LA 4629 Knowledge for the masses

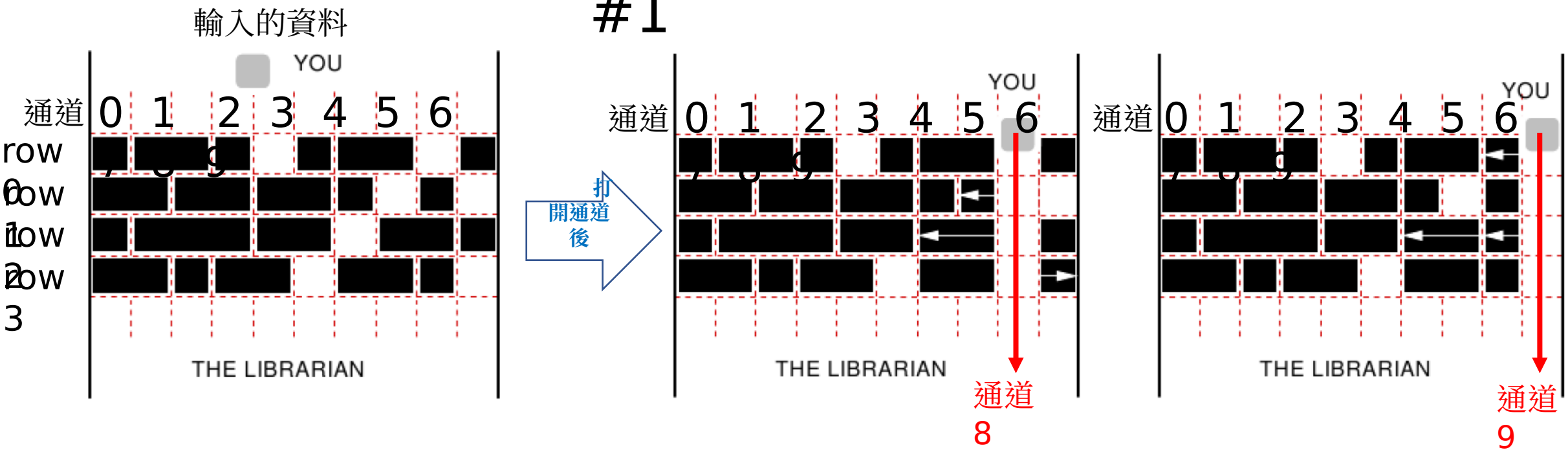
(Central Europe RC 2009, UVa 1444)

LA 4629 Knowledge for the masses (Time Limit: 5 seconds)

圖書館借書：到圖書館借書，當拿到書後要找管理員登記，借書者與管理員間有 n 個 row 軌道 (每個 row 軌道長度皆相同), 軌道上有一些書櫃可以左右移動，借書者必須左右移動書櫃形成通道才能走到後面 (經過 n 個 row 軌道) 找管理員登記，假設每移動一個書櫃成本 (或代價) 為 1 (不管移動距離多長), 請問借書者至少要移動多少書櫃 (成本或代價) 才能找到管理員，並輸出借書者可以走的通道編號，如果不只一條通道，請依通道編號由小至大輸出。(輸入的 n 個 row 軌道至少可以移出一個通道讓借書者通過)

Example:

Test Case #1



書櫃搬動的最少數目 (成本): 3

打開的通道編號 (由小至大): 8, 9

Sample Input

```
1
4 10
8 1 2 1 0 1 2 0 1
7 2 2 2 1 0 1 0
6 1 3 2 0 2 1
```

1 Test Case 個數

4 10 row,col 數目

8 1 2 1 0 1 2 0 1 該 row 段數接著各段的長度

(0: 代表寬度 1 沒擺書櫃處；非 0: 代表書櫃的寬度)

