1998 年資訊奧林匹亞試題及參考解答(一)

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、第一天第一題：接觸

Astro Insky 博士在某無線遙測天文台工作，最近她注意到一種由銀河中心發射出來的微波脈衝，十分令人好奇。這脈衝可能是由外太空智慧生物所發出的？或只是星球的尋常脈動而已？

任務 (Task)

你需要提供一個工具去分析她檔案中的位元字串，以幫助 Insky 博士發現真相。Insky 博士希望能找到長度介於(包含)A 到 B 間的字串，這些字串較常出現於每日的檔案之中。對每一個檔案而言，她希望找到較大的 N 種不同頻率（即出現的次數）。不同字串可以互相重疊，且只考慮至少出現過一次的字串。

輸入資料 (Input Data)

檔案 CONTACT.IN 存放資料序列，格式如下：

第 1 橫列：整數 A 表示最短字串長度。

第 2 橫列：整數 B 表示最長字串長度。

第 3 橫列：整數 N 表示有幾種不同頻率。

第 4 橫列：一串 0 與 1 字元，以字元 2 結束。

輸入範例 (Sample Input)：

```
2
4
10
01010010010001000111110110000101
00110011110000100100111110010000
0002
```

在此例中，需要找出下列字元序列中最常出現的 10 種字串頻率，其長度介於 2 到 4 之間。

```
01010010010001000111110110000101
00110011110000100100111110010000
000
```

注意此處輸入檔案的第四橫列在印出時是因排版空間不夠而斷開。在此例中，字串 100 出現了 12 次，字串 1000 出現了 5 次，而 00 是最常出現的字串。

輸出資料(Output Data)

輸出檔 CONTACT.OUT 中最多可有 N 個橫列，分別列出 N 個最常出現的頻率以及相對應的各個字串。列印時應以頻率的遞減順序印出，格式如下：

frequency pattern pattern ... pattern

此處frequency爲其後各字串出現的頻率。各橫列中的不同字串應以其長度的遞減順序排列。同長度的各字串再依遞減的數值順序排列。如果出現的頻率數小於N種，則輸出橫列數小於N。

輸出範例 (Sample Output):

對應輸入範例，輸出應如下：

```
23 00
15 10 01
12 100
11 001 000 11
10 010
8 0100
7 1001 0010
6 0000 111
5 1000 110 011
4 1100 0011 0001
```

條件限制(Constraints)

輸入檔案中最多可含 2 Megabytes。

參數 A、B 及 N 限制如下：

$0 < N \le 20$

$0 < A \le B \le 12$

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)

/*這題一看到題目的要求：輸入檔案最大達 2Megabytes，就知道必須找出一個時間複雜度低於或等於 O(n)的演算法，而且係數還不能太大。不過由於要求的是統計的工作，所以 O(n)是跑不掉的，因此要想辦法降低其係數。因爲資料的內容爲 1 或 0，故可以下列方法達成：

(1) 將一字串視爲二進位數字，加上長度資訊便可表示這字串，例如：01000 可視爲 8，長度 5，再開一二維陣列存每一種字串的頻率，例如：01000 的頻率存在 times[5][8]中。

(2) 在讀資料時，採每次讀一字元的方式，並以 table[n]表示以目前讀到的字元爲結尾且長度爲 n 的字串，每讀入一字元時做 table[n] =(table[n]%(2^n))*2+k 運算即可轉爲下一筆長度爲 n 的字串(table[n]爲目前長度 n 的字串，2^n 指 2 的 n 次方，k 是下一字元的值。) 例如：在資料 0001011001000...中讀到第六字元則 table[4]應等於 5，即字串 0101 讀下一字元時只須做(4%8)*2+1 得 tabl[4]=11，即第四筆長度爲四的字串爲 1011，接著 times[4][11]++累加頻率。

以此 DP 的方式減少係數後可在約 17 秒左右解出 2Megabytes 大小的測試值，程式碼如下（註：在 TC 下若全域變數使用超

過 64K 的記憶體，會導致變數存取速度減慢，而不能在 20 秒內解出。）:*/

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

FILE *fp, *fi;
int a, b, n;
int table[22];
    //記錄目前各種長度的字串內容
long *times[13];
    //儲存各種字串的頻率
long radix[22];
long map[22];
    //記錄前 n 大的頻率值

void GetData( void )

