第九屆(1997)年國際資訊奧林匹亞競賽試題分析(一)

何榮桂
國立臺灣師範大學　資訊教育系
龔律全
國立臺灣大學　資工系
陳康本
國立臺灣大學　電機系

一、前言

國際資訊奧林匹(IOI)競賽，分兩天比賽，每天比賽時間連續五個小時，解三個題目。比賽前一天各國領隊人員入闈修改題目，直到都沒有異議，再由各國領隊人員譯成自己國家的文字。直到第二天比賽開始一小時後，始可出闈。

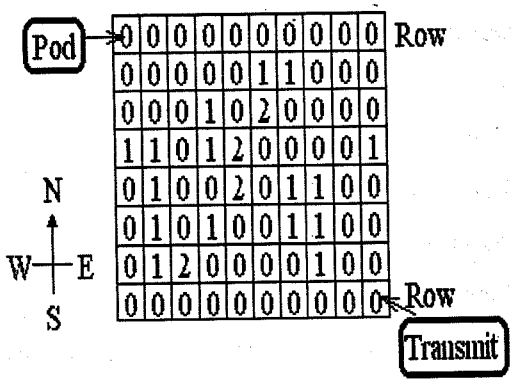
比賽結束後，再由主辦國之 Scientific Commettee 陪同考生及領隊人員逐一線上評分，每一考生所完成的程式通常要經過幾組 test data 的檢驗，才能決定得分。考生及領隊人員在評分過程中也可爭取分數，但最後的裁決由 Scientific Commette 決定。

以下即爲 1997 年的中譯(意譯)考題中的兩道題目，與其參考解答及全體考生得分分配。提供有興趣的同學參考。第一天的考題(前三題)由 1997 年我國 IOI 代表之一陳康本同學(獲得銀牌，現就讀台灣大學電機系一年級)試做，第二天的考題(後三題)則由龔律全同學(獲得金牌，現就讀台灣大學資工系一年級)試做。因爲題目及解答均甚長，因此分期(共 6 題，每期 2 題)刊登。參考解答未必是最佳解，有興趣的同學也可嘗試用自己的方法解解看。

二、第一天試題

1. 第一題題目:MARS 火星探測(MARS:Mars explorer)

未來火星探測執行任務方法如下，首先在表面某個叫 POD 的定位放下一些火星探測車(MEV)，每台 MEV 由 POD 出發，朝表面另一個叫 TRANSMITTER 的點行進，探測路途中必須採集沿路經過的岩石標本，同一岩石標本只可被一探測車採集一次，岩石標本被採集後，某他 MEV 能通過原來岩石標本所在位置。探測車不可經過某些不良表面(ROUGH TERRAIN)，只能往南垂直或往東水平沿方格線(GRID PATTERN)由 POD 往 TRANSMITTER 行進，同一時間同一地方可以有兩部 MEV。



警告：

若MEV無法由POD到達TRANSMITTER，則其目前所採集所有岩石標本完全無效，且其他 MEV 也不可以再採集這些岩石標本。

TASK：

根據題最後的計分公式(以採集到TRANSMITTER 的岩石標本個數，到達TRANSMITTER 的 MEV 數套入公式計分)找出 MEV 的行進次序，使得該函數計分最大。

INPUT：

火星上以 PxQ 方格表示的可行進表面，每個方格點上可以－可走方格點：以 0 代表；不良表面方格點：以 1 代表；岩石標本：以 2 代表。

輸入 file 格式：

MEV 探測車車輛數目：

P

Q

(x1,y1) (x2,y1) ..... (xp,y1)

(x1,y2) (x2,y2) ..... (xp,y2)

(x1,yq) (x2,yq) ..... (xp,yq)

P 和 Q 爲火星表面方格長寬大小，車輛數<1000，Q 代表方格列數，每一列有 P 格，P 和 Q 不會超過 128。

範例輸入：

MARS.DAT	解釋
2	車輛數
P	
Q	
:	:

