

|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

# 期末考

## Final Exam

長榮大學  
資訊工程學系

班級：資工 2B

姓名：郭智榮

學號：109B30612

日期：2022/06/13

版本校定紀錄：

| 版本      | 更新紀錄 | 發佈日期       |
|---------|------|------------|
| 0.0.0.0 | 初版完成 | 2022/06/13 |
|         |      |            |
|         |      |            |

|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

## 一. 實驗需求：

### (一). 題目說明

描述：

有一個方格棋盤的每一個格子裡都標示了一個不同的整數，有一個機器人在此方格棋盤上行動，每一次只會移動到目前位置的上下左右四個相鄰格子其中之一。

起點須為數值最小的格子，每次移動則在可以移動位置中挑選數值最小的格子，但是走過的格子就不會再走，當無路可走的時候，機器人就會停下來。

輸入方格棋盤中每個格子的數值，請模擬機器人走過的路徑；輸出機器人走過的格子的數值總和。

輸入說明：

第一行是兩個不超過 100 的正整數  $m$  與  $n$ ，代表是一個  $m * n$  的方格棋盤。

接著有  $m$  行，每一行  $n$  個整數，分別是方格棋盤由上而下，由左而右的數字。方格內的數值皆不超過  $10^5$ ，同一行數值以空白間隔。

輸出說明：

機器人走過的格子中數值的總和。

|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

## (二). 解題思維

以下所呼叫之變數名稱若沒有特別設定則以資料型態為變數名稱，以方便前後文閱讀時不需重複查看變數名稱。

### i. 建置所需的型態陣列

#### 1. priority\_queue：

運用 **優先佇列會自動排序** 的特點，將 **x 的鄰居(上、下、左、右)** 放入對應的 **priority\_queue 陣列** 位置，並呼叫 **priority\_queue[x].top()** 快速獲得 **x 的最小鄰居數值**。

※ 此處所宣告的優先佇列，採用 **數值較小優先權較高**，宣告方式如下

「`priority_queue<int,vector<int>,greater<int>>` 變數名稱」

此優先佇列假設放入順序為：1, 5, 3, 6, 7, 2

則優先佇列取出順序為：1, 2, 3, 5, 6, 7

#### 2. vector：

使用 **vector 儲存兩個數值**，分別為 **該數值位置的行與列**，後續便可呼叫 **vector[x]** 快速獲取 **x 的位置**，省去尋找數值位置的時間。並在後續 **紀錄位置已造訪時**，快速獲取正在造訪數值的行、列位置並提供紀錄。

#### 3. bool：

使用 **bool 紀錄某位置是否被造訪**，若 **bool[row][column]** 為 **true** 代表被造訪，反之若為 **false** 則為未造訪，因題目有規定 **以造訪不可重複**，因此使用 **boolean** 相較其他紀錄方式可能較為簡單、方便。

#### 4. int：

使用 **int 紀錄某位置的數值**，例如 **int[row][column]** 為 15，藉此紀錄每個位置的數值，在新增鄰居至 **priority\_queue** 時，能 **快速且簡易的定位上下左右的鄰居**。

|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

ii. 透過以下方式將 x 的鄰居放入 priority\_queue[x] 中

數值新增鄰居的狀況會有三個：

1. 當目前輸入位置在第一行時不會新增上方數值(因為不存在)

|   |     |     |
|---|-----|-----|
| 1 | ← 2 | ← 5 |
| 4 | x   | 6   |

圖 1. 輸入位置在第一行時的鄰居狀況

2. 當目前輸入位置在第一列時不會新增左方數值(因為不存在)

|     |   |   |
|-----|---|---|
| 1   | 2 | 5 |
| ↑ 4 | x | 6 |

圖 2. 輸入位置在第一列時的鄰居狀況

3. 當目前輸入位置不在第一行及第一列，則新增上方及左方數值

|   |     |   |
|---|-----|---|
| 1 | 2   | 5 |
| 4 | ↑ x | 6 |

圖 3. 輸入位置不在第一行及第一列的鄰居狀況

依照上述三點則最後鄰居新增會如圖 4 所示，箭頭顏色所觸發的時機會對應的字體顏色，例如：黃色 4 被輸入時，才有黃色箭頭的指向，便將紅色 1 新增為黃色 4 的鄰居。