    //開檔及讀 N,A,B
{
      fp = fopen( "contact.in", "r+t");
      fi = fopen( "contact.out", "w+t");
      fscanf( fp, "%d%d%d\n", &a, &b, &n);
      radix[0] = 1;
      for( int i = 1; i < 20; i++)
            radix[i] = radix[i-1] * 2;
      for( i = 0; i <= 12; i++)
      {
            times[i] = new long[radix[i]];
            for( int j = 0; j < radix[i]; j++)
                  times[i][j] = 0;
      }
}

void Insert( long x)
{
      for( int i = 0; i < n; i++)
            if( map[i] < x)
                  map[i] ^= x ^= map[i] ^= x;
            else if( map[i] == x)
                  return;
}

void Count( void )

    //計算每種字串出現頻率
{
      int k;
      long t = 0;
      char ch;
      while( 1 )
      {
            fscanf( fp, "%c", &ch);
            if( ch == '2')
                  break;
            if( ch != '0' && ch != '1')
                  continue;
            k = ch - '0';
            for( int i = a; i <= b; i++)
            {
                  table[i] = table[i] * 2 + k;
                  if( t >= i)
                        table[i] %= radix[i];
                  if( t >= i - 1)
                        times[i][table[i]]++;
            }
            t++;
      }
      for( int i = a; i <= b; i++)
            for( int j = 0; j < radix[i]; j++)
                  Insert( times[i][j]);
}

void Print( int len, int x)
{
      fprintf( fi, " ");
      for( int i = len - 1; i >= 0; i--)
      {
            fprintf( fi, "%d", x / radix[i]);
            x %= radix[i];
      }
}
void Show( void )
{
      int i = 0, j ,k;
      fprintf( fi, "%ld", map[i]);
      for( j = b; j >= a; j--)
            for( k = radix[j] - 1; k >= 0; k--)
                  if( times[j][k] == map[i])
                        Print( j, k);
      for( i = 1; i < n; i++)
```

```
    {
        if( map[i] == 0)
            break;
        fprintf( fi, "\n%ld", map[i]);
        for( j = b; j >= a; j--)
        for( k = radix[j] - 1; k >= 0; k--)
            if( times[j][k] == map[i])
                Print( j, k);
    }
    for( i = 0; i < 12; i++)
        delete times[i];
}
int main( void )
{
    GetData();
    Count();
    Show();
    fclose( fp );
    fclose( fi );
    return 0;
}
```

三、第一天第二題：星光滿夜

在夜晚的高空上，充滿了閃亮的星群(cluster)。所謂星群是指一群在上、下、左、右或斜對角方向相互連接的星星。任一星群不得爲其他更大星群的子星群。

星群可以是相似的。兩個星群如果有相同的形狀，縱使方向不同，即稱之爲相似的(similar)星群。大致來說，每一個星群有 8 個可能的轉變方向，如圖 1：

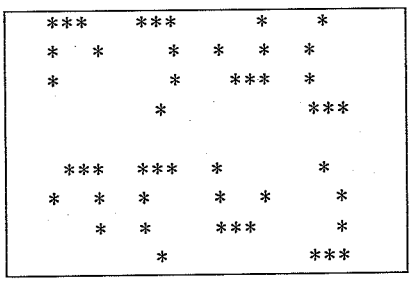

圖1. 8個相似的星群

此夜晚的天空圖(sky map)是以一個 2 維陣列表示。陣列中每一個格子(cell)代表一個星星。（以 1 代表星星，0 代表空白。）

Task（任務）

給一個天空圖，將所有星群用小寫字母標示出來。請將相似的星群用同一個小寫字母表示，不相似的星群則以不同的字母表示。請將天空圖中所有的星星(1)改成它的星群所對應的小寫字母。

Input Data（輸入資料）

輸入檔 STARRY.IN 的前二橫列爲 W（天空圖的寬度）及 H（天空圖的長度）。天空圖自第 3 橫列開始，共有 H 橫列，每列 W 個字元（0 或 1）。

輸入範例: (Sample Input)