Test Case #1

輸入的資料

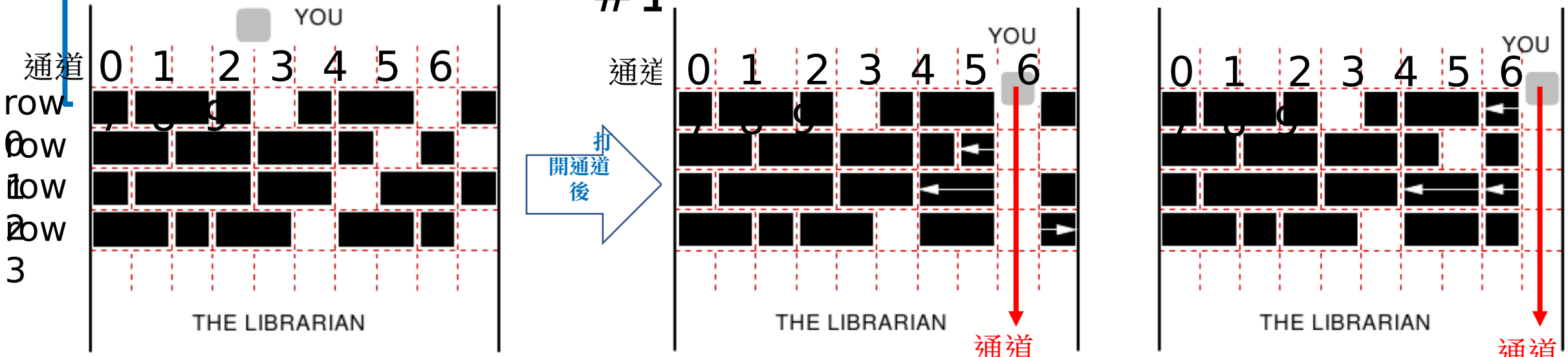
Sample Output

```
3
8 9
```

3 書櫃搬動的最少數目（成本）

8 9 打開的通道編號（由小至大）

Test Case #1

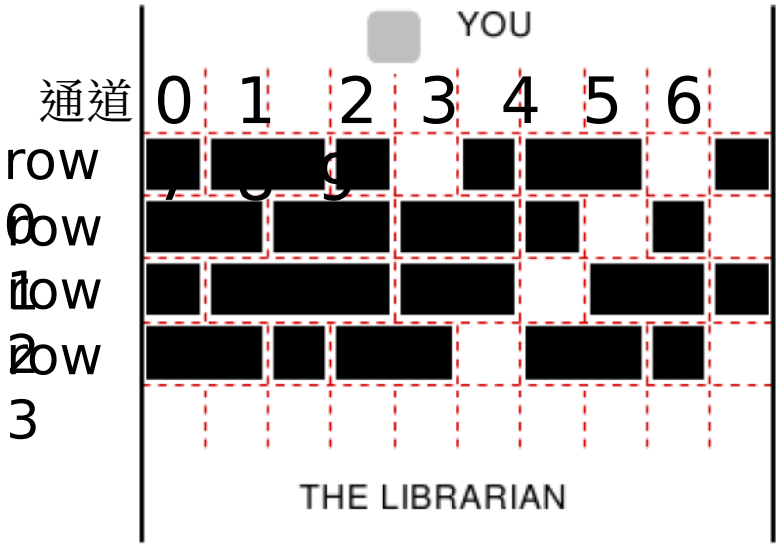


Solution

- Greedy Method
  - 針對每個軌道逐一計算每段書櫃可以左移與右移的距離
  - 並計算能打通的通道其代價為何
  - 最後綜合結果找出需最少代價的通道使得借書者可以找到管理員

想法

想法



左移代價 [u]: 藉左移書櫃通道 u 打通需要最小的代價 (書櫃移動數)

右移代價 [u]: 藉右移書櫃通道 u 打通需要最小的代價 (書櫃移動數)

g[u]: 通道 u 打通累積需要最小的代價 (書櫃移動數)

can[u]: 通道 u 打通的軌道數 (row 數目)

row 0

通道	0	1	2	3	4	5	6
左移代價	X	X	9	X	X	0	1	X
右移代價	3	2	1	X	1	0	2	1
綜合結果	X	X	X	X	1	0	1	1
g[u]	3	2	1	X	1	0	1	1
can[u]	1	1	1	0	1	1	1	1

← min(左移代價 , 右移代價)

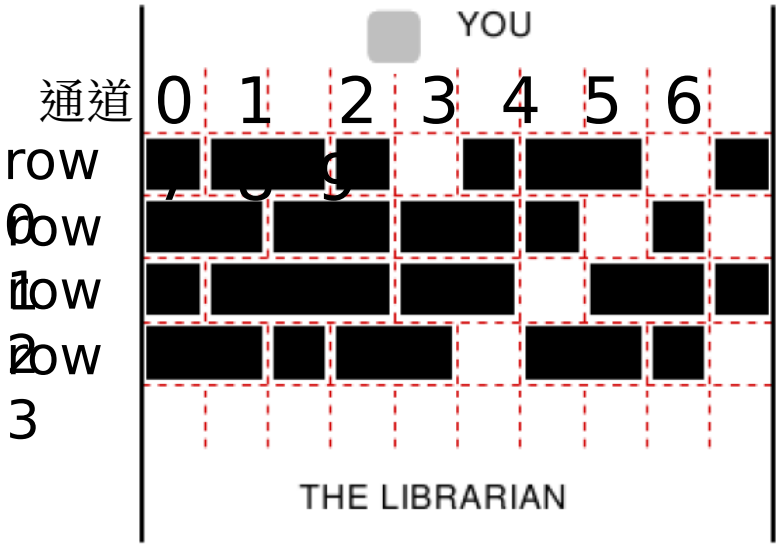
row 1

通道	0	1	2	3	4	5	6
左移代價	X	X	9	X	X	X	X	X
右移代價	4	3	2	4	4	3	3	1
綜合結果	4	3	2	4	4	3	3	1
g[u]	7	6	4	4	5	3	4	2
can[u]	2	2	1	1	2	2	2	2

← min(左移代價 , 右移代價)

X: 代表無窮大值

想法



左移代價 [u]: 藉左移書櫃通道 u 打通需要最小的代價 (書櫃移動數)

右移代價 [u]: 藉右移書櫃通道 u 打通需要最小的代價 (書櫃移動數)

g[u]: 通道 u 打通累積需要最小的代價 (書櫃移動數)

can[u]: 通道 u 打通的軌道數 (row 數目)

row 2

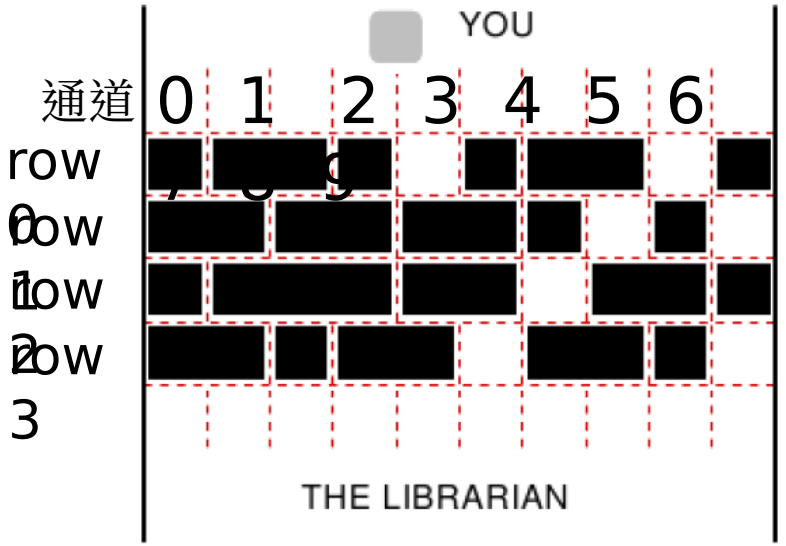
通道	0	1	2	3	4	5	6	
左移代價	X	X	9	X	X	X	0	
右移代價	X	X	2	X	X	1	0	
綜合結果	X	X	X	X	X	1	0	← min(左移代價 , 右移代價)
g[u]	10	10	9	4	5	4	4	2	2
can[u]	2	3	3	1	2	3	2	3

