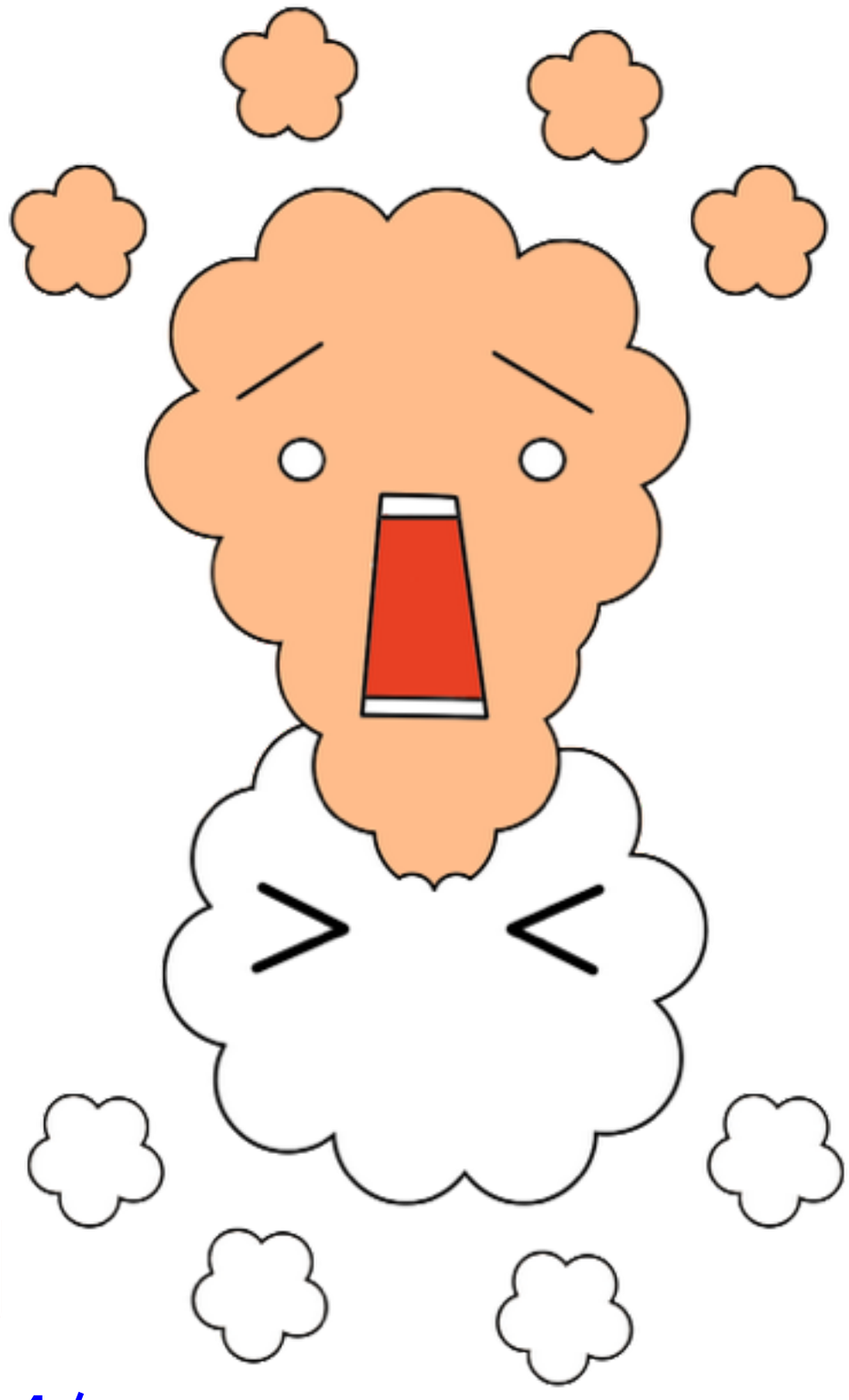
練習：踩地雷

題號：214



HINT : 二維陣列 + 巢狀回圈

http://2015.sprout.csie.org/oj/pro/214/

踩地雷(214)