OUTPUT：

一行一行，每行代表 MEV 車往 TRANSMITTER 的行進次序，每一行含一車輛編號和一個 0 或 1 的數字，0 代表往南，1 代表往東走一格

範例輸出：

MARD.OUT	解釋
1 1	1 號車往東
1 0	1 號車往南
2 1	2 號車往東
2 0	2 號車往南

上例中，TRANSMITTER 共到達 2 輛 MEV，共收集了 3 個岩石標本，依下列公式得 5/5 = 100% 分。

計分公式

公式以 TRANSMITTER 共到達 MEV 輛數及收集岩石標本總數爲依據：若輸出

不合法的 MOVE，則得 0 分，不合法的 MOVE 代表移動到 ROUGH TERRAIN 或起出方格點，分數 = (TRANSMITTER 收集到的岩石標本數 + 到達 MEV 車數 − 未到車數) 除以 (上述可以最高數字) 的百分比數，分數最高 100%，最低 0%。

2. 參考解答

/*我所用的演算法爲 Greedy Method (貪心)，每次我都會找一條路能採集最多岩石，但此演算法在沒有障礙物時才能得到最佳解，在有障礙物時，得出來的解並不一定是最佳解，但我相信是一組不錯的解答，於是我採用這個方法，在五組測試後，得到 99%，有的同學和我採用一樣的演算法卻得到 100%，是因爲他們考慮「東、南」的順序和我不同，我先考慮南再考慮東，他們是先考慮東再考慮南，所以得到 100%，這是 depend on test data，所以基本上，大多數的人都採用這個演算法，而這題也是得分最容易的一題*/

```
#define NDEBUG
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#define Max_car 1000
#define Max 128
#define Max_size 16384
#define Rock 2
#define Gotten 4
#define Bad 1
#define Left 1
#define Up 2
#define NoDirec 4
#define Max_clock 330
clock_t start ;
int ncar, nrow, ncol, rowrock[Max], pathlen ;
char map[Max][Max], path[Max_size] ;

void Getdata(void)
{
    FILE *fi ;
    int r, c, k ;
    fi = fopen("mars.dat", "r") ;
    assert(fi) ;
    fscanf(fi, "%d", &ncar) ;
    fscanf(fi, "%d", &ncol) ;
    fscanf(fi, "%d", &nrow) ;
    for(r=0; r<nrow; r++)
        for(c=0; c<ncol; c++) {
            fscanf(fi, "%d", &k) ;
            map[r][c] = (char)k ;
        }
    fclose(fi) ;
}

void OutputPath(int car)
{
    int i, direc ;
```

```
    static FILE *fo=NULL ;
    if(!fo) {
      fo = fopen("mars.out", "w") ;
      assert(fo) ;
  }

  if(car==0) return ;

  for(i=pathlen-1; i>=0; i--) {
    if(path[i] == Left) direc = 1 ;
    else if(path[i] == Up) direc = 0 ;
    else assert(0) ;
    fprintf(fo, "%d %d\n", car, direc) ;
  } }

  void Solve(void)
  {
  int getnum[Max][Max], r, c, car, nextr,
    nextc ;
  int mr[] = {0, 0, -1} ;
  int mc[] = {0, -1, 0} ;
  char direc[Max][Max] ;
// 每部車都跑一次
  for(car=1; car<=ncar; car++) {
    for(r=0; r<nrow; r++) {
      memset(&getnum[r][0], 0,
        sizeof(int)*Max) ;
      memset(&direc[r][0], 0,
        sizeof(char)*Max) ;
    }
    direc[0][0] = NoDirec ;
// 用兩輛迴圈算出一條可拿到最多岩石的
  path，用的是 Dynamic Programming
  for(r=0; r<nrow; r++) {
    for(c=0; c<ncol; c++) {
      if(map[r][c] == Bad) continue ;
      getnum[r][c] = 0 ;
      if(r > 0) {
  if(map[r-1][c] != Bad && direc[r-1][c]) {
    getnum[r][c] = getnum[r-1][c] ;
    direc[r][c] = Up ;
          } }
          if(c > 0) {
  if(getnum[r][c-1] > getnum[r][c]
    || !direc[r][c]) {
    if(map[r][c-1] != Bad && direc[r][c-1]) {
      getnum[r][c] = getnum[r][c-1] ;
      direc[r][c] = Left ;
          } } }
          if(map[r][c] == Rock)
            getnum[r][c]++ ;
      } }
    if(direc[nrow-1][ncol-1] != Left &&
      direc[nrow-1][ncol-1] != Up) {
  #ifndef NDEBUG
      printf("No Path\n") ;
  #endif
      break ;
    }
  // 回溯出這條路徑，並將它輸出到檔
```

