -1

此範例中有 10 個燈泡，且只按下按鈕一次。編號爲 7 的燈泡其最後狀態爲暗的。

輸出資料(Output Data)

檔案 PARTY.OUT 必須含有所有燈泡之所有可能的最後的狀態(不允許重覆)。每一種可能狀態必須各自寫在不同的橫列上。這些狀態可用任意次序列出。

每一橫列含有 N 個字母，其中第 1 個字母表示編號爲 1 的燈泡狀態，第 N 個字母表示編號爲 N 的燈泡狀態。而字母爲 0 (zero) 表示那盞燈是暗的；字母爲 1 (one) 表示那盞燈是亮的。

輸出範例(Sample Output)：

0000000000
0110110110
0101010101

在此範例中，有三種可能的最後狀態：

1. 所有的燈泡爲暗的；
2. 編號爲 1、4、7、10 的燈泡爲暗的，且編號爲 2、3、5、6、8、9 的燈泡爲亮著；
3. 編號爲 1、3、5、7、9 的燈泡爲暗的，且編號爲 2、4、6、8、10 的燈泡爲亮著。

條件限制(Constraints)

參數 N 及 C 有下列限制：

$10 \le N \le 100$

$1 \le C \le 10000$

最後狀態被限定爲暗的燈泡的數量將小於或等於 2。

最後狀態被限定爲亮的燈泡的數量將小於或等於 2。

每一個輸入測試檔都至少會有一組最後狀態的解。

二、參考解答

試作者：鐘楷閔(台北市立成功高中三年級學生，IOI'98 銀牌獎)

/*因爲每個按鈕所產生的影響爲獨立的，且兩次爲一循環，我們可以知道其實這四個按鈕所能產生的樣本是不多的，只有 2^4 種。另外可發現這 16 種樣本的燈泡明暗狀態皆是六個爲一循環。因此甚至將這些樣本加以記錄、建表處理就可以解決這一題 。但我當時是用基本的 search 演算法加上重複盤面不往下展開的 backtracking 暴力的解題，以下是當時的程式碼：*/

```
#include<stdio.h>
#include<conio.h>

FILE *fp, *fi;
int used[5];
int map[101], n, c, on[2] = { -1, -1}, off[2] =
{ -1, -1};

void GetData( void )
{
      int k;
      fp = fopen( "party.in", "r+t");
      fi = fopen( "party.out", "w+t");
      fscanf( fp, "%d%d", &n, &c);
      c %= 4;
      if( c == 0) c = 4;
      k = 0;
```

```
        while( 1 )
        {
            fscanf( fp, "%d", &on[k]);
            if( on[k] == -1)
                break;
            k++;
        }
        k = 0;
        while( 1 )
        {
            fscanf( fp, "%d", &off[k]);
            if( off[k] == -1)
                break;
            k++;
        }
        for( int i = 1; i <= n; i++)
            map[i] = 1;
}

void Do( int kind )
{
    int i;
    switch( kind)
    {
        case 1:
            for( i = 1; i <= n; i++)
                map[i] = !map[i];
            break;
        case 2:
            for( i = 1; i <= n; i += 2)
                map[i] = !map[i];
            break;
        case 3:
            for( i = 2; i <= n; i += 2)
                map[i] = !map[i];
            break;
        case 4:
            for( i = 1; i <= n; i += 3)
                map[i] = !map[i];
            break;
    }
}

void Show( void )
{
    for( int i = 0; i < 2; i++)
        if( on[i] == -1) break;
        else if( map[on[i]] != 1) return;
    for( i = 0; i < 2; i++)
        if( off[i] == -1) break;
        else if( map[off[i]] != 0) return;
    for( i = 1; i <= n; i++)
        fprintf( fi, "%d", map[i]);
    fprintf( fi, "\n");
}

void Count( int index, int min)
{
    if( index == c || ( index < c && index % 2
== c % 2))
        Show();
    for( int i = min; i <= 4; i++)
        if( used[i] == 0)
        {
            Do( i );
            used[i] = 1;
            Count( index + 1, i + 1);
            Do( i );
            used[i] = 0;
        }
}

int main( void )
{
    GetData();
    Count(0, 1);
    fclose( fp );
    fclose( fi );
    return 0;
}
```