給一個地雷盤面，問每格周圍八格有多少個地雷

輸入格式

第1行有兩個正整數n和m(1≤n,m≤100)，代表著盤面的長和寬

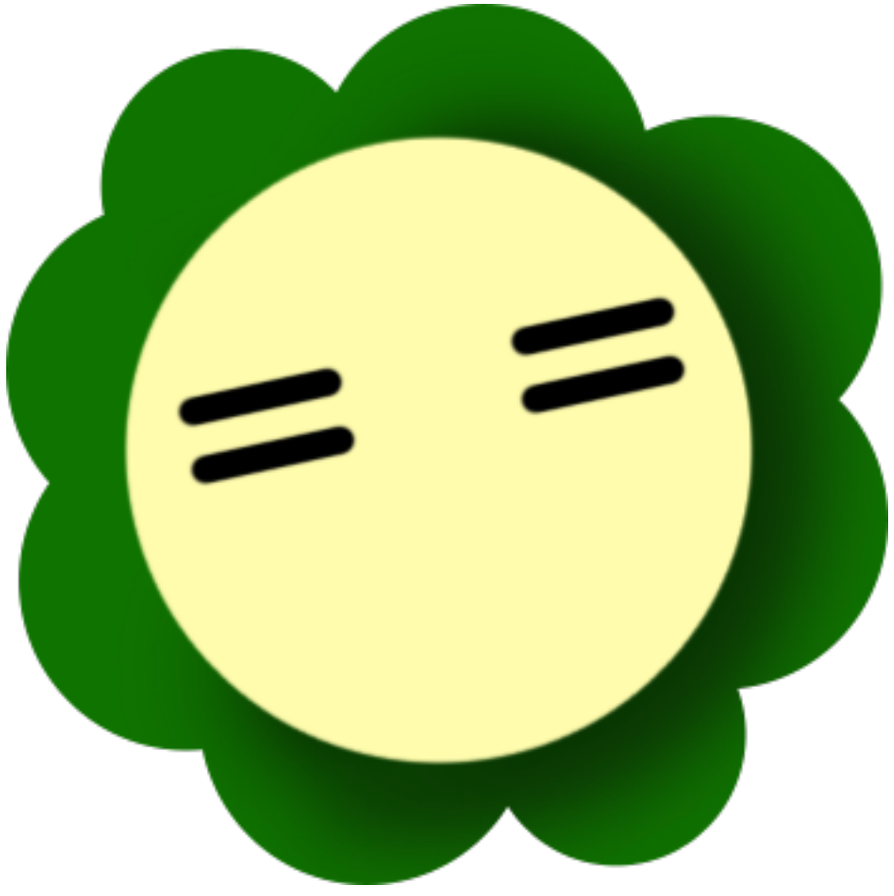
第2行到第n+1行，每行會有m個數字，0表示沒有地雷，1表示有地雷。

輸出格式

請輸出n行，每行會有m個數字，代表該格周圍八格有多少地雷

踩地雷(214)

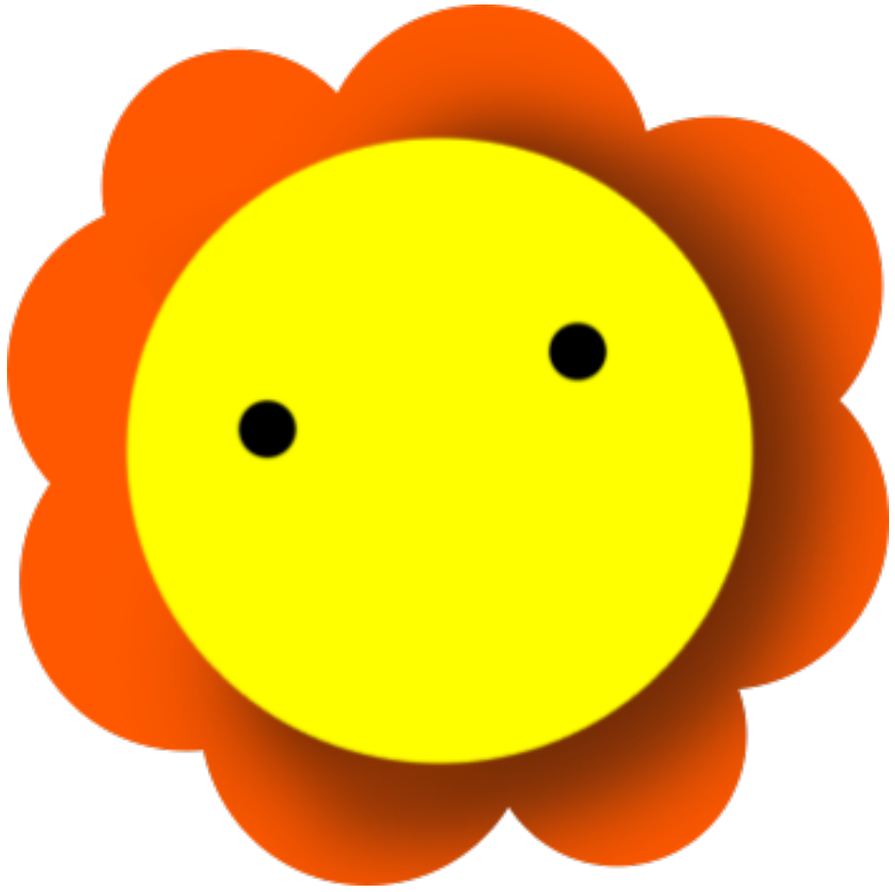
最直覺的想法：統計



踩地雷(214)

大方向：

計算每一個點的周圍



有幾個地雷

踩地雷(214)

談計算之前...