row 3

通道	0	1	2	3	4	5	6	
左移代價	X	X	9	X	X	X	0	X
右移代價	3	3	0	4	3	3	0	2
綜合結果	X	X	0	4	3	3	3	2	← min(左移代價 , 右移代價)
g[u]	13	14	0	8	8	7	4	4
can[u]	4	4	2	3	4	3	4

X: 代表無窮大值

想法



左移代價 [u]: 藉左移書櫃通道 u 打通需要最小的代價 (書櫃移動數)

右移代價 [u]: 藉右移書櫃通道 u 打通需要最小的代價 (書櫃移動數)

g[u]: 通道 u 打通累積需要最小的代價 (書櫃移動數)

can[u]: 通道 u 打通的軌道數 (row 數目)

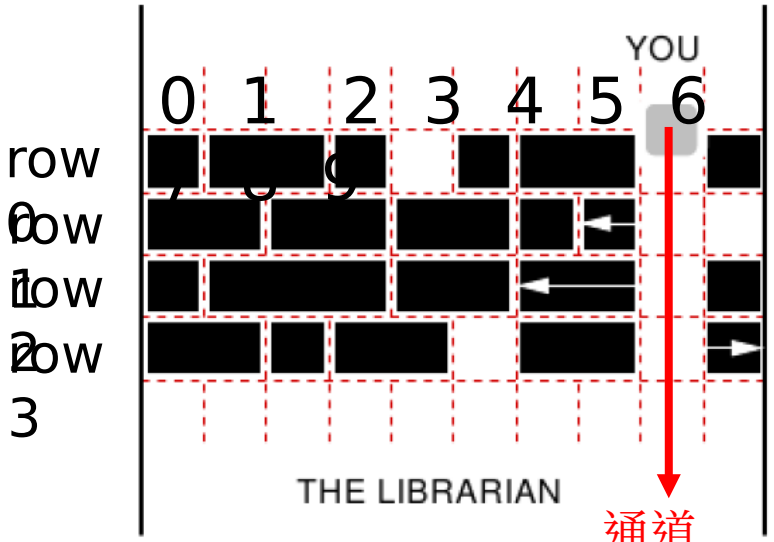
最後

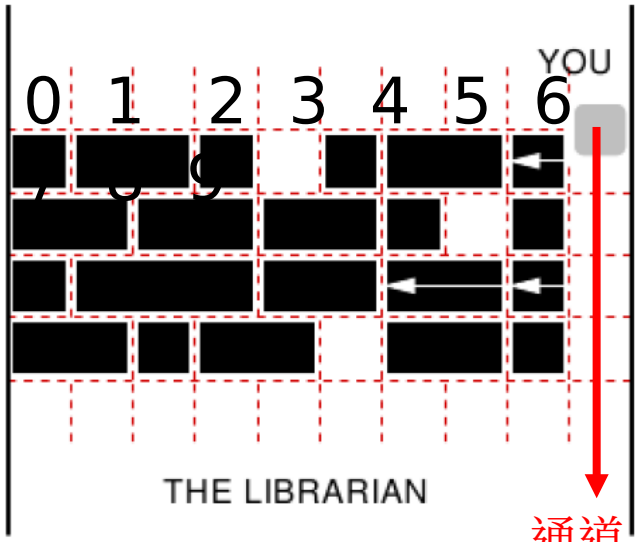
通道	0	1	2	3	4	5	6
g[u]	7	8	9	8	8	7	4	4
can[u]	4	4	3	3	3	3	4	

從 can[u]=4 中找 g[u] 最小值 , 將它們輸出

最小值 : 3

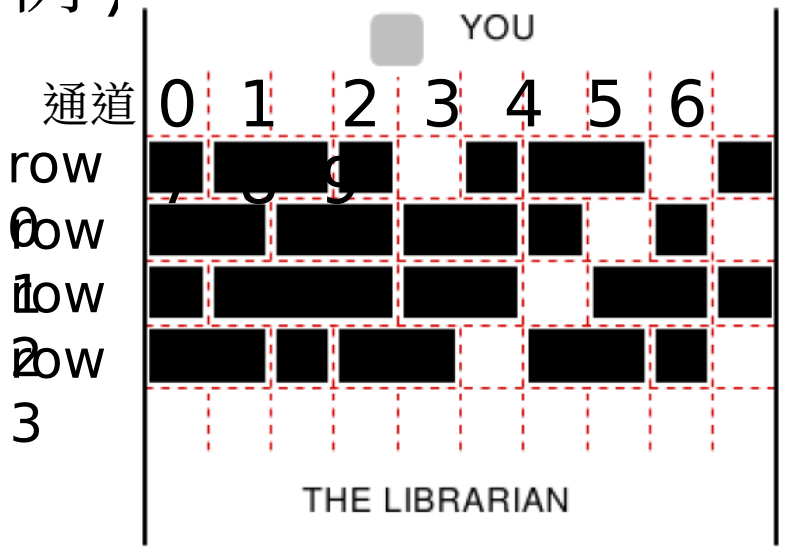
通道 (u): 8, 9





如何計算左移代價

如何計算左移代價 (以 row 0 為例)



