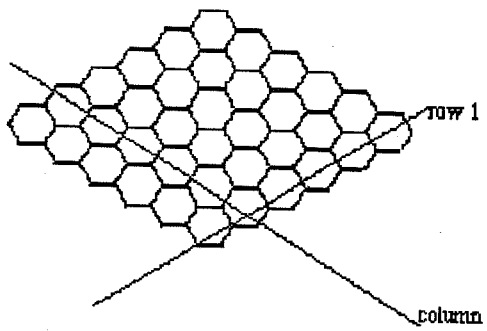
第九屆(1997年)國際資訊奧林匹亞競賽試題分析(二)

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

1. 第一天第三題題目：Hex 六邊棋遊戲(Hex:The Game of Hex)

這個遊戲的目的是第一位棋士(first player)將行 1(column1)中一個屬於他六邊棋子(Hex counter)連接到行 N(column N)中一個屬於他的六邊棋子。

六邊棋遊戲規則(Rules of Hex)：



六邊棋(Hex)是兩個人在用 NxN 個六邊形(hexagons)組成的菱形(rhombus)棋盤上玩的兩人下棋遊戲。圖例中 N=6。

對奕的兩位棋士(player)分別爲你的程式(your program)及評分程式集(evaluation library)。你的程式每次都先下第一步。兩位棋士輪流將自己的六邊棋子(Hex counters)放在棋盤上。每個六邊棋子(Hex counter)只能放在棋盤上空的位置上，假若兩組六邊形(hexagons)有一邊相鄰(share an edge)則這兩個六邊形相鄰(adjacent)。屬於同一棋士的六邊棋子如果擺在相鄰的六邊形(adjacent hexagons)上，則這兩個棋子就是相連接著的(connected)。

連接性(connectivity)是 transitive 和 commutative：換句話說：如果棋子 1 連接到棋子 2，而棋子 2 連接到棋子 3，則棋子 3 就連接到棋子 1，且棋子 1 也連接到棋子 3。

Task：

你必須寫一程式來下六邊棋(Game of Hex)，先下的棋士(亦即是你的程式)的目標是將行 1(column1)中的一個屬於你的六邊棋子(hex counters)一直連到行 N(column N)中另一個屬於你的六邊棋子。另一位棋子(亦即是評分程式集)的目標則是將列 1(row 1)中的一個屬於他的六邊棋子一直連到列 N(column N)中另一個屬於他的六邊棋子。假如你的程式採取最佳下法，則每次都會贏。

Input and Output：

你的程式**不可**讀入或寫出至任何檔案。**不可**從鍵盤輸入任何資料，也不可輸出任何結果(output)至螢幕(screen)。輸入是由 HexLib 程式集內的函數(function)來提供。這個程式集會自動產生一個檔案名爲 HEX.OUT 的輸出檔，你可以不必理會這個檔的內容。

在遊戲剛開始時，你的程式將被給定一個盤面，上面可能會已有一些六邊棋子(Hex counters)，但先下棋者仍然可以贏棋。你的程式必須用 GetMax 及 LookAtBoard 函數來了解棋盤盤面狀態。在剛開始時，盤面上屬於你的程式的六邊棋子的數目和評分程式集的六邊棋子的數目**相等**。

限制條件：

棋盤大小一定介於 1 跟 20 之間(包括 1 和 20)，你的程式最多有可能需下到 200 步才結束。整個程式運作須在 40 秒內完成，評分程式集(evaluation library)運作的時間將一定不會超過 20 秒。

Library：

你必須將 HexLib 這個程式集 link 到你的程式碼內。在題目的目錄中將有範例檔案教你如何做到。這些範例檔案分別是 TESTHEX.CPP , TEXTHEX.C , TEXTHEX.PAS , TEXTHEX.BAS。存在於 HexLib 的函數分別爲 (先 Pascal, C/C++, 再 Basic)function LookAtBoard(row, column : integer) : integer ;int LookAtBoard(int row, int column) ;declare function LookAtBoard cdecl(byval x as integer, byval y as integer) ;

傳回值：

- 1　若 row<1 或 row>N 或 column<1 或 column>N

  若此棋盤位置上無任何六邊棋子；

  若此棋盤位置上有一個你的六邊棋子(player 1)；

  若此棋盤位置上有一個評分程式集(evaluation library)的六邊棋子(player 2)。

procedure PutHex(row, column : integer) ;

void PutHex(int row, int column) ;

declare sub PutHex cdecl(byval x as integer, byval y as integer)