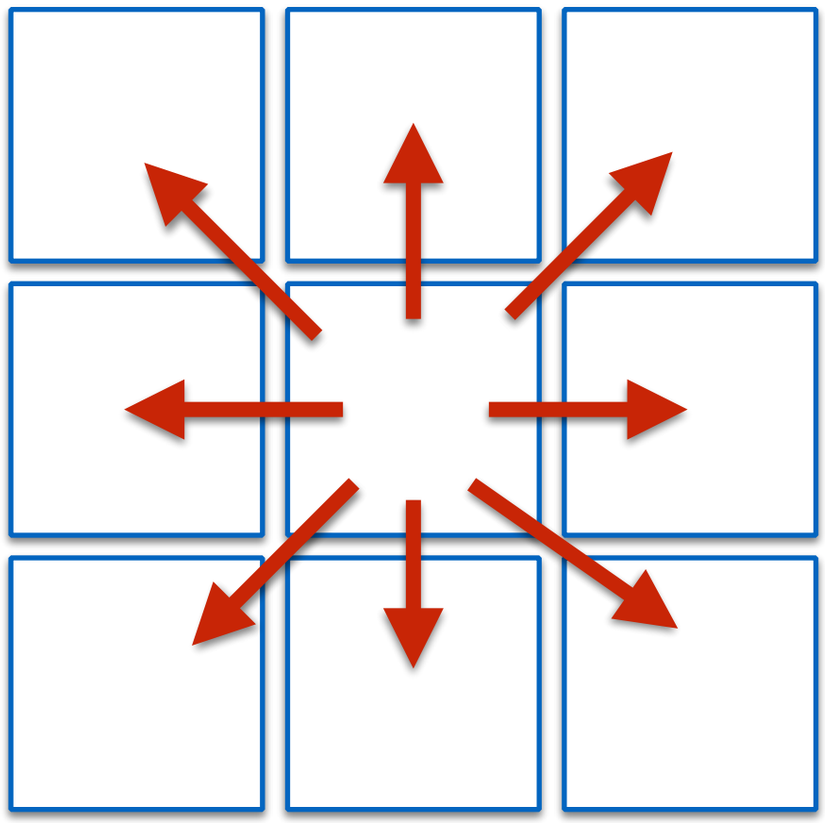
我們先來談談**周圍**

踩地雷(214)

周圍：八方位

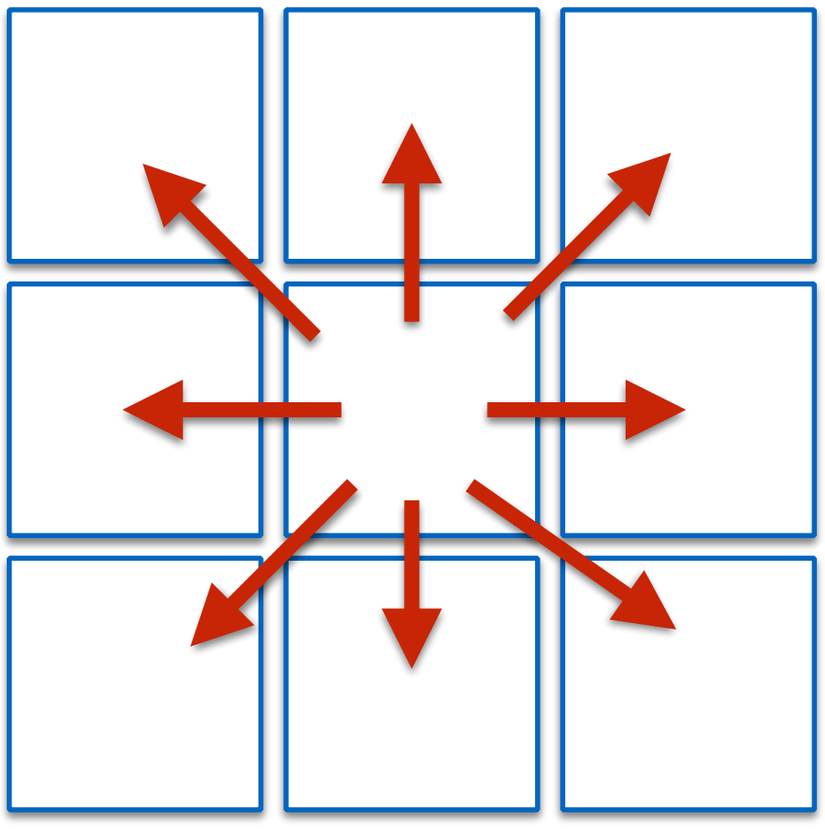
踩地雷(214)

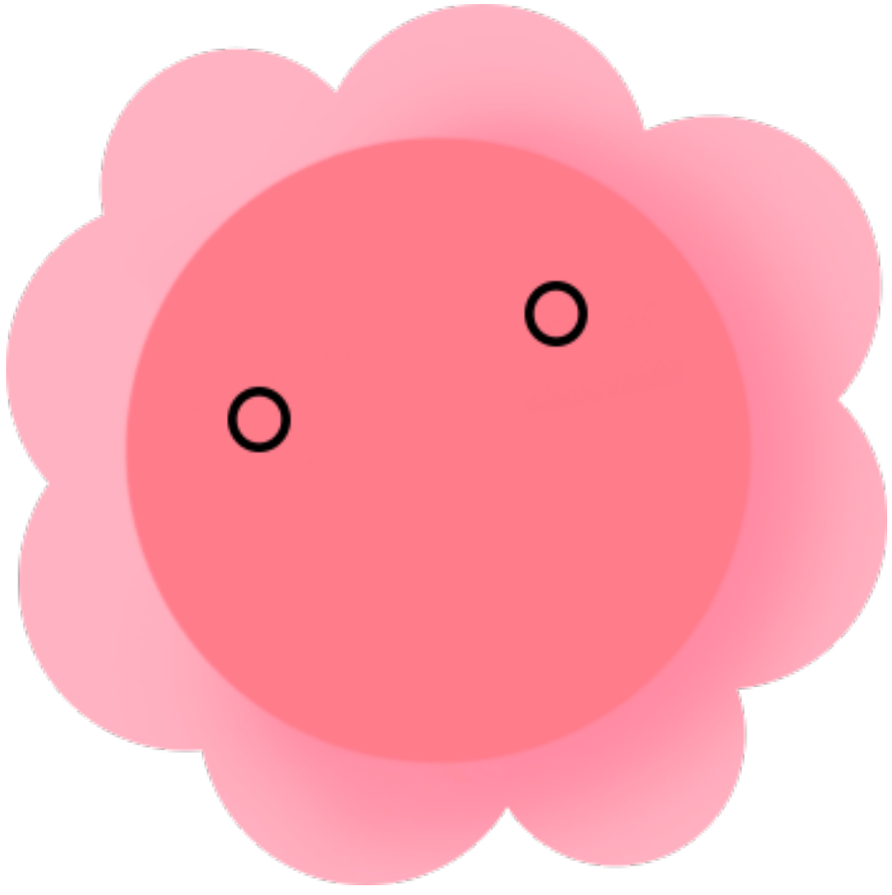
檢查：八方位



踩地雷(214)