三、第二天第一題：圖片問題

牆壁上貼有許多長方形的海報、相片等長方形的圖片。圖片各邊均爲垂直或水

平。圖片的部份或全部可能被其他圖片所遮蓋。所有長方形圖片聯集體的內外框長度和，稱爲總周長(perimeter)。

Task（任務）

寫一程式計算總周長。

圖 1 顯示的例子中有 7 個長方形圖片。

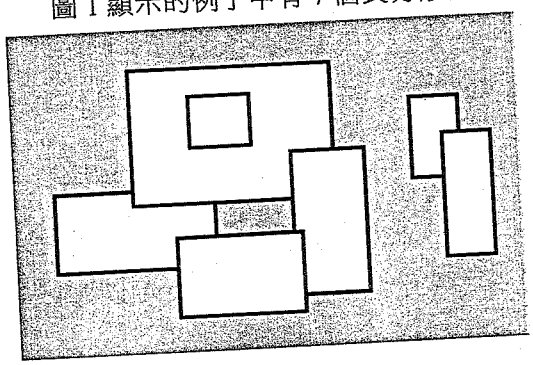

圖 1. 七個長方形圖片

其相對應的內外框即爲圖 2 中所有線段的集合。

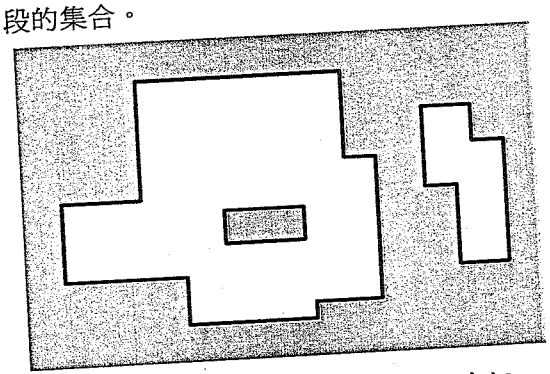

圖 2. 長方形圖片聯集體的內外框

所有長方形的端點座標均爲整數。

Input Data（輸入資料）

檔案 PICTURE.IN 的第一橫列是牆上所貼的長方形圖片總數。之後每一橫列是一個長方形的左下角與右上角的整數座標。各座標的 x 值在前，y 值在後。

Sample Input（範例輸入）

```
7
-15 0 5 10
-5 8 20 25
15 -4 24 14
0 -6 16 4
2 15 10 22
30 10 36 20
34 0 40 16
```

本範例相對應圖 1 所示的各長方形圖片。

Output Data（輸出資料）

檔案 PICTURE.OUT 中應有一非負整數，即爲長方形圖片聯集體的總周長。

Sample Output（範例輸出）

```
228
```

此值爲上述範例的總周長。

Constraints（限制）

0 ≤ 長方形圖片數 < 5000

各座標值均在[-10000,10000]範圍內，且任一長方形圖片之面積均爲正值。

輸出數值有可能需要用到 32 位元數值表示法(32-bit signed representation)。

四、參考解答

試作者：吳光哲（台北市立建國中學三年級學生，IOI'98 銀牌獎）

/*這題算是今年 IOI 六題中最難的一題，我及另外三名隊友都只答對前兩筆 test data。這裡所引用的演算法與程式都是大會提供的參考解答，相關資料請見:

http://olympiads.win.tue.nl/ioi/ioi98/contest/picture

這題是屬於幾何演算法的題目，雖然本題

對於每一組測試資料都只有唯一解,但解法相當多種, 大家有機會可以試著寫寫看。解法很多, 在這邊我提出我覺得以前比較少見的演算法:在這個演算法中, 不只是輸入的矩形要被討論, 兩個矩形重疊的交集, 也要視爲一個新的矩形。這麼做雖然有可能會產生多達 2^n-1 個矩形, 但一般來說, 這種情形並不會太嚴重。

根據排容原理:

| S1 ∪ S2 ∪ S3 ∪ ... ∪ Sn |
= | S1 | + | S2 | + ... + | Sn |
- | S1∩S2 | - | S1∩S3 | - .....
+ | S1∩S2∩S3 | + | S1∩S2∩S4 | + ...
- .....
+((-1)^(m+1)) | S1∩S2∩.....∩Sn |

全部矩形聯集的總周長
= 每個矩形的周長 - 任兩矩形交集的周長
+ 任三矩形交集的周長 - .....
+ (-1)^(m+1)*全部矩形交集圖形的周長

又無論多少矩形的交集, 都將還是矩形, 因此可以利用類似動態規劃的手法, 利用 k-1 個矩形的交集得到 k 個矩形的交集。 */

```
/* Author: Michel Wermelinger
(mw@di.fct.unl.pt) */
#include <stdio.h>
#define max(a, b) ((a) > (b) ? (a) : (b))
#define min(a, b) ((a) > (b) ? (b) : (a))

typedef struct {
  char sign;
  short int x, y, X, Y;
  /* sign 1,-1 表示奇或偶數個矩形的交集,
      x,y,X,Y 表示矩形左下與右上角的座
標 */
} pict;

#define MAX_PICT 7000
/* 最多幾塊矩形（含交集的），須 9*
MAX_PICT bytes */
pict p[MAX_PICT];

void intersect (pict p1, pict p2, pict *p3) /* 將
p1,p2 兩矩形的交集存於 p3 */
{
  p3->x = max(p1.x, p2.x); p3->y = max(p1.y,
p2.y);
  p3->X = min(p1.X, p2.X); p3->Y =
min(p1.Y, p2.Y);
  p3->sign = -p1.sign * p2.sign;
}

int main (void)
{
  long perim = 0;
  int npf, npt = 0; /* 分別表示輸入的矩形
數與計算時實際的矩形數量 */
  FILE *f;
  int i, j, k;

  if ((f = fopen("PICTURE.IN", "r")) ==
NULL) {
    printf("Empty file!\n");
    return 0;
  }

  fscanf (f, "%d\n", &npf);
  for (i = 0; i < npf ; i++) {
    k = npt++;
    fscanf(f, "%d %d %d %d\n", &p[k].x,
&p[k].y, &p[k].X, &p[k].Y);
    p[k].sign = 1;
    for (j = 0;  j < k; j++)  {
      intersect  (p[j] , p[k] , &p[npt]);
      if ((p[npt].X > p[npt].x && p[npt].Y
>= p[npt].y) ||
          (p[npt].X == p[npt].x &&
```

（下轉 52 頁）