案裡

```
    pathlen=0 ;
    for(r=nrow-1, c=ncol-1; r!=0 || c!=0;) {
        if(map[r][c] == Rock)
            map[r][c] = Gotten ;
        path[pathlen++] = direc[r][c] ;
            nextr = r+mr[direc[r][c]] ;
            nextc = c+mc[direc[r][c]] ;
            r = nextr ;
            c = nextc ;
        }
        OutputPath(car) ;
        if( (clock()-start) > Max_clock)
            exit(0) ;
    }   }

    int main()
    {
    start = clock() ;
    Getdata() ;
    pathlen = 0 ;
    OutputPath(0) ;
    Solve() ;
    fcloseall() ;
    return 0 ;
    }
```

3. 參賽者在 MARS 之得分分布

如表一顯示，全體考生（221 人）在 MARS 得分之分布呈兩極化，約有 1/4(25.34%)的考生得分在 0～10 之間，而有 41.18%的考生得分在 91～100 之間。整體而言，此題屬於較簡單的題目。

表一　參賽者在 MARS 之得分分布表

分數	人(次)數	累積人(次)數	百分比	累積百分比
0-10	56	56	25.34	25.34
10-20	7	63	3.17	28.51
21-30	4	67	1.81	30.32
31-40	7	74	3.17	33.48
41-50	5	79	2.26	35.75
51-60	11	90	4.98	40.72
61-70	6	96	2.71	43.44
71-80	27	123	12.22	55.66
81-90	7	130	3.17	58.82
91-100	91	221	41.18	100.00

4. 第二題題目：TOXIC 有毒蟲 (Toxic:The Toxic iShongololo)

iShongololo 是祖魯(Zulu)族對一種長有黑色光澤的多足蟲的稱呼。iShongololo 吃長爲 L，寬爲 W，高爲 H 的實心立方體食物 (L,W,H 均爲整數) 。

Task：

在滿足以下限制條件(constraints)下，找出 iShongololo 所能吃最多的單位正立方體(block)個數，程式必須輸出 iShongololo 在食物內吃 blocks 及移動的過程。iShongololo 由食物外開始行動。首先吃位於(1,1,1)的 block，然後移動到此一被吃掉的空間。它在無法合法吃 blocks 且無法合法移動時停止。

限制條件：

一隻 iShongololo 大小只佔一單位立方體空間。iShongololo 一次只吃一個 block。iShongololo 不可以移動到以前曾經所在過的任何位置(也就是說，不可以後退，也不可以移動路線相交)。iShongololo 不可以移動到實心(未吃)的部分，也不可以移動到食物(fruit)之外。iShongololo 只可以移動或吃掉和它目前所在位置相接一個面(face)的 block 上。除此之外，被吃掉的 block 除了和目前所在位置相接的一個面外，不可以有其他面和以前吃掉過的 block 相接。

Input：

程式輸入 L(長)，W(寬)，H(高)三整數，L,W,H 各佔一行。L,W,H 大小在 1 和 32 之間(可以是 32)。

範例輸入：

TOXIC.DAT

2　L(長)爲 2

3　W(寬)爲 3

H(高)爲 2

Output：

一行行，每行開頭是一個英文字母。此字母爲 E(代表吃)或 M(代表移動)。後面再接 3 個整數，分別代表被吃或移動到 block 所在的 L,W,H。

以下範例爲上述範例輸入的一組合法解。合法解代表在不違反限制條件下的動作過程。合法解不一定代表所吃 block 數就一定最多(optimal，即最佳解)。

範例輸出(以下也許不是最佳解)：

TOXIC.OUT

E 1 1 1　吃在(1,1,1)上的 block

M 1 1 1　移動到(1,1,1)