輸入的資料

i	0	1	2	3	4	5	6	7
blo[i]	1	2	1	0	1	2	0	1

	0	1	2	3	4
zero[i]					

左邊 0 的個數 zn: 0

0 的所在的位置 j

now now

通道	0	1	2	3	4	5	6	7	8	9
	blo[0]	blo[1]		blo[2]	blo[3]	blo[4]	blo[5]		blo[6]	blo[7]

blo[0] = 1 () → now+=blo[0]　k=min(zn,blo[0])=0　現在 blo[0] 書櫃無法左移

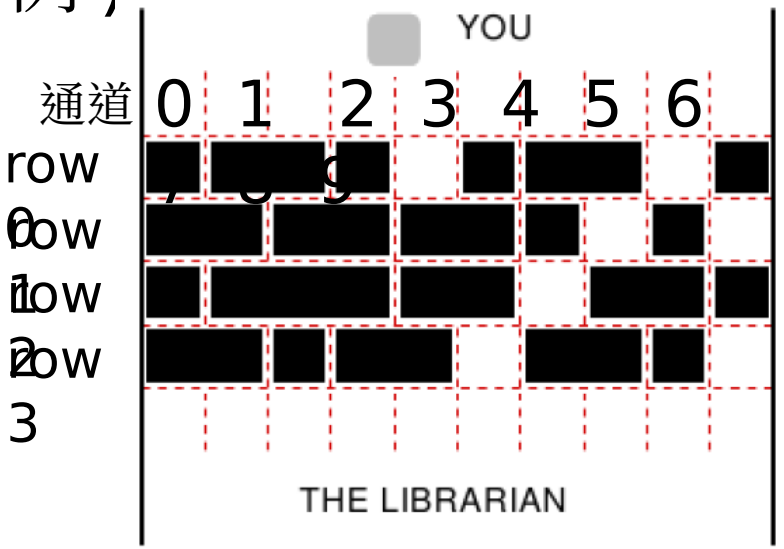
通道 u	0	1	2	3	4	5	6	7	8	9
value[u]	=1	-1	-1	-1	-1	-1	-1	-1	-1	-1
g[u]	0	0	0	0	0	0	0	0	0	0
can[u]										

K: 可以左移的距離

now: 下一個開始的通道

value[u]: 通道 u 打通需要的代價 (-1 表示無窮大)

如何計算左移代價 (以 row 0 為例)



輸入的資料

i	0	1	2	3	4	5	6	7
blo[i]	1	2	1	0	1	2	0	1

	0	1	2	3	4
zero[i]					

左邊 0 的個數 zn: 0

0 的所在的位置

now now

通道	0	1	2	3	4	5	6	7	8	9
	blo[0]	blo[1]		blo[2]	blo[3]	blo[4]	blo[5]		blo[6]	blo[7]

blo[1] = 1 () → now+=blo[1] k=min(zn,blo[1])=0

現在 blo[1] 書櫃無法左移

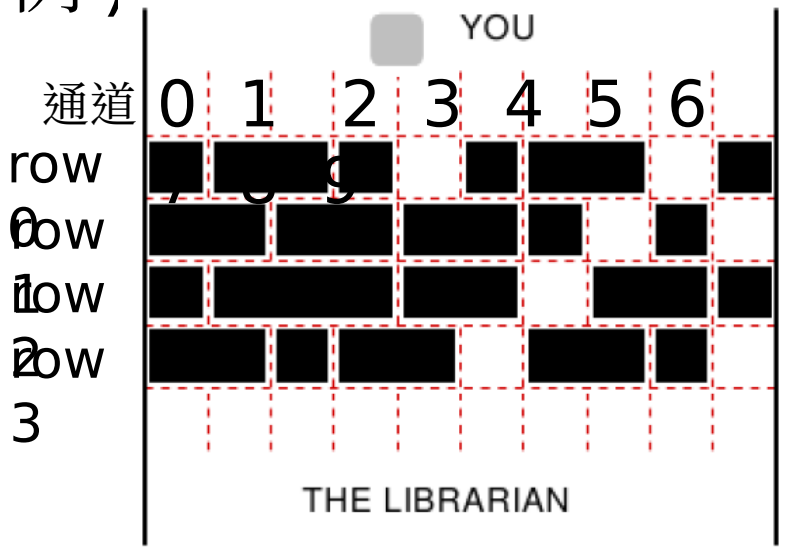
通道 u	0	1	2	3	4	5	6	7	8	9
value[u]	=1	-1	-1	-1	-1	-1	-1	-1	-1	-1
g[u]	0	0	0	0	0	0	0	0	0	0
can[u]										

K: 可以左移的距離

now: 下一個開始的通道

value[u]: 通道 u 打通需要的代價 (-1 表示無窮大)

如何計算左移代價 (以 row 0 為例)



輸入的資料

i	0	1	2	3	4	5	6	7
blo[i]	1	2	1	0	1	2	0	1

	0	1	2	3	4
zero[i]					