假若指定的棋盤位置是空的，則將你的一個六邊棋子放到指定的位置上去。

function GameIsOver: integer ;

int GameIsOver(void) ;

declare function GameIsOver cdecl()

傳回底下任一整數：

遊戲尚未結束

棋盤上每一個位置都已擺滿六邊棋子

你的程式贏了

評分程式贏了

procedure MakeLibMove ;

void MakeLibMove(void) ;

declare sub MakeLibMove cdecl()

讓評分程式集(evaluation library)計算下一步該下的位置並將它的一個六邊棋子放在該位置上。棋盤盤面的改變可以用 LookAtBoard 及其他的函數來查看。

function GetRow : integer ;

int GetRow(void) ;

declare function GetRow cdecl()

傳回最近評分程式集(evaluation library)所下的六邊棋子的列數(row)，或傳回-1 若尚未下。此函數將一直傳回相同的行數值一直到你的程式呼叫 MakeLibMove 爲止。

function GetColumn : integer ;

int GetColumn (void) ;

declare function GetColumn cdecl()

傳回最近評分程式集(evaluation library)所下的六邊棋子的行數(column)，或傳回-1 若尚未下。此函數將一直傳回相同的行數值一直到你的程式呼叫 MakeLibMove 爲止。

function GetMax : integer ;

int GetMax(void) ;

declare function GetMaxcdecl()

傳回棋盤大小，N。

評分方式

假如你的程式下贏的話，則得滿分。假如你的程式下輸的話，則得滿分之 20%。假如你的程式在遊戲結束前結束或執行時間超過時間限制，則得 0 分。

2. 參考解答

```
/* 這個程式用的演算法是 Greedy Method，在五組測試後，所得分數爲 68 分，是 IOI'97 中本題最高的得分，總計是贏三盤輸兩盤，我是從 column 1 找起，找到所能連接到最大的 column 值，就選擇這一步走下去，這樣可以很快的從 column 1 找到 column N*/
#define NDEBUG
#ifndef NDEBUG
#include <conio.h>
#endif
#include <stdio.h>
#include <stdlib.h>
```

```
#include "hexc.h"
#define MAX 21
#define Disable (8)
#define Human 1
#define Comp 2
int max, mr[] = {1, 1, 0, 0, -1, -1},
   mc[] = {0, 1, -1, 1, -1, 0} ;
int canchoose[MAX][MAX] ;
char board[MAX][MAX] ;
/*這個函式在此程式中扮演了極重要的角色，是找出從 column 1 所能連接到的格子，也就是下一步的「候補格子」*/
void Updatecanchoose(void)
{
   int r, c, k ;
   for(r=1; r<=max; r++)
   if(board[r][1]!=Comp) canchoose[r][1] = 1 ;
for(r=2; r<=max; r++)
   for(c=1; c<=max; c++) {
    if(board[r][c] == Comp) continue ;
    for(k=0; k<6; k++)
   if(board[r+mr[k]][c+mc[k]]==Human &&
    canchoose[r+mr[k]][c+mc[k]]) {
    canchoose[r][c] = 1 ;
    break ;
                    }
            }
}
void InitBoard(void)
{
    int r, c ;
    max = GetMax() ;
    for(r=0; r<=max+1; r++)
board[r][0] = board[r][max+1] = Disable ;
    for(c=0; c<=max+1; c++)
board[0][c] = board[max+1][c] = Disable ;
    for(r=1; r<=max; r++)
     for(c=1; c<=max; c++)
      board[r][c] = LookAtBoard(r, c) ;
    Updatecanchoose() ;
}

void MaxcolChoose(int *prow, int *pcol)
{
    int r, c ;
    for(c=max; c>=1; c--)
          for(r=max; r>=1; r--)
if(canchoose[r][c] && board[r][c]==NULL) {
   *prow = r ;
   *pcol = c ;
   return ;
   }
}

void FindMove(int *prow, int *pcol)
{
// 找尋 column 最大值
      MaxcolChoose(prow, pcol) ;
      board[*prow][*pcol] = Human ;
      Updatecanchoose() ;
```

```
}
#ifndef NDEBUG
void display(void)
{
      int row, col, max, l;
      max = GetMax ();
for (row=1; row<=max; row++)
{
   for (col=1; col<row; col++)
      cprintf (" ");
   for (col=1; col<=max; col++)
   { l = LookAtBoard (max+1-row, col);
      textcolor (l == 0 ? 15 : l*3-2);
      cprintf ("%d ",l);
   }
   textcolor (WHITE);
   cprintf ("\r\n");
      }
}
#endif
int main()
{
      int gameover, row, col ;
// 初始化
      InitBoard() ;
#ifndef NDEBUG
      display() ;
#endif
   do {
   // find a good move
      FindMove(&row, &col) ;
      PutHex(row, col) ;
      MakeLibMove() ;
      board[GetRow()][GetColumn()] = Comp ;
#ifndef NDEBUG
            display() ;
            getch() ;
#endif
            gameover = GameIsOver() ;
      } while(gameover == 0) ;
      return 0 ;
}
```