```
23
15
10001000000000010000000
01111100011111000101101
01000000010001000111111
00000000010101000101111
00000111010001000000000
00001001011111000000000
10000001000000000000000
00101000000111110010000
00001000000100010011111
00000001110101010100010
00000100110100010000000
00010001110111110000000
00100001110000000100000
00001000100001000100101
00000001110001000111000
```

在此範例中，此天空圖寬度爲 23 且長度爲 15。爲使此圖更容易了解，圖 2 只顯

示星星的位置。

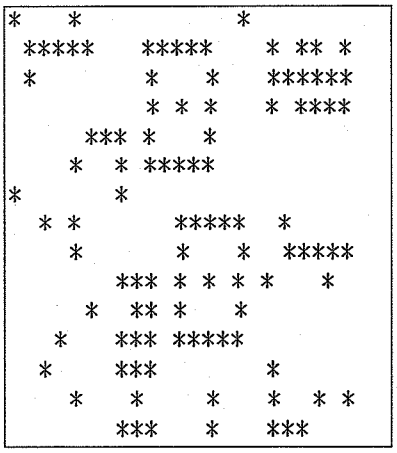

圖 2. 天空圖範例

Output Data（輸出資料）

輸出檔爲 STARRY.OUT，應含有與輸入檔 STARRY.IN 相同的天空圖，但每個星群內的星星已按照任務說明的方式改爲小寫字母。底下爲一組解答，而圖 3 是將 0 去掉後的示意圖。

輸出範例: (Sample Output)

```
a000a0000000000b0000000
0aaaaa000ccccc000d0dd0d
0a0000000c000c000dddddd
000000000c0b0c000d0dddd
00000eee0c000c000000000
0000e00e0ccccc000000000
b000000e000000000000000
00b0f000000ccccc00a0000
0000f000000c000c00aaaaa
0000000ddd0c0b0c0a000a0
00000b00dd0c000c0000000
000g000ddd0ccccc0000000
00g0000ddd0000000e00000
0000b000d0000f000e00e0b
0000000ddd000f000eee000
```

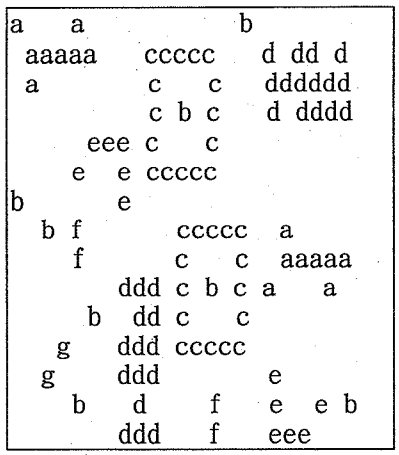

圖 3. 星群已標示完成的天空圖

Constraints（條件限制）

0 ≤ W (天空圖的寬度) ≤ 100

0 ≤ H (天空圖的長度) ≤ 100

0 ≤ 星群數≤ 500

0 ≤ 不相似的星群數 ≤ 26 (a..z)

1 ≤ 每一星群的星星數 ≤ 160

四、參考解答

試作者：江盈宏（高雄市立高雄中學三年級學生，IOI'98 銅牌獎）

```
#include<stdio.h>
/*
整體變數
map 代表讀入的資料(0 或 1),2 代表已經處理過
result 爲處理完的結果
compare 是比對時使用的二維陣列
w 爲寬,h 爲高
*/
char map[100][100];
char result[100][100];
char compare[100][100];
int w,h;

/*Cluster 結構與陣列*/
struct Clu {
```

```
  int stars;                /*星數*/
  int  w,h,l,u;             /*此星群之: w(寬),
h(高), l(最左), u(最上)*/
  int pos[160][2];      /*星星之座標*/
};
struct Clu clu[26],now;
int clus;

/*由檔案讀入資料*/
void getdata()
{
  int i,j;
  FILE *inf;
  inf=fopen("starry.in","r");

  fscanf(inf,"%d",&w);
  fscanf(inf,"%d",&h);
  for(i=0;i<h;i++)
    fscanf(inf,"%s",map[i]);
  for(i=0;i<h;i++)
    for(j=0;j<w;j++)
      result[i][j]='0';

  fclose(inf);
}

/*判斷是否爲合法的位置*/
int Position(int x,int y)
{
  return (x>=0 && x<h && y>=0 && y<w);
}