E 2 1 1　吃在(2,1,1)上的 block

E 1 1 2　吃在(1,1,2)上的 block

E 1 2 1

M 1 2 1

E 1 3 1

M 1 3 1

E 2 3 1

E 1 3 2

M 1 3 2

評分方式

若 iShongololo 違反任一限制條件，則 0 分。(總共吃的 blocks 數) 除以 (最多能吃的 blocks 數) 的百分比，爲程式所得分數。程式最多得 100 分 (亦即是 100%)。

5. 參考解答

/*我所用的演算法爲 Greedy Method (貪心)，在五組測試資料中，得了 63 分，大約是中上的成績，但還有更好的方法。由於時間緊迫，所以這個程式寫得並不是很好，有興趣的人，可以自己撰寫一次*/

```
#define NDEBUG
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#define Max 32
#define Eaten 1
#define Disable 2
```

```
//原本想做 k-degree 的 Greedy Method，
但是有實作上的困難，於是設 Degree=1
#define Degree 1
#define Eat_f 2
#define Move_f 1
int bx, by, bz, nowx, nowy, nowz ;
int lasteatx, lasteaty, lasteatz, final ;
char state[Max+2][Max+2][Max+2] ;
FILE *fo ;

void Getdata(void)
{
      FILE *fi ;
      fi = fopen("toxic.dat", "r") ;
      assert(fi) ;
      fscanf(fi, "%d", &bx) ;
      fscanf(fi, "%d", &by) ;
      fscanf(fi, "%d", &bz) ;
      fclose(fi) ;
}

void Eat(int x, int y, int z)
{
      fprintf(fo, "E %d %d %d\n", x, y, z) ;
      state[x][y][z] = Eaten ;
      lasteatx = x ;
      lasteaty = y ;
      lasteatz = z ;
      final = Eat_f ;
}

void Move(int x, int y, int z)
{
      fprintf(fo, "M %d %d %d\n", x, y, z) ;
      nowx = x, nowy = y, nowz = z ;
      final = Move_f ;
}

int CanEat(int x, int y, int z)
{
int face=0 ;
if(state[x][y][z]==Eaten) return 0 ;
if(x>1 && state[x-1][y][z]==Eaten) face++ ;
if(y>1 && state[x][y-1][z]==Eaten) face++;
if(z>1 && state[x][y][z-1]==Eaten) face++;
if(x<bx && state[x+1][y][z]==Eaten) face++;
if(y<by && state[x][y+1][z]==Eaten) face++;
if(z<bz && state[x][y][z+1]==Eaten) face++;
      return face <= 1 ;
}

void Eatadj(void)
{
if(nowx > 1 && CanEat(nowx-1, nowy, nowz))
      Eat(nowx-1, nowy, nowz) ;
if(nowy > 1 && CanEat(nowx, nowy-1, nowz))
      Eat(nowx, nowy-1, nowz) ;
if(nowz > 1 && CanEat(nowx, nowy, nowz-1))
      Eat(nowx, nowy, nowz-1) ;
if(nowx < bx && CanEat(nowx+1, nowy, nowz))
            Eat(nowx+1, nowy, nowz) ;
if(nowy < by && CanEat(nowx, nowy+1, nowz))
```

```
        Eat(nowx, nowy+1, nowz) ;
if(nowz < bz && CanEat(nowx, nowy, nowz+1))
        Eat(nowx, nowy, nowz+1) ;
        }

int AdjCanEat(int x, int y, int z)
{
        int face = 0 ;
        if(x>1 && CanEat(x-1, y, z)) face++ ;
        if(y>1 && CanEat(x, y-1, z)) face++ ;
        if(z>1 && CanEat(x, y, z-1)) face++ ;
        if(x<bx && CanEat(x+1, y, z)) face++ ;
        if(y<by && CanEat(x, y+1, z)) face++ ;
        if(z<bz && CanEat(x, y, z+1)) face++ ;
        return face ;
}

int Eatnum(int x, int y, int z, int depth)
{
        int max ;
        char record = state[x][y][z] ;
        if(state[x][y][z] != Eaten) return 0 ;
        if(depth >= Degree) {
                max = AdjCanEat(x, y, z) ;
        }
        state[x][y][z] = record ;
        return max ;
}

int FindMove(int *px, int *py, int *pz)
{
int max = 0, f ;
if(nowx>1)
   if( (f=Eatnum(nowx-1, nowy, nowz, 1)) > max)
     max = f, *px=nowx-1, *py=nowy, *pz=nowz ;
if(nowy>1)
if( (f=Eatnum(nowx, nowy-1, nowz, 1)) > max)
     max = f, *px=nowx, *py=nowy-1, *pz=nowz ;
      if(nowz>1)
if( (f=Eatnum(nowx, nowy, nowz-1, 1)) > max)
     max = f, *px=nowx, *py=nowy, *pz=nowz-1 ;
      if(nowx<bx)
if( (f=Eatnum(nowx+1, nowy, nowz, 1)) > max)
     max = f, *px=nowx+1, *py=nowy, *pz=nowz ;
      if(nowy<by)
if( (f=Eatnum(nowx, nowy+1, nowz, 1)) > max)
     max = f, *px=nowx, *py=nowy+1, *pz=nowz ;
      if(nowz<bz)
if( (f=Eatnum(nowx, nowy, nowz+1, 1)) > max)
     max = f, *px=nowx, *py=nowy, *pz=nowz+1 ;
      return max > 0 ;
}

void Solve(void)
{
        int found, x, y, z ;
// initial 初始化
        fo = fopen("toxic.out", "w") ;
        nowx = nowy = nowz = 1 ;
        Eat(1, 1, 1) ;
        Move(1, 1, 1) ;
        while(1) {
```

```
// 將相鄰的全部吃掉
        Eatadj() ;
// 用 Greedy Method 找到哪一 move 是較好的
        found = FindMove(&x, &y, &z) ;
        if(!found) break ;
        Move(x, y, z) ;
    }
// 若最後一個命令是 Eat, 則再 Move 到最後一個位置
    if(final == Eat_f)
      Move(lasteatx, lasteaty, lasteatz) ;
}

int main()
{
    Getdata() ;
    Solve() ;
    fcloseall() ;
    return 0 ;
}
```