|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

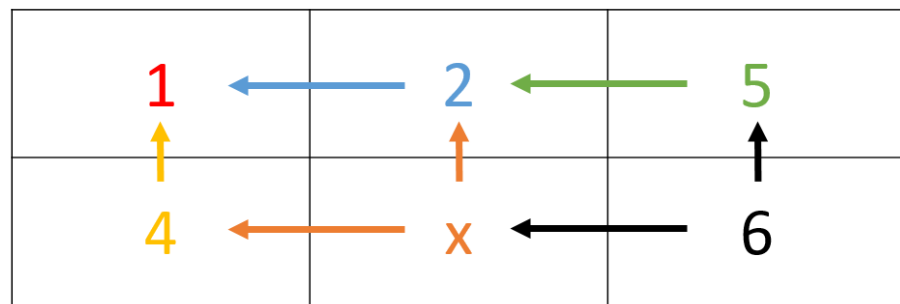


圖 4. 所有數值在輸入時新增的鄰居

然而，鄰居不僅是上方及左方，還有下方及右方；但因輸入到任一位置時，其下方及右方鄰居皆未被輸入，因此我們透過「數值 x 的上方鄰居的下方鄰居便是 x」這點，來獲得某位置的下方鄰居，我們透過反向的方式將箭頭位置反轉如圖 5，便能證明這一想法。

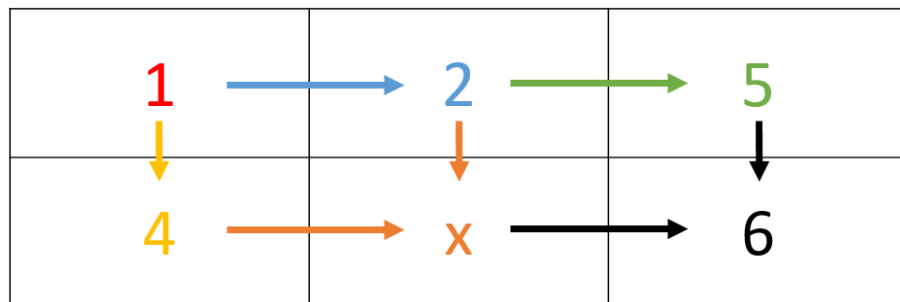


圖 5. 所有數字的下方及右方鄰居對應

箭頭顏色所觸發的時機會對應的字體顏色，例如：黃色 4 被輸入時，才有黃色箭頭的指向，便將黃色 4 新增為紅色 1 的鄰居。

由上述新增方式便可得到如圖 6 的鄰居狀況

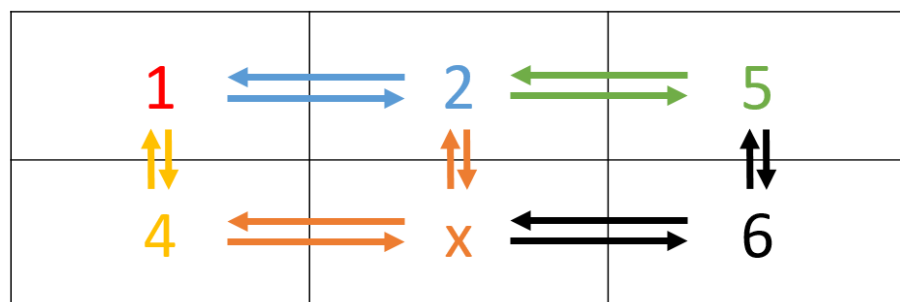


圖 6. 所有鄰居對應狀況

此部分由 2 \* 3 之表格呈現，但不論棋盤大小皆能透過此方式對應鄰居關係，但須注意數值是否重複之問題，若題目給予之數值會重複，則需改變鄰居記錄方式，以免發生有 2 個 5，導致其鄰居後續獲取位置錯誤。