左邊 0 的個數 zn: 0

0 的所在的位置 now now

通道	0	1	2	3	4	5	6	7	8	9
	blo[0]	blo[1]		blo[2]	blo[3]	blo[4]	blo[5]		blo[6]	blo[7]

blo[2] = 1 () → now+=blo[2] k=min(zn,blo[2])=0

現在 blo[2] 書櫃無法左移

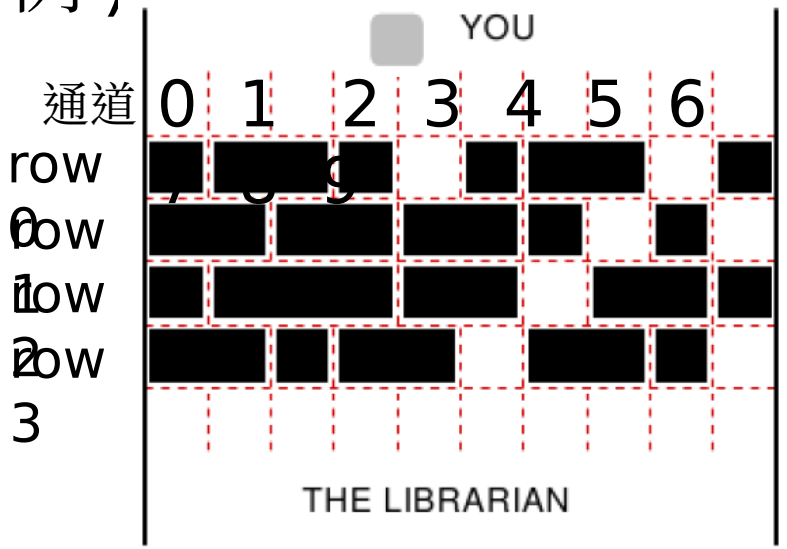
通道 u	0	1	2	3	4	5	6	7	8	9
value[u]	=1	-1	-1	-1	-1	-1	-1	-1	-1	-1
g[u]	0	0	0	0	0	0	0	0	0	0
can[u]										

K: 可以左移的距離

now: 下一個開始的通道

value[u]: 通道 u 打通需要的代價 (-1 表示無窮大)

如何計算左移代價 (以 row 0 為例)



輸入的資料

i	0	1	2	3	4	5	6	7
blo[i]	1	2	1	0	1	2	0	1

	0	1	2	3	4
zero[i]	3				

左邊 0 的個數 zn: 1

0 的所在的位置 j: 0

now now

通道	0	1	2	3	4	5	6	7	8	9
	blo[0]	blo[1]		blo[2]	blo[3]	blo[4]	blo[5]		blo[6]	blo[7]

blo[3] = 0 → zero[zn++]=3 ; can[now]++; now++;

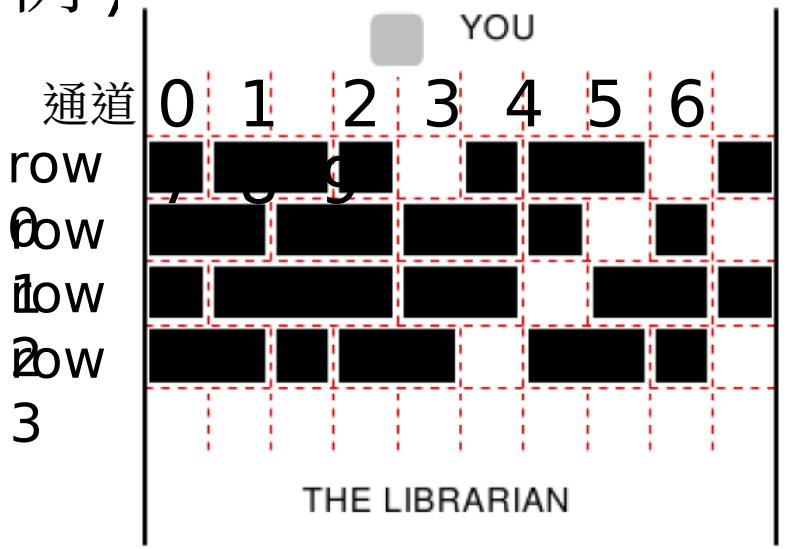
通道 u	0	1	2	3	4	5	6	7	8	9
value[u]	=1	-1	-1	-1	-1	-1	-1	-1	-1	-1
g[u]	0	0	0	0	0	0	0	0	0	0
can[u]					1					

K: 可以左移的距離

now: 下一個開始的通道

value[u]: 通道 u 打通需要的代價 (-1 表示無窮大)

如何計算左移代價 (以 row 0 為例)



輸入的資料

i	0	1	2	3	4	5	6	7
blo[i]	1	2	1	0	1	2	0	1

	0	1	2	3	4
zero[i]	3				

左邊 0 的個數 zn: 1

0 的所在的位置 0

now now

通道	0	1	2	3	4	5	6	7	8	9
	blo[0]	blo[1]		blo[2]	blo[3]	blo[4]	blo[5]		blo[6]	blo[7]

blo[4] = 1 () → now+=blo[4] k=min(zn,blo[4])=1 現在 blo[4] 書櫃可以左移 1 個距離

→ value[now-1]=i-zero[zn-1]-1+1=1 (value[5]=1);can[now-1]++;

g[now-1]+=value[5];