/*逐一找出 connected component 之函式,使
用 BFS*/
void FindComponent(int x,int y)
{
  int queue[160][2],front=0,rear=0;
  int i;
  int nx,ny,zx,zy;
  int l,r,u,d;
  int   move[8][2]={{-1,-1},{-1,0},{-1,1},{0,-
1},{0,1},{1,-1},{1,0},{1,1}};

  queue[front][0]=x,queue[front][1]=y,front++
;
  map[x][y]='2';
  now.stars=0;
  while(front!=rear) {
    nx=queue[rear][0];
    ny=queue[rear][1];
    now.pos[now.stars][0]=nx,now.pos[now.s
tars][1]=ny;
    now.stars++;
    for(i=0;i<8;i++) {
      zx=nx+move[i][0],zy=ny+move[i][1];
      if(Position(zx,zy))
        if(map[zx][zy]=='1')
          queue[front][0]=zx,queue[front][
1]=zy,map[zx][zy]='2',front++;
    }
    rear++;
  }

  /*找出星群之最左及最上,並將座標減去
最左及最上,成相對位置*/
  u=d=now.pos[0][0];
  l=r=now.pos[0][1];
  for(i=1;i<now.stars;i++) {
    if(now.pos[i][0]<u) u=now.pos[i][0];
    if(now.pos[i][0]>d) d=now.pos[i][0];
    if(now.pos[i][1]<l) l=now.pos[i][1];
    if(now.pos[i][1]>r) r=now.pos[i][1];
  }
  now.h=d-u;
  now.w=r-l;
  now.l=l,now.u=u;

  for(i=0;i<now.stars;i++) {
    now.pos[i][0]-=now.u;
    now.pos[i][1]-=now.l;
  }
}

/*判斷星群 n 與星群 now 之異同*/
int Check(int n,int usen,int con)
{
```

```
   int i,j;
   int use[2][2]={ {0,1},{1,0} };
   int co[4][4]={ {0,-1,0,-1},{0,-1,1,1},{1,1,0,-
1},{1,1,1,1} };
   for(i=0;i<clu[n].stars;i++)
      if(compare[co[con][0]*clu[n].h-
co[con][1]*now.pos[i][use[usen][0]]]
                    [co[con][2]*clu[n].w-
co[con][3]*now.pos[i][use[usen][1] ]]==1)
         compare[co[con][0]*clu[n].h-
co[con][1]*now.pos[i][use[usen][0]]]
                    [co[con][2]*clu[n].w-
co[con][3]*now.pos[i][use[usen][1] ]]=0;
      else
         return 0;
   return 1;
}

/*將已有的星群逐一比對,並做 8 次旋轉*/
int TheSame(int n)
{
   int i,j,k;
   for(i=0;i<2;i++)
      for(j=0;j<4;j++) {
         for(k=0;k<clu[n].stars;k++)
            compare[clu[n].pos[k][0]][clu[n].po
s[k][1]]=1;
         if(Check(n,i,j)) return 1;
      }

   for(i=0;i<clu[n].stars;i++)
      compare[clu[n].pos[i][0]][clu[n].pos[i][1]]
=0;

   return 0;
}

/*傳回 now 所代表的星群,若傳回 clus 代表
爲新增之星群*/
int Compare()
{
   int i;
   for(i=0;i<clus;i++)
      if(now.stars==clu[i].stars)
         if(now.w==clu[i].w && now.h ==clu[i].
h || now.w==clu[i].h &&
now.h==clu[i].w)
            if(TheSame(i)) return i;
   return clus;
}

/*主要解題函式,對每一個座標點尋找未發
現過的星群並輸出至 result*/
void solve()
{
   int i,j,k;
   int n;
   for(i=0;i<h;i++)
      for(j=0;j<w;j++)
         if(map[i][j]=='1') {
            FindComponent(i,j);
            n=Compare();
            if(n==clus) clu[clus++]=now;
            for(k=0;k<now.stars;k++)
               result[now.pos[k][0]+now.u][now.
pos[k][1]+now.l]='a'+n;
         }
}

/*輸出至檔案*/
void output()
{
   int i,j;
   FILE *outf;
   outf=fopen("starry.out","w");
   for(i=0;i<h;i++) {
      for(j=0;j<w;j++)
         fprintf(outf,"%c",result[i][j]);
      fprintf(outf,"\n");
   }
   fclose(outf);
}

/*主函式*/
int main()
{
   getdata();
   solve();
   output();
   return 0;
}
```