|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

最終新增鄰居步驟如下：

step1. 由 `vector[x][0]` 的 first 及 second 獲取 x 位置的行、列

※ `vector[x][0]` 中的 [0] 是因為 `vector[x]` 也是陣列，因此需要再繼續取出第 0 個才能獲得數值位置的行與列

step2. 判斷是否為第一行，若是則跳過 step3 ~ step5

step3. `int up = value[行-1][列]`，其中 value 以位置儲存該格數值

step4. `priority_queue[x].push(up)`，新增 x 上方鄰居

step5. `priority_queue[up].push(x)`，新增 x 上方鄰居的下方鄰居

step6. 判斷是否為第一列，若是則跳過 step7 ~ step9

step7. `int left = value[行][列-1]`，其中 value 以位置儲存該格數值

step8. `priority_queue[x].push(left)`，新增 x 左方鄰居

step9. `priority_queue[left].push(x)`，新增 x 左方鄰居的右方鄰居

|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

iii. 透過以下方式做到快速尋找下一步的數值及位置：

假設：x 代表目前位置數值 15，其鄰居分別為 22, 12, 19, 21，如圖 7  
目前路徑為：12 → 15，紅色代表已造訪，橘色代表目前位置

|    |    |    |
|----|----|----|
| 31 | 22 | 25 |
| 19 | 15 | 21 |
| 33 | 12 | 55 |

圖 7. 假設情況

step1. 下一步為 `priority_queue[x].top()`，此處下一步為最小值 12

|    |    |    |
|----|----|----|
| 31 | 22 | 25 |
| 19 | 15 | 21 |
| 33 | 12 | 55 |

圖 8. 獲得鄰居最小值 12

step2. `priority_queue[x].pop()`，取出佇列最小值，以免重複

※ 此時 12 已被移出 `priority_queue[15]`，後續 `top()` 不會再獲得 12

※ 使用 `pop` 不影響後續程式，因為 `pop` 的數值若未被造訪則便是下一步。

step3. `bool[vector[x][0].first][vector[x][0].second]` 是否為真，  
若為真代表已被造訪，若不是則跳過 step4 ~ step7

※ 假設中的情況，12 已被造訪過，因此需要執行 step4 ~ step7

step4. 下一步設為無，以利後續判斷是否還需要執行下一步

step5. 判斷 `priority_queue[x]` 是否為空，若是則跳過 step6 ~ step8

|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

※ 假設後續 `priority_queue[15]` 為空，代表鄰居皆已確認被造訪過  
step6. 下一步為 `priority_queue[x].top()`，此處下一步為最小值 19

|    |    |    |
|----|----|----|
| 31 | 22 | 25 |
| 19 | 15 | 21 |
| 33 | 12 | 55 |

圖 9. 獲取鄰居最小值 19

step7. `priority_queue[x].pop()`，取出佇列最小值，以免重複  
step8. 回到 step3 (此處 step3 ~ step7 代表 while 迴圈判斷是否造訪)  
※ 若鄰居皆被造訪，則重複 step3 ~ step8 直到 `priority_queue` 為空。  
step9. 判斷下一步是否為無，若是則結束程式，若不是則 `x = 下一步`  
※ 目前情況下一步為 19，因此下方範例的 `x` 視為 19  
step10. 取出 `vector[x][0]` 的 first 及 second，分別為 `x` 的行與列位置  
step11. 將 `bool[行][列]` 設為已造訪，用於紀錄目前位置 `x` 已被造訪  
※ 此時狀況如圖 10 所示，路徑為：12 → 15 → 19

|    |    |    |
|----|----|----|
| 31 | 22 | 25 |
| 19 | 15 | 21 |
| 33 | 12 | 55 |

圖 10. 目前移動狀況

重複執行 step1 ~ step11 即可獲得答案(若無下一步將在 step9 中斷)，  
此情況的路徑為：12 → 15 → 19 → 31 → 22 → 25 → 21 → 55



|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

### (三). 演算法

#### i. 虛擬碼

定義 using namespace 型態的 std；//無此定義則 vector 宣告將報錯  
定義 max 為 100001；

宣告 vector 型態變數 place 儲存資料為兩個整數，大小為 max，並用於儲存「數值位置的行, 列」；

宣告 priority\_queue 型態變數 neighbor，用於儲存數值周遭的鄰居，大小為 max，並設定 priority\_queue 型態儲存數值較小值的優先權較高；

```
int main(){
```

宣告兩個整數型態變數 m, n 用於儲存「行與列的數量」；

輸出提示訊息「請輸入行與列的數量」；

使用者輸入兩個整數，將其儲存到 m, n；

宣告二維整數型態陣列 value，大小為 m\*n，用於儲存「對應位置的數值」；

宣告二維布林型態陣列 visit，大小為 m\*n，用於儲存「對應位置是否被造訪過」；

宣告四個整數型態變數 v, v\_up, v\_left, value\_min，各用於儲存「輸入的數值、目前輸入的位置上方位置的數值、目前輸入的位置左方位的數值、所有輸入的最小值」；

```
for i = 0 ~ i < m (for i++){
```