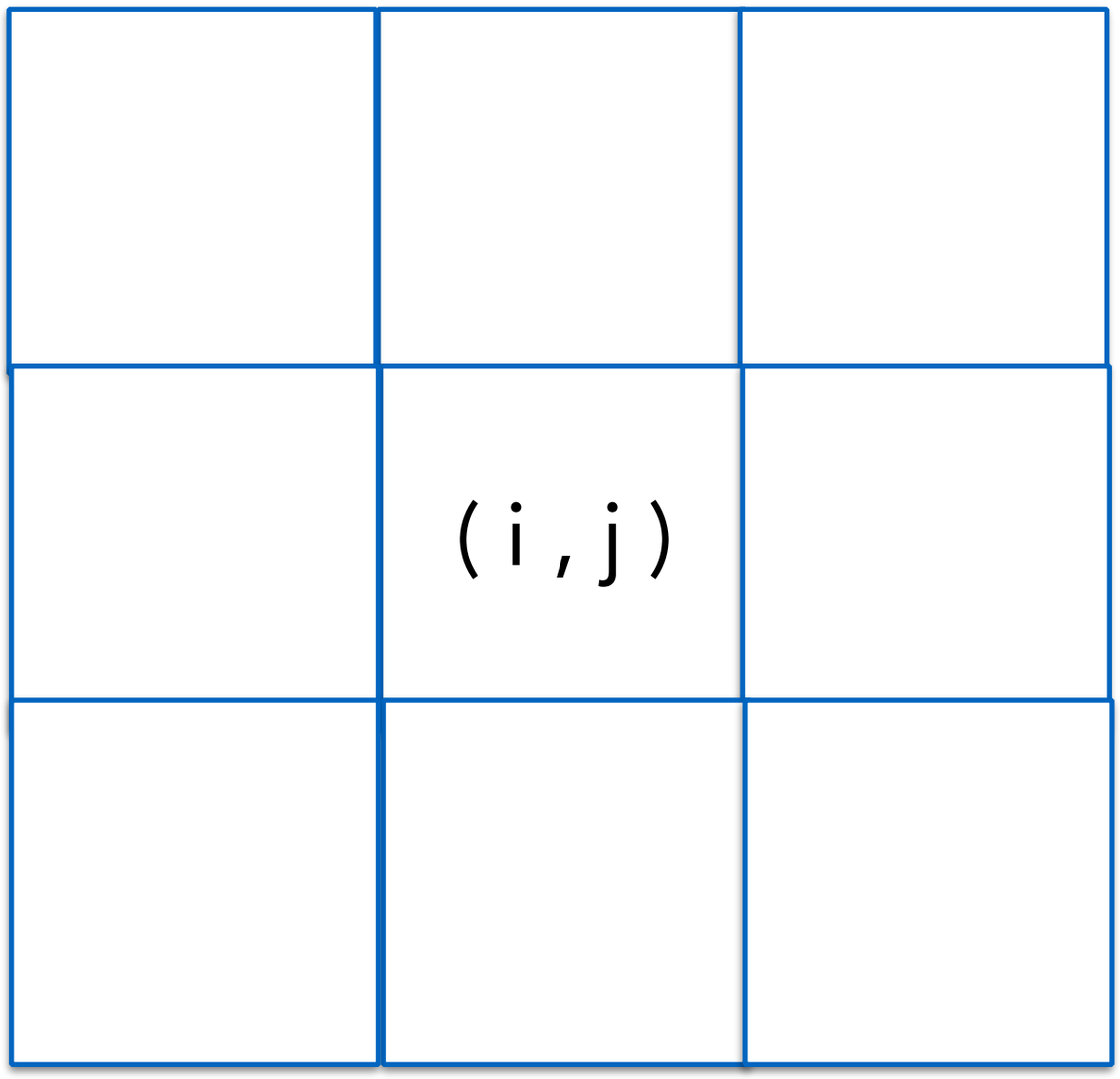
一個一個檢查？





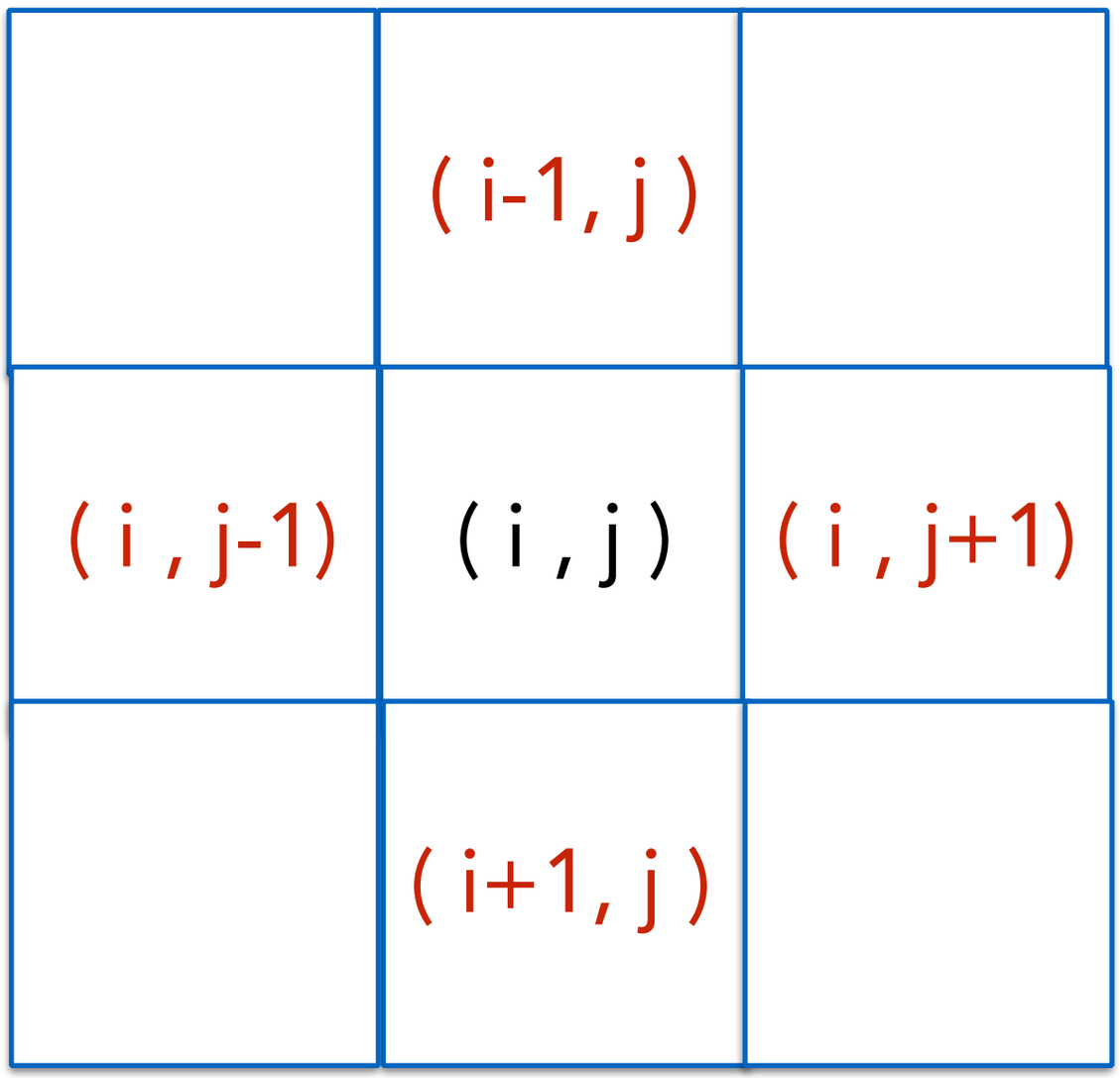
踩地雷(214)

先來觀察：



踩地雷(214)

先來觀察：



踩地雷(214)

先來觀察：

(i-1, j-1)	(i-1, j)	(i-1, j+1)
(i , j-1)	(i , j)	(i , j+1)
(i+1, j-1)	(i+1, j)	(i+1, j+1)

踩地雷(214)

歸納：

(i-1, j-1)	(i-1, j)	(i-1, j+1)
(i , j-1)	(i , j)	(i , j+1)
(i+1, j-1)	(i+1, j)	(i+1, j+1)

(-1, -1)	(-1, 0)	(-1, +1)
(0, -1)	(0 , 0)	(0, +1)
(+1, -1)	(+1, 0)	(-1, -1)

踩地雷(214)

建表：

```
int d[8][2] = {
        {1,  0},
        {0,  1},
        {-1, 0},
        {0, -1},
        {1,  1},
        {1, -1},
        {-1, 1},
        {-1,-1}
};
```

踩地雷(214)

```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

設當前點為 (i , j)

踩地雷(214)

```
int d[8][2] = {
        {1,  0},
        {0,  1},
        {-1, 0},
        {0, -1},
        {1,  1},
        {1, -1},
        {-1, 1},
        {-1,-1}
};
```

設當前點為 (i , j)

設 k = 0 ~ 7

踩地雷(214)

```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

設當前點為 (i , j)

設 k = 0 ~ 7

周圍的點為 (i + d[k][0] , j + d[k][1])

踩地雷(214)