6. Toxic 之得分分布

由表二知，此題偏難，約有 37.56% 的考生得分在 0～10 之間，而得分在 91～100 者僅佔 7.24%。

表二　參賽者在 Toxic 之得分分布表

分數	人(次)數	累積人(次)數	百分比	累積百分比
0-10	83	83	37.56	37.56
10-20	17	100	7.69	42.25
21-30	8	108	3.62	48.87
31-40	17	125	7.69	56.56
41-50	5	130	2.26	58.82
51-60	16	146	7.24	66.06
61-70	17	163	7.69	73.76
71-80	14	177	6.33	80.09
81-90	28	205	12.67	92.76
91-100	16	221	7.24	100.00

(上接 72 頁)

偉民先生：

來函的解釋，實際上已在 195 期 50 頁及 51 頁回答，和你所說的是一致。原 50 頁 2(d)倒數第三行中間，「重」鉻酸根...（即向右），「重」爲誤打，應予以刪去，而倒數 1 行及 2 行的（亦可參考 3b 之結果），即 51 頁中間(b)前二式，（即你在上文所書寫 1(b')，計算所得 pH=4.20 加以闡釋。如果再仔細品味 50 頁(d)所寫，「此答案，似乎相當奇特」已回答了，本題「暗藏玄機」的地方了！

化學奧林匹亞工作小組　方泰山