輸出提示訊息「輸入第 (i+1) 行的數值」；

```
for j = 0 ~ j < n (for j++){
```

將使用者輸入的數值存入 v；

// 此處會將整行輸入以空格分隔，每次存入 1 個數值

將 {i, j} 放入 place[v]，用於記錄數值 v 位置的行與列；

|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

```

    將 visit[i][j] 設為 false，記錄第 i 行第 j 列未造訪；
    將 value[i][j] 設為 v，記錄第 i 行第 j 列的數值為 v；
    如果 i 不是 0，則{
        將 v_up 設為 value[i-1][j]；
        將 v 存入 neighbor[v_up]，新增 v_up 的鄰居 v；
        將 v_up 存入 neighbor[v]，新增 v 的鄰居 v_up；
    }

    如果 j 不是 0，則{
        將 v_left 設為 value[i-1][j]；
        將 v 存入 neighbor[v_left]，新增 v_left 的鄰居 v；
        將 v_left 存入 neighbor[v]，新增 v 的鄰居 v_left；
    }

    如果 value_min 大於 v，則{
        將 value_min 設為 v；
    }
}
}

```

宣告兩個整數型態變數 next, path\_total，用於儲存「下一步數值、路徑數值總和」；

將 next 設為 value\_min，使得起始點為輸入最小值；  
將 path\_total 設為 0；

```

迴圈(當 next 不為 max){
    宣告整數型態變數 now，用於儲存「目前位置的數值」；
    將 now 設為 next；

    將 path_total 加上 now；

    將 visit[place[now][0].first][place[now][0].second]
    設為 true，用於將目前位置設為以造訪；

    將 next 設為 max；

```

|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

```
    迴圈(neighbor[now]不為空){
        將 next 設為 neighbor[now].top();
        neighbor[now].pop();

        如果 visit 不為真{
            //因長度問題備註，visit 為 visit[place[next][0].first][place[next][0].second]
            break;
        }
        否則{
            將 next 設為 max;
        }
    }
}

輸出訊息「所經過路徑數值總和：path_total」;

return 0;
}
```



|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

## 二. 完整程式碼：

```
#include<bits/stdc++.h>
using namespace std;
#define max 100001
// 定義數值 max 為 100001

vector< pair<int, int> > place[max];
// 宣告 vector 型態變數 place 儲存資料為兩個整數
// 此 vector 儲存數值 v 位置的行, 列在 place[v] 中

priority_queue< int, vector<int>, greater<int> > neighbor[max];
// 運用 priority_queue 型態較小值在前的特性，獲取周遭最小值
// 此處將優先值設為越小越前，因此與先前的實作不同

int main(){
    int m, n;
    // 宣告兩個整數型態變數 m, n 用於儲存「行與列的數量」
    printf("\n Input the row and column : ");
    // 輸出提示訊息，讓使用者輸入行與列
    scanf("%d%d", &m, &n);
    // 使用者輸入兩個整數，並將其儲存到 m, n

    int value[m][n];
    // 宣告一個二維整數型態陣列存放對應位置的值
    // 例如第 1 行的第 1 列儲存在[0][0]，並儲存該位置的數值
    // 此陣列是為了在後續透過位置快速找出鄰居關係

    bool visit[m][n];
    // 宣告一個二維布林型態陣列紀錄對應位置是否被造訪過
    // 例如第 1 行的第 1 列在[0][0]，並紀錄該位置是否有被造訪

    int v, v_up, v_left, value_min = max;
    // 宣告四個整數型態變數
    // v 儲存「輸入的數值」
    // v_up 儲存「目前輸入的位置上方位置的數值」
```