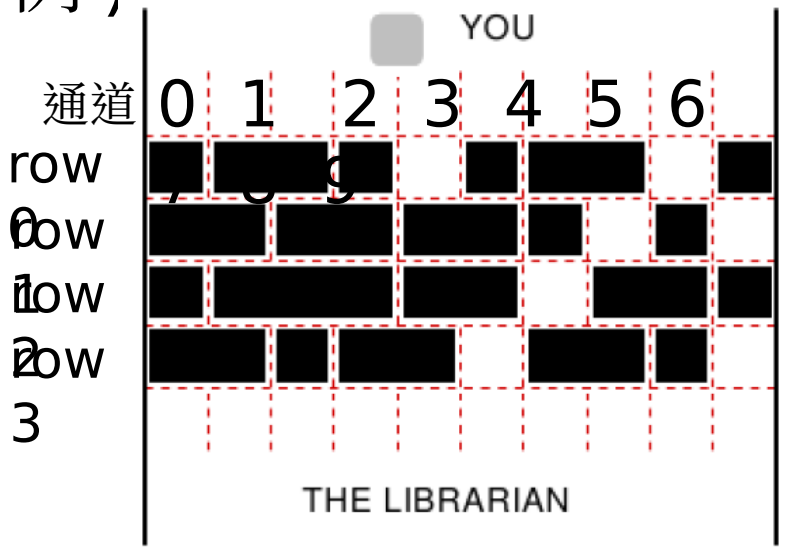
通道 u	0	1	2	3	4	5	6	7	8	9
value[u]	=1	-1	-1	-1	-1	1	-1	-1	-1	-1
g[u]	0	0	0	0	0	0	0	0	0	0
can[u]					1	1				

K: 可以左移的距離

now: 下一個開始的通道

value[u]: 通道 u 打通需要的代價 (-1 表示無窮大)

如何計算左移代價 (以 row 0 為例)



輸入的資料

i	0	1	2	3	4	5	6	7
blo[i]	1	2	1	0	1	2	0	1

	0	1	2	3	4
zero[i]	3				

左邊 0 的個數zn: 1

0 的所在的位置

now now

通道	0	1	2	3	4	5	6	7	8	9
	blo[0]	blo[1]		blo[2]	blo[3]	blo[4]	blo[5]		blo[6]	blo[7]

blo[5] = 2 () → now+=blo[5]; → k=min(zn,blo[5])=1 現在 blo[5] 書櫃可以左移 1 個距離

→ value[now-1]=i-zero[zn-1]-1+1=2 (value[7]=1);can[now-1]++;

g[now-1]+=value[5];

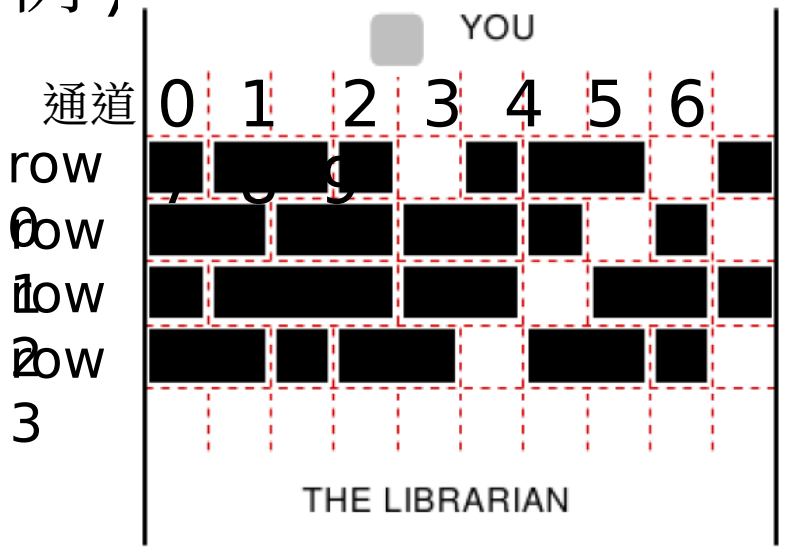
通道 u	0	1	2	3	4	5	6	7	8	9
value[u]	=1	-1	-1	-1	-1	1 1	-1	2 2	-1	-1
g[u]	0	0	0	0	0 1	0 1	0	0 1	0	0
can[u]										

K: 可以左移的距離

now: 下一個開始的通道

value[u]: 通道 u 打通需要的代價 (-1 表示無窮大)

如何計算左移代價 (以 row 0 為例)



輸入的資料

i	0	1	2	3	4	5	6	7
blo[i]	1	2	1	0	1	2	0	1

	0	1	2	3	4
zero[i]	3	6			

左邊 0 的個數 zn: 2

0 的所在的位置

now now

通道	0	1	2	3	4	5	6	7	8	9
	blo[0]	blo[1]		blo[2]	blo[3]	blo[4]	blo[5]		blo[6]	blo[7]

blo[6] = 0 → zero[zn++]=6 ; can[now]++; now++;

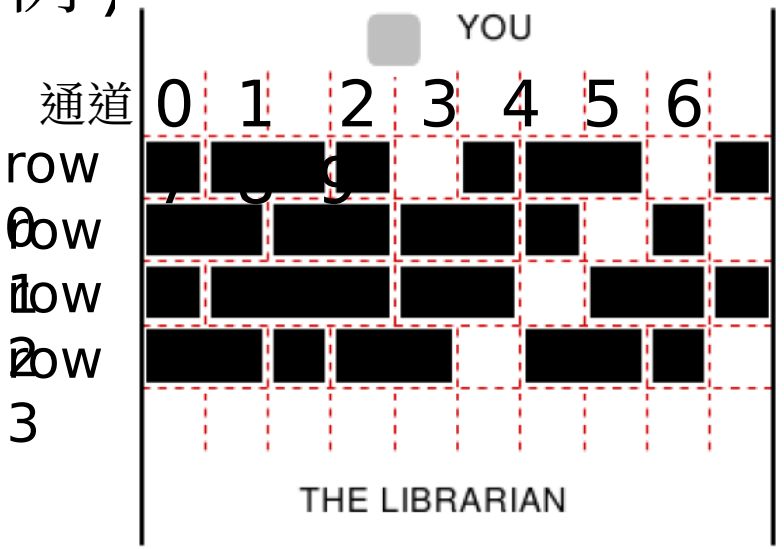
通道 u	0	1	2	3	4	5	6	7	8	9
value[u]	=1	-1	-1	-1	-1	1 1	-1	2 2	-1	-1
g[u]	0	0	0	0	0	0	0	0	0	0
can[u]					1	1		1		