3. Hex 之得分分布

此題是第一天之三題中較難的一題，從表三 Hex 之得分分布表可知，約有一半(51.58%)的考生得分在 20 分以下，得分在 20 分以上者已寥寥無幾，得分在 61～70 之間者僅有 2.71%。

表三　參賽者在 Hex 之得分分布

人(次)數	累積人(次)數	百分比	累積百分比
69	69	31.22	31.22
114	183	51.58	82.81
3	186	1.36	84.16
20	206	9.05	93.21
1	207	0.45	93.67
8	215	3.62	97.29
6	221	2.71	100.00

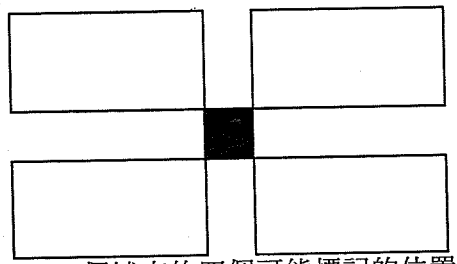
三、DAY2 試題

4. 第二天第一題(Maps:Map Labelling)

你是地籍測量員助理，你的工作是必

須將城市的名稱寫到新的地圖上。

每一張地圖是用 1000*1000 格的矩形代表。每一個城市在地圖上佔用一格的空間，而城市名稱必須放在用格子所組成的長方形內，我們稱此長方形爲標記(Labels)。



圖一：一個城市的四個可能標記的位置

標記擺放的位置必須符合下列條件：

1. 一個城市的標記必須出現在四個可能的位置上的其中一個，如圖一所示。
2. 標記不能相互重疊。
3. 標記不能覆蓋過任何城市
4. 標記必須至於整張地圖內。

每一個標記包含城市名稱的每一個字母加上一個空白；每一個城市名稱的字母的寬度(width)及高度(height)將被輸入，而空白的大小與這些字母一致。

地圖的原點格(0,0)是在左下角，在圖二的範例中，Langa 城是在(0,3)，Ceres 城在(6,1)，而 Paarl 城在(7,3)的位置上，所有的標記的擺設都是合法的，但這不是唯一合法的擺設方式。

Task：

你的程式必須讀入每一個城市位置，城市名稱字母的寬高，及城市名稱，然後在不違反上述標記擺設條件下，儘可能將越多的城市標記(label)擺在地圖上。然後將這些城市標記位置輸出。

Input：

輸入檔第一行是城市數 N。之後每一行記錄著一個城市的資訊，包括：

城市位置：(水平位置 X，垂直位置 Y)、城市名稱字母大小：字母寬度 W，字母高度 H、城市名稱。每個城市名稱都只有一個字(Word)，且最多有 1000 個城市，城

4		L	a	n	g	A	○								
3	●							●							
2									P	a	a	r	l	○	
1							●								
0	○	C	e	r	e	S									
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

圖二：

● 代表城市的位置　　○代表一個空白

市名稱最長不超過 200 個字母(letters)。

Sample Input：

MAPS.DAT	Explanation
3 0 3 1 1 Langa 6 1 1 1 Ceres 7 3 1 2 Paarl	N=3 X=0,Y=3,W=1,H=1

Output：

你的程式須輸出 N 行，每一行有一個城市標記的左上角的水平及垂直方位。水平及垂直方位中間用一個空白分開。假若有城市無法被合法的擺設到地圖上，則輸出**-1 -1**。所有輸出行的順序必須跟輸入檔內城市的順序一致。

Sample Output:

MAPS.OUT	Explanation
1 4	Langa's label is at (1,4)
0 0	Ceres's label is at (0,0)
8 2	Paarl's label is at (8,2)