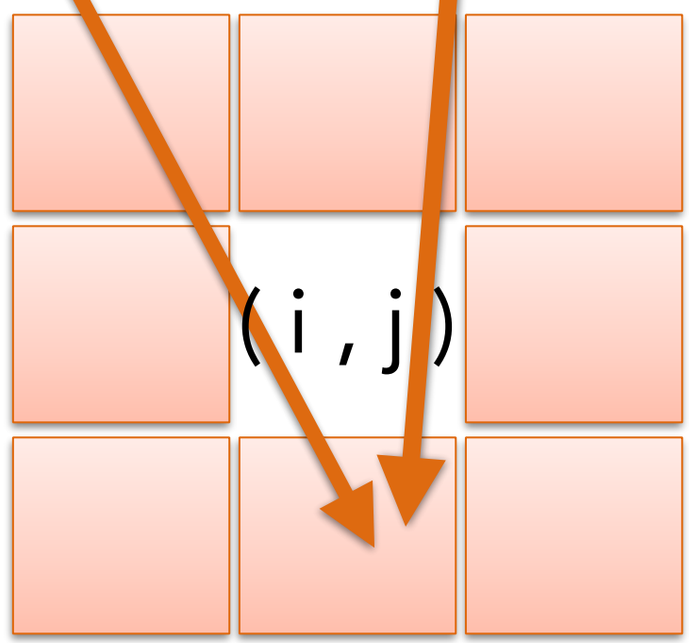
```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

設當前點為 (i , j)

設 k = 0 ~ 7

周圍的點為 (i + d[k][0] , j + d[k][1])

k = 0



踩地雷(214)

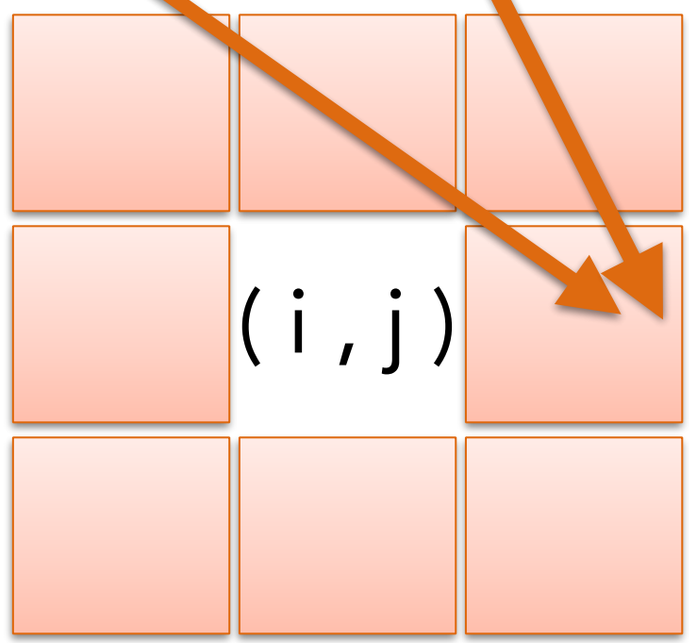
```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

設當前點為 (i , j)

設 k = 0 ~ 7

周圍的點為 (i + d[k][0] , j + d[k][1])

k = 1



踩地雷(214)