K: 可以左移的距離

now: 下一個開始的通道

value[u]: 通道 u 打通需要的代價 (-1 表示無窮大)

如何計算左移代價 (以 row 0 為例)



輸入的資料

i	0	1	2	3	4	5	6	7
blo[i]	1	2	1	0	1	2	0	1

	0	1	2	3	4
zero[i]	3	6			

左邊 0 的個數 zn: 2

0 的所在的位置

now now

通道	0	1	2	3	4	5	6	7	8	9
	blo[0]	blo[1]		blo[2]	blo[3]	blo[4]	blo[5]		blo[6]	blo[7]

blo[7] = 1 () → now+=blo[7] → k=min(zn,blo[7])=1

現在 blo[7] 書櫃可以左移 1 個距離

→ value[now-1]=i-zero[zn-1]-1+1=1 (value[9]=1);can[now-1]++;

g[now-1]+=value[9];

通道 u	0	1	2	3	4	5	6	7	8	9
value[u]	=1	-1	-1	-1	-1	1 1	-1	2 2	-1	1 1
g[u]	0	0	0	0	0 1	0 1	0	0 1	0 1	0 1
can[u]										

K: 可以左移的距離

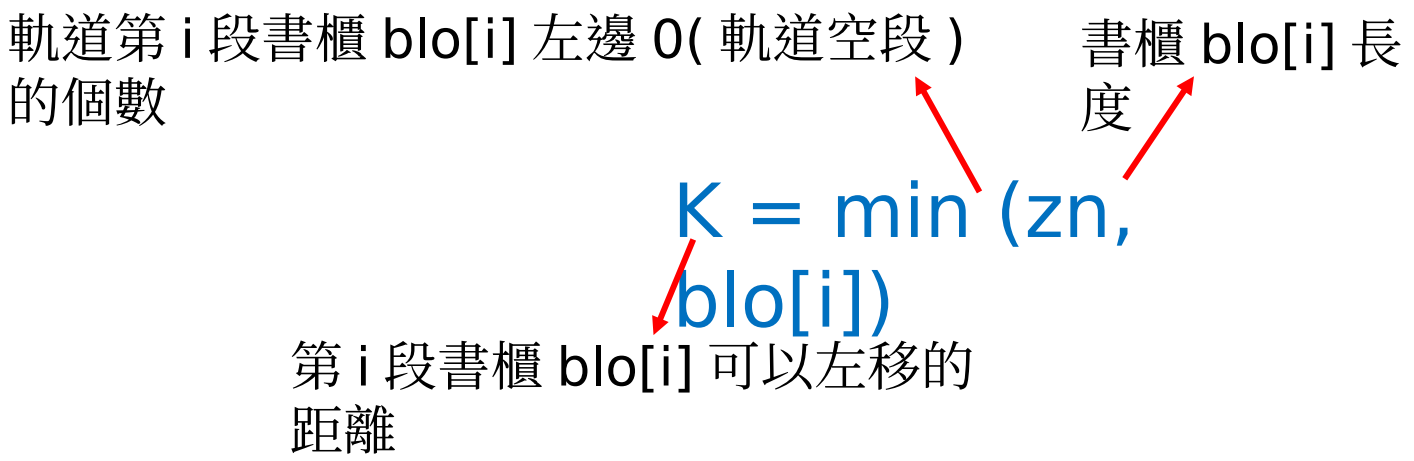
now: 下一個開始的通道

value[u]: 通道 u 打通需要的代價 (-1 表示無窮大)

計算左移代價

目的 : 找出 blo[i] 所佔之通道是否可以打通 ? 其代價為何 ?