評分方式

每一組測試資料：分數是你所有合法擺設到地圖上的城市標記數除以最佳數之百分比，最低 0%，最高 100%。如果你有任何標記違反了任一擺設限制條件，則得 0 分。如果標記與城市位置不符合，則得 0 分。

5. 參考解答

```
#include<fstream.h>
#include<assert.h>
#include<string.h>
#include<stdlib.h>
////////////////////////////////
// maps.cpp                   //
// wrote on 12/04/97 in Cape Town, SA //
// comments added on 2/07/98  //
//   by Lu-chuan Kung         //
// 說明：                     //
//   本城市的目的在輸入資料中給的許//
//多城市中，選出最多個不相重疊的城//
//市來標記。本程式所採用的演算法爲//
//「貪心演算法」，先依照城市標記由//
//小排到大，然後一個一個城市試試看//
//，若放在地圖上不會和之前所選的城//
//市或標記重疊，則選擇之。(此解法//
//並不能解出最佳解，但可以求出不錯//
//的解)                      //
////////////////////////////////
// 常數定義
#define MAX_N (1000) //城市數目的最大值
#define MAX_LEN (201) //城市名稱的最大長度
#define MAX_CORD (1000) //坐標的最大值
////////////////////////////////
// 定義型態
typedef int (*FCMP)
   (const void *,const void *);

struct POINT //定義「點」型態爲兩個整數
{
   int x,y; // x 爲 x 座標 , y 爲 y 座標
};

struct SortNode  //定義用來排序的資料結構
{
      int index;  // 城市的編號
```

```
    long area; // 面積
};

struct OutNode //定義用在輸出時排序的資料結構
{
  int index; //城市的編號
  int x,y;    // 座標
};
////////////////////////////////
// 全域變數
char huge map[MAX_CORD][MAX_CORD/8];
// 用 bitmap 來確定座標點是否用過
int nCity;          // 輸入資料的城市個數
SortNode citys[MAX_N];
 // 用來依照面積大小排序的陣列
OutNode outCity[MAX_N];
 // 用來依照名稱排序的陣列
int nameLen[MAX_N];
 // 城市名稱的長度
int xOfCity[MAX_N];
 // 城市標記的 x 座標
int yOfCity[MAX_N];
 // 城市標記的 y 座標
int wOfCity[MAX_N];
 // 城市標記的寬度
int hOfCity[MAX_N];
 // 城市標記的高度

// read 函式傳回座標(x,y)是否已經被占據
inline int Read(int x,int y)
{
    return map[x][y/8] & (1<<(y%8)) ;
}
// write 函式將 座標 (x,y) 設定爲已占據
inline void Write(int x,int y)
{
    map[x][y/8] = (1<<(y%8)) ;
}
// 依面積排序的函式（給 quicksort 呼叫）
int SortFunc
 (const SortNode *pa,const SortNode *pb)
{
    long x = pa->area - pb->area;
    if( x < 0L )
        return -1;
    else if( x > 0L)
        return 1;
    else
        return 0;
}
//依編號排序的函式（恢復輸入時的順序）
int SortByIndex
 (const OutNode *pa,const OutNode *pb)
{
    return pa->index - pb->index;
}
// 由輸入檔讀入城市資料
void ReadInput(void)
{
    ifstream input("maps.dat");
```

```
    assert(input);
    char str[MAX_LEN];
    int i,w,h;
    input >> nCity;    // 讀入城市個數
    for(i=0;i<nCity;i++) {
        citys[i].index = i;
        input>>xOfCity[i] >> yOfCity[i];
        Write( xOfCity[i], yOfCity[i] );
          // 設定此座標爲已被占據
        input >> w >> h ;
        wOfCity[i] = w;
        hOfCity[i] = h;
        input >> str ;
        nameLen[i] = strlen(str)+1;
// 面積爲
  w（寬度）×h（高度）×length（字數）
citys[i].area = (long)w * h * (nameLen[i]) ;
    }
    // 依面積排序
    qsort(citys,nCity,sizeof(SortNode),
     (FCMP)SortFunc);
}
// 此函測試標記的 4 個方向是否可放，
// 若可以，則傳回標記的 maxX 與 maxy
// 若失敗，則傳回 -1
int CanPutAndPutDir(int cityno,
  int &minx,int &maxy)
{
    POINT startPoint[4];
    POINT endPoint[4];
    int i,x,y,sucess;
    // top left 左上角
    startPoint[0].x = xOfCity[cityno] -
     wOfCity[cityno] * nameLen[cityno];
    startPoint[0].y = yOfCity[cityno] + 1;
    endPoint[0].x
     = xOfCity[cityno] - 1;