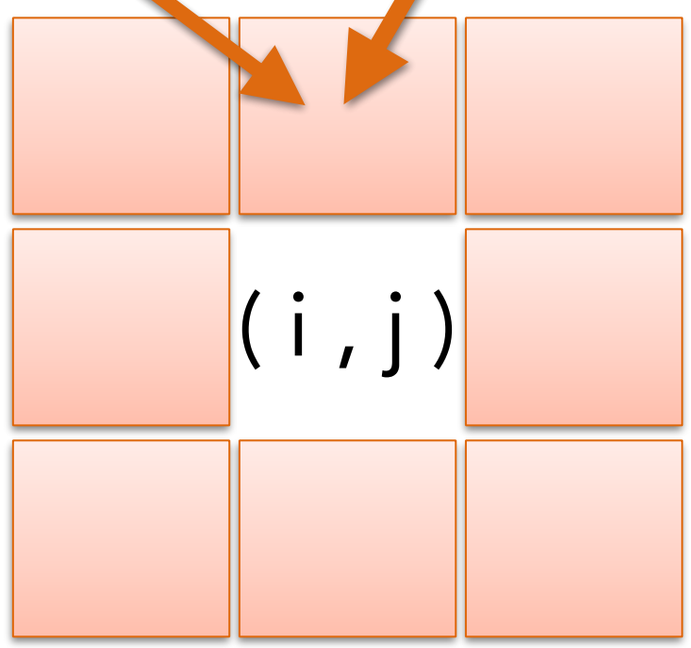
```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

設當前點為 (i , j)

設 k = 0 ~ 7

周圍的點為 (i + d[k][0] , j + d[k][1])

k = 2



踩地雷(214)

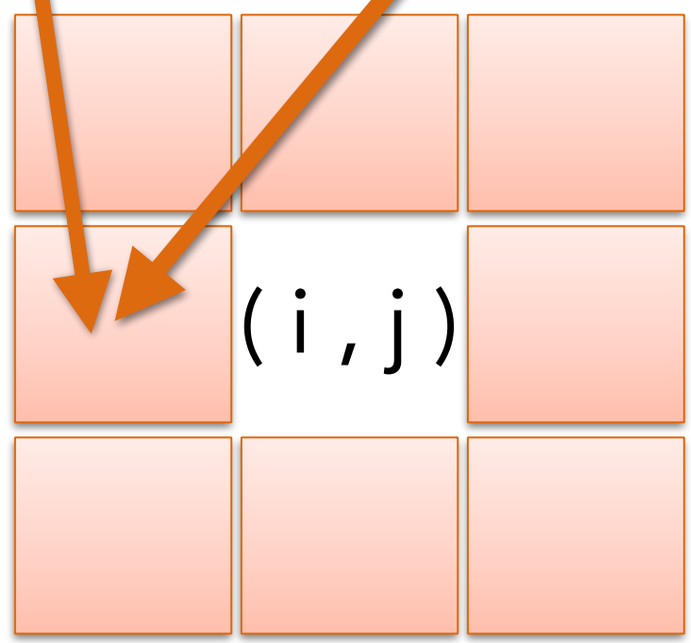
```
int d[8][2] = {
    {1,  0},
    {0,  1},
    {-1, 0},
    {0, -1},
    {1,  1},
    {1, -1},
    {-1, 1},
    {-1,-1}
};
```

設當前點為 (i , j)

設 k = 0 ~ 7

周圍的點為 (i + d[k][0] , j + d[k][1])

k = 3



踩地雷(214)

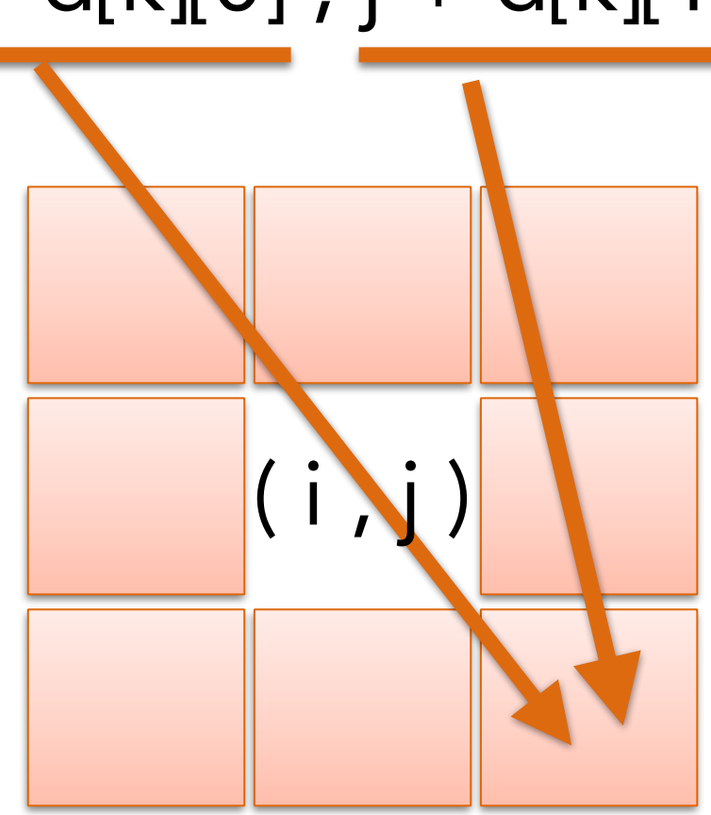
```
int d[8][2] = {
        {1,  0},
        {0,  1},
        {-1, 0},
        {0, -1},
        {1,  1},
        {1, -1},
        {-1, 1},
        {-1,-1}
};
```

設當前點為 (i , j)

設 k = 0 ~ 7

周圍的點為 (i + d[k][0] , j + d[k][1])

k = 4



踩地雷(214)

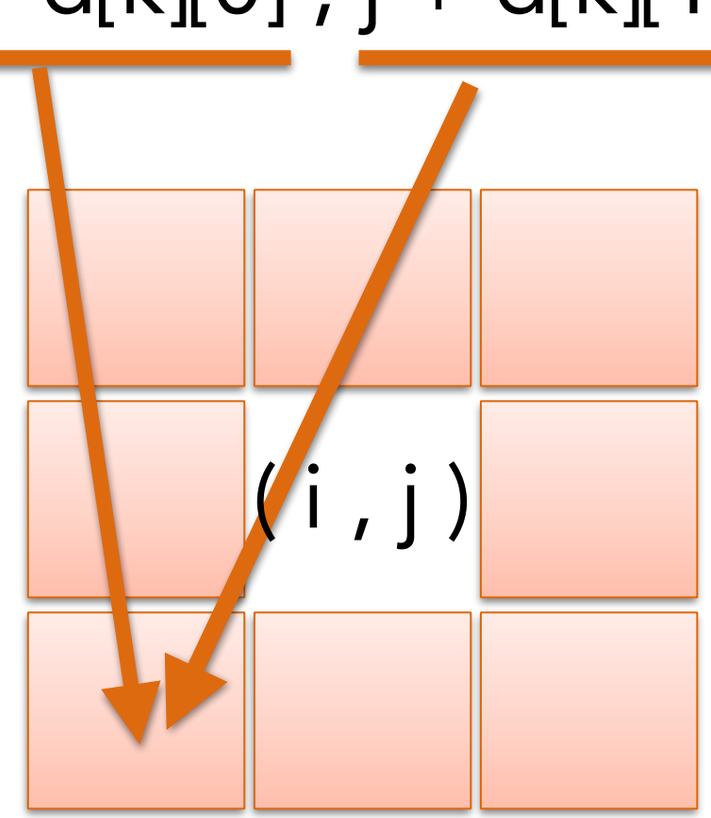
```
int d[8][2] = {
        {1,  0},
        {0,  1},
        {-1, 0},