|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

```

// v_left 儲存「目前輸入的位置左方位置的數值」
// value_min 儲存「所有輸入的最小值」

for(int i = 0 ; i < m ; i++){
// 共有 m 行，因此外迴圈重複 m 次
    printf("\n Input value of row %d  :", (i + 1));
    // 輸出提示訊息，讓使用者輸入第 i+1 行的數值

    for(int j = 0 ; j < n ; j++){
// 共有 n 列，因此每行(內迴圈)重複 n 次
        scanf("%d", &v);
        // 輸入第 i 行第 j 列的數值
        // 實際的第 1 行及第 1 列值之 i, j 電腦中會-1

        place[v].push_back({i, j});
        // 將目前數值的位置 i, j 存入 place[目前輸入數值] 中
        visit[i][j] = false;
        // 將位置 i, j 的造訪設為 false 代表未造訪
        value[i][j] = v;
        // 紀錄位置 i, j 的數值為目前輸入數值

        if(i != 0){
// 當不是第 1 行時進入 if 內，詳見解題思維第 ii 點第 1、3 部分
            v_up = value[i-1][j];
            // 將目前位置上方的數值存入 v_up
            neighbor[v_up].push(v);
            // 新增 v 到 v_up 的鄰居佇列
            neighbor[v].push(v_up);
            // 新增 v_up 到 v 的鄰居佇列
        }

        if(j != 0){
// 當不是第 1 列時進入 if 內，詳見解題思維第 ii 點第 2、3 部分
            v_left = value[i][j-1];
            // 將目前位置左方的數值存入 v_left

```

|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

```

        neighbor[v_left].push(v);
        // 新增 v 到 v_left 的鄰居佇列
        neighbor[v].push(v_left);
        // 新增 v_left 到 v 的鄰居佇列
    }

    if(value_min > v){
        // 若目前輸入過的數值最小值小於目前輸入，進入 if 內
        value_min = v;
        // 輸入最小值設為 v
    }
}

int next = value_min, path_total = 0;
// 宣告兩個整數型態變數
// next 儲存「下一步走的數值」，初始值為輸入最小值
// path_total 儲存「路徑的數值總和」

while(next != max){
    // 當下一步不是 max 時，進入 while 內，同解題思維第 iii 點 step9
    int now = next;
    // 宣告整數型態變數 now 儲存「目前位置的數值」

    path_total += now;
    // 將 path_total 加上 now，紀錄路徑數值的總和
    visit[place[now][0].first][place[now][0].second] = true;
    // 將 now 的位置記為已造訪
    // place[now] 為陣列，因此需要特別標註[0]取出位置
    // place[now][0] 的 first 為 now 的行，second 為列

    next = max;
    // 將下一個位置的數值設為 max
    // neighbor[now] 為空時，若不設置 next 為 max，會造成無窮迴圈

```

|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

```

while(!neighbor[now].empty()){
    // 當 neighbor[now] 不為空時，進入 while 內
    next = (neighbor[now].top());
    // 將 neighbor[now] 最前的數值取出，也就是 now 周遭最小值
    neighbor[now].pop();
    // 因已取出最前的值，故需要用 pop 將該值移出
    // 若沒有移出，且最前的值已被造訪，則會造成無窮迴圈

    if(!visit[place[next][0].first][place[next][0].second]){
        // 若 next 的位置未被造訪，則進入 if 內
        break;
        // 跳出迴圈，讓 next 為最小值
    }
    else{
        next = max;
        // 重置 next 為 max
        // 若不重置，且當前 neighbor[now] 已空，下個位置會錯誤
    }
}

printf("\n the total value of path : %d", path_total);
// 輸出所走路徑的值總和


return 0;
}

```




|        |        |       |            |
|--------|--------|-------|------------|
| 資訊工程學系 | 資料結構應用 | 文件編號： | 期末考        |
|        |        | 發佈日期： | 2022/06/13 |

### 三. 輸入及輸出結果：

 D:\Program\C&CPP\finalexam\finalexam.exe

```
Input the row and column : 2 7
Input value of row 1 : 6 8 7 2 1 4 5
Input value of row 2 : 9 3 10 11 12 13 14
the total value of path : 36
```

 D:\Program\C&CPP\finalexam\finalexam.exe

```
Input the row and column : 4 5
Input value of row 1 : 24 7 14 20 30
Input value of row 2 : 11 6 4 21 29
Input value of row 3 : 2 8 1 35 40
Input value of row 4 : 3 9 5 12 15
the total value of path : 132
```

### 四. 心得與討論：

本學期最後一次的實作，在發揮的空間上非常大，並沒有特別限制需要使用的資料結構或演算法。因此我嘗試將期中後所學到的 STL 盡量套用，也就是上面所撰寫的程式，透過 vector 及 priority\_queue 的特點來快速達到需要的功能，在最後一次實作將本學期的實作經驗做一次重點歸納。

整體來說收穫非常豐富，不僅學到了 C/C++ 的語法，也從中了解到 C++ 在撰寫程式上其實也非常方便，改變了學期開始前認為 C++ 撰寫麻煩的固有觀念，也在此感謝龔老師本學期用心的指導，讓我收穫非常多。