    endPoint[0].y    = yOfCity[cityno] +
hOfCity[cityno];
    // bottom left 左下角
    startPoint[1].x = xOfCity[cityno] -
     wOfCity[cityno] * nameLen[cityno];
    startPoint[1].y = yOfCity[cityno] -
     hOfCity[cityno];
    endPoint[1].x   = xOfCity[cityno] - 1;
    endPoint[1].y   = yOfCity[cityno] - 1;
    // top right 右上角
    startPoint[2].x = xOfCity[cityno] + 1;
    startPoint[2].y = yOfCity[cityno] + 1;
    endPoint[2].x    = xOfCity[cityno] +
     wOfCity[cityno] * nameLen[cityno];
    endPoint[2].y    = yOfCity[cityno] +
     hOfCity[cityno];
    // bottom right 右下角
    startPoint[3].x = xOfCity[cityno] + 1;
    startPoint[3].y = yOfCity[cityno] -
     hOfCity[cityno];
    endPoint[3].x    = xOfCity[cityno] +
     wOfCity[cityno] * nameLen[cityno];
    endPoint[3].y   = yOfCity[cityno] - 1;
```

```
    // 測試四個方向
    for(i=0;i<4;i++) {
// 若超過邊界，則試下一種方向
if( startPoint[i].x < 0
  || startPoint[i].x >= MAX_N ||
  startPoint[i].y < 0
  || startPoint[i].y >= MAX_N )
  continue;
 if( endPoint[i].x < 0
  || endPoint[i].x >= MAX_N ||
  endPoint[i].y < 0
  || endPoint[i].y >= MAX_N )
  continue;
 sucess = 1;
 // 測試在標記內的每一個坐標是否已被占據
 for( x=startPoint[i].x ;
 x <=endPoint[i].x ;x++ ) {
   for(y=startPoint[i].y ;
    y<=endPoint[i].y ; y++) {
if( Read(x,y) ) {// 若已被占據，
 // 則離開 for 迴圈
    sucess=0;
      goto ExitFor;
                    }
                }
            }
ExitFor:
    if( sucess ) {
     // mark on the map 在地圖上
     // 標記的範圍內登記爲被占據
    for( x=startPoint[i].x ;
     x <=endPoint[i].x ;x++ ) {
    for(y=startPoint[i].y ;
     y<=endPoint[i].y ; y++) {
      Write(x,y);
                    }
                }
                // 傳回標記的左上角座標
                minx = startPoint[i].x;
                maxy = endPoint[i].y;
                return i;
        }
    }
    return -1;
}
// 放置地圖
void PutLabel(void)
{
     int i,j;
     int z;
     int x,y;
     // 初始化 outCity 陣列
     for(i=0;i<nCity;i++) {
          outCity[i].x = -1;
          outCity[i].y = -1;
     }
     // 依面積由小到大的順序 try try 看
     for(i=0;i<nCity;i++) {
    j = citys[i].index ;
    outCity[i].index = j;
    z = CanPutAndPutDir(j,x,y);
    if( z >= 0 ) {        // 可以放得進去
      outCity[i].x = x;
      outCity[i].y = y;
```

```
            }
        }
        // 依照輸入時的順序排序
        qsort( outCity,nCity,sizeof(OutNode),
          (FCMP)SortByIndex);
}
// 輸出答案
void Output(void)
{
        ofstream output("maps.out");
        int i;
        for(i=0;i<nCity;i++) {
              output << outCity[i].
               x << " " << outCity[i].y << endl;
        }
        return;
}
// 主函式
int main(void)
{
        ReadInput();
        PutLabel();
        Output();
        return 0;
}
```

6. Maps 之得分分布

第二天的第一題也相當難，得分在 0～10 者佔絕大多數(78.73%)。但也有約 9.05%的考生得分在 91 分以。

表四　參賽者在 Maps 之得分分布

人(次)數	累積人(次)數	百分比	累積百分比
174	174	78.73	78.73
10	184	4.52	83.26
1	185	0.45	83.71
3	188	1.36	85.07
1	189	0.45	85.52
4	193	1.81	87.33
3	196	1.36	88.69
5	201	2.26	90.95
20	221	9.05	100.00