        {0, -1},
        {1,  1},
        {1, -1},
        {-1, 1},
        {-1,-1}
};
```

設當前點為 (i , j)

設 k = 0 ~ 7

周圍的點為 (i + d[k][0] , j + d[k][1])

k = 5



踩地雷(214)

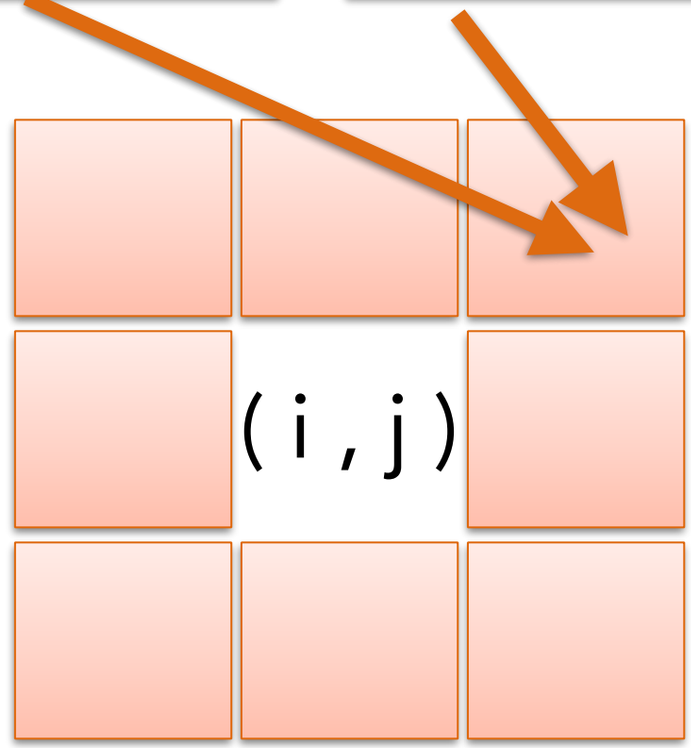
```
int d[8][2] = {
        {1,  0},
        {0,  1},
        {-1, 0},
        {0, -1},
        {1,  1},
        {1, -1},
        {-1, 1},
        {-1,-1}
};
```

設當前點為 (i , j)

設 k = 0 ~ 7

周圍的點為 (i + d[k][0] , j + d[k][1])

k = 6



踩地雷(214)

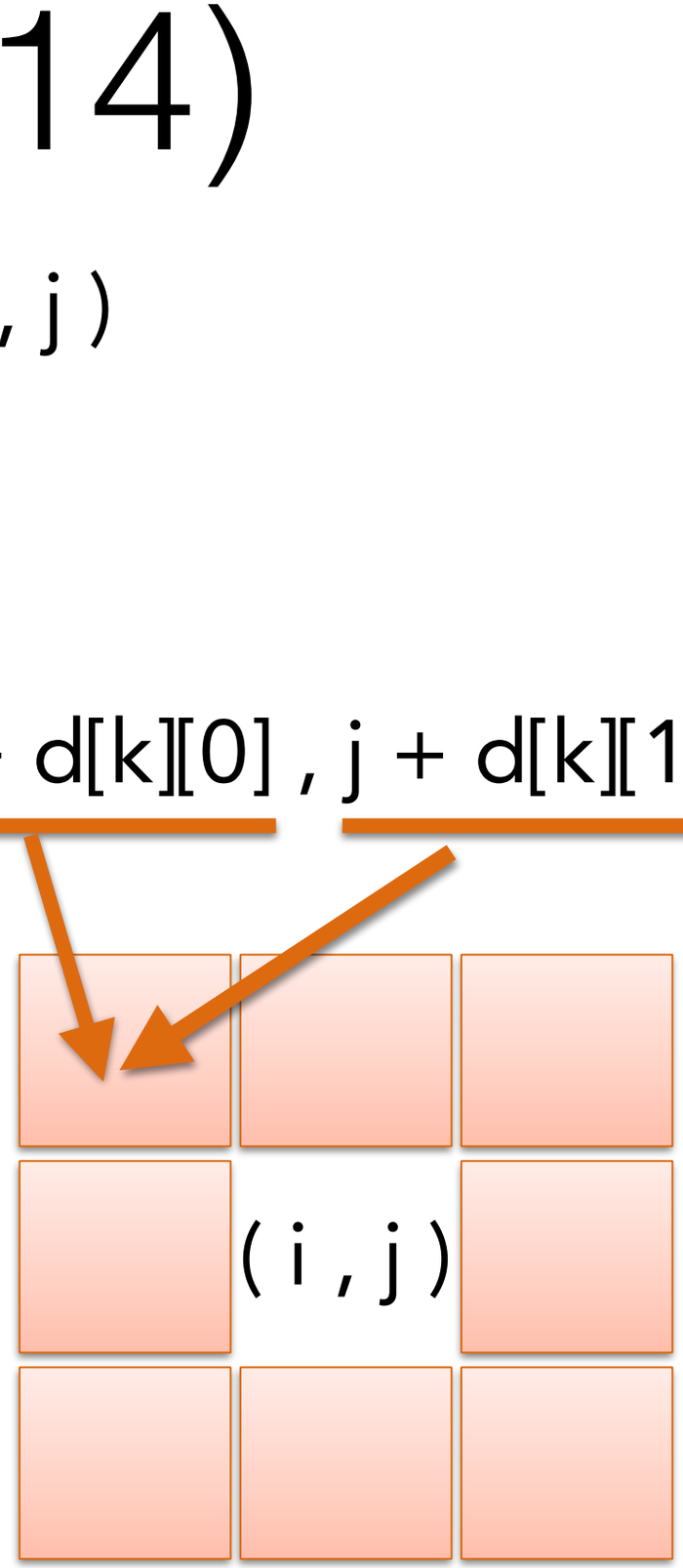
```
int d[8][2] = {
        {1,  0},
        {0,  1},
        {-1, 0},
        {0, -1},
        {1,  1},
        {1, -1},
        {-1, 1},
        {-1,-1}
};
```

設當前點為 (i , j)

設 k = 0 ~ 7

周圍的點為 (i + d[k][0] , j + d[k][1])

k = 7



踩地雷(214)

幫你節省不少程式碼ＸＤ

踩地雷(214)

最後.....

踩地雷(214)

注意邊界！！！！！

注意邊界！！！！！

~~很重要所以要說三次~~

踩地雷(214)