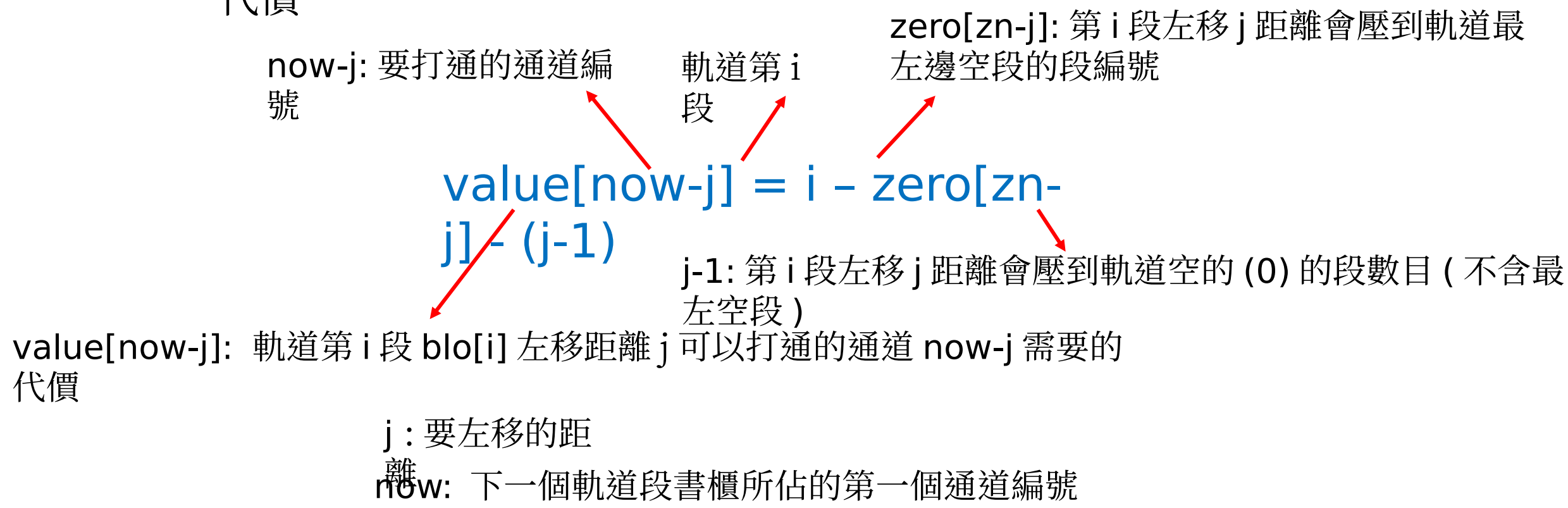
- 首先計算軌道第 i 段書櫃 blo[i] 可以左移的距離



計算左移代價

目的 : 找出 blo[i] 所佔之通道是否可以打通 ? 其代價為何 ?

- 接著計算左移距離 j (從 1 到 k) 分別計算軌道第 i 段書櫃 blo[i] 可以打通的通道代價



計算左移代價

- 每次算完左移代價即更新 g[now-j] 與 can[now-j] 值

如何計算右移代價

- 將輸入資料 blo 陣列左右翻轉 , 以計算左移代價方式計算
- 由於要算左移與右移代價最小值 , 在計算左移代價時先存入 g[now-j], 因此算出右移代價時取最小值之際 , 可能要修正 g[now-j] (因為之前左移代價不是最小值) 。

LA 4629 Code (1/5)

```
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;

const int INF = 0x3f3f3f3f;
const int N = 1000005;

int t, R, L, n, blo[N], zero[N], value[N], can[N], g[N];
```

LA 4629 Code (2/5)

```
void build() {
    scanf("%d", &n);
    int zn = 0, now = 0;
    memset(value, -1, sizeof(value));
    for (int i = 0; i < n; i++) {        //  逐一讀入軌道每一段資料 blo[i], 計算左移代價
        scanf("%d", &blo[i]);
        if (blo[i] == 0) {               // 空段
            zero[zn++] = i;              // 記錄空段
            can[now]++;                  // 此軌道通道 now 可以通過，記錄通道 now 可以通過的軌道數 can[now]
            now++;                       // 更新下一段書櫃開始的通道編號 now
        }  else {
            now += blo[i];                    // 更新下一段書櫃開始的通道編號 now
            int k = min(zn, blo[i]);          // K: 第 i 段書櫃 blo[i] 可以左移的距離
            for (int j = 1; j <= k; j++) {
                value[now - j] = i - zero[zn - j] - j + 1;
                                                  // value[now-j]:  軌道第 i 段 blo[i] 左移距離 j 可以打通的通
now-j 需要的代價
                can[now - j]++;               // 更新 g[now-j] 與 can[now-j] 值
                g[now - j] += value[now - j];
            }
        }
    }
```

LA 4629 Code (3/5)

```
reverse(blo, blo + n);  // 輸入資料 blo 陣列左右翻轉 , 以計算左移代價方式計算右移代價
zn = 0; now--;
for (int i = 0; i < n; i++) {  //  逐一讀入軌道每一段資料 blo[i], 計算左移代價
    if (blo[i] == 0) {          // 空段
        zero[zn++] = i;         // 記錄空段
        now--;                  // 更新下一段書櫃開始的通道編號 now
     }  else {
        now -= blo[i];                      // 更新下一段書櫃開始的通道編號 now
        int k = min(zn, blo[i]);            // K: 第 i 段書櫃 blo[i] 可以左移的距離
        for (int j = 1; j <= k; j++) {
                    if (value[now + j] == -1) {  // 之前計算左移時通道 now+j 是不通的
        value[now + j] = i - zero[zn - j] - j + 1;
                                                 // value[now+j]: 軌道第 i 段 blo[i] 左移距離 j 可以打通的通道
now+j 需要的代價
        g[now + j] += value[now + j];  // 更新 g[now-j] 與 can[now-j] 值
        can[now + j]++;
                     } else {   // 之前計算左移時通道 now+j 是通的
            int tmp = i - zero[zn - j] - j + 1;   // tmp: 右移的代價
            g[now + j] += min(0, tmp - value[now + j]);
                                                 // 可能要修正 g[now+j] ( 因為之前左移代價
最小值 )
                }
          }
        }
}
```

LA 4629 Code (4/5)

```
void solve() {        // 找最後答案
    int an = 0, Min = INF;
    for (int i = 0; i < L; i++) {
        if (can[i] != R) continue;   // R: 軌道數
        if (Min > g[i]) {
            Min = g[i];
            an = 0;
            value[an++] = i;
         }  else if (Min == g[i]) { value[an++] = i; }
    }
    printf("%d\n", Min);
    for (int i = 0; i < an; i++)
       if (i==an-1) printf("%d", value[i]); else printf("%d ", value[i]);
    printf("\n");
}
```

LA 4629 Code (5/5)

```
int main() {
   freopen("4629.in","r",stdin);
   freopen("4629.out","w",stdout);
   scanf("%d", &t);
    while (t--) {
          scanf("%d%d", &R, &L);
          memset(can, 0, sizeof(can));
          memset(g, 0, sizeof(g));
          for (int i = 0; i < R; i++)    // 處理每一軌道 (row)
              build();
          solve();                       // 找最後答案
    }
    return 0;
}
```