

# 韌體實驗第13題

**Ting-Yu Lin, Adjunct Lecturer (林庭宇)**

**Office: AMOS Lab E517**

**Email: [tonylin0413@gmail.com](mailto:tonylin0413@gmail.com)**



**AMOS Lab.**  
Advanced Mixed-Operation System Lab.  
Dept. of Electrical and Computer Engineering,  
Tamkang University, Taiwan



**BIDAS Technology  
GrounTruth Workshop**  
Computer Vision  
Image Processing  
Edge Computing

# 第十三題

## 淡江大學電機工程學系 106 學年度韌體實驗題目

### 13. 【大理石在那兒？/Where is the Marble?】

現有  $N$  個大理石，每個大理石上寫了一個非負整數。首先把各數從小到大排序，然後回答  $Q$  個問題，每個問題問是否有一個大理石寫著某個整數  $x$ ？如果是，還要回答哪個大理石上寫著  $x$ （如果有兩個以上時，回答排序後算來的第一個）。排序後的大理石從左到右編號為  $1 \sim N$ 。

輸入（註 1）：

數字個數  $N$  4 1 查詢次數  $Q$

2

3

5

1

5

數列

查詢的數字

數字個數  $N$  5 2 查詢次數  $Q$

1

3

3

3

1

2

3

0 0

數列

查詢的數字

結束

輸出（註 2）：

CASE# 1:

5 found at 4

CASE# 2:

2 not found

3 found at 3



# 第十三題

輸入 (註 1):

數字個數N 4 1 查詢次數Q

2  
3  
5  
1

數列

5

查詢的數字

數字個數N 5 2 查詢次數Q

1  
3  
3  
3  
1

數列

2

查詢的數字

3

0 0 結束

輸出 (註 2):

CASE# 1:

5 found at 4

CASE# 2:

2 not found

3 found at 3

2	3	5	1
---	---	---	---



排序(小→大)

1	2	3	5
0	1	2	3



查詢(5)

5 found at 4 (矩陣index為3，但題意要從1算起)

淡江大學電機工程學系 106 學年度韌體實驗題目

13. 【大理石在那兒? /Where is the Marble?】

現有  $N$  個大理石，每個大理石上寫了一個非負整數。首先把各數從小到大排序，然後回答  $Q$  個問題，每個問題問是否有一個大理石寫著某個整數  $x$ ？如果是，還要回答哪個大理石上寫著  $x$ （如果有兩個以上時，回答排序後算來的第一個）。排序後的大理石從左到右編號為  $1 \sim N$ 。

# 第十三題

輸入 (註 1):

數字個數N 4 1 查詢次數Q

2  
3  
5  
1

數列

5 查詢的數字

數字個數N 5 2 查詢次數Q

1  
3  
3  
3  
1

數列

2 查詢的數字

3

0 0 結束

輸出 (註 2):

CASE# 1:

5 found at 4

CASE# 2:

2 not found

3 found at 3

1	3	3	3	1
---	---	---	---	---



排序(小→大)

1	1	3	3	3
0	1	2	3	4



查詢(2)(3)

2 not found

3 found at 3 (矩陣index為2，但題意要從1算起)

淡江大學電機工程學系 106 學年度韌體實驗題目

13. 【大理石在那兒？/Where is the Marble?】

現有  $N$  個大理石，每個大理石上寫了一個非負整數。首先把各數從小到大排序，然後回答  $Q$  個問題，每個問題問是否有一個大理石寫著某個整數  $x$ ？如果是，還要回答哪個大理石上寫著  $x$ （如果有兩個以上時，回答排序後算來的第一個）。排序後的大理石從左到右編號為  $1 \sim N$ 。

# 第十三題

## Member Functions (1/2)

- `void vector::push_back(const <T> &val);`
  - Add a new element at the end of the vector, after its current last element
  - The content of *val* is copied to the new element
  - The size of the vector is increased by one
- `void vector :: push_back (const value<type>& val)`
  - (1) 新增一個新的元素在 vector 尾端。
  - (2) 新增的元素為 val。
  - (3) vector 的大小為自動增加 1。

- 本題使用 vector 容器解題

- vector

<https://www.cplusplus.com/reference/vector/vector/>

- vector :: push\_back (val)

[https://www.cplusplus.com/reference/vector/vector/push\\_back/](https://www.cplusplus.com/reference/vector/vector/push_back/)

- vector :: begin()

<https://www.cplusplus.com/reference/vector/vector/begin/>

- vector :: end()

<https://www.cplusplus.com/reference/vector/vector/end/>

- lower\_bound()

[https://www.cplusplus.com/reference/algorithm/lower\\_bound/](https://www.cplusplus.com/reference/algorithm/lower_bound/)

- upper\_bound()

[https://www.cplusplus.com/reference/algorithm/upper\\_bound/](https://www.cplusplus.com/reference/algorithm/upper_bound/)

# 第十三題

## Member Functions (2/2)

- $\langle T \rangle$  \*vector::begin()
  - Return a pointer to the first element in the vector
- $\langle T \rangle$  \*vector::end()
  - Return a pointer to the *past-the-end* element in the vector, not be dereferenced
- vector :: **begin** ()
 

回傳 vector 中 **第一個元素指標位置**。
- vector :: **end** ()
 

回傳 vector 中 **最後一個元素指標位置**。

- 本題使用 vector 容器解題

- vector

<https://www.cplusplus.com/reference/vector/vector/>

- vector :: push\_back (val)

[https://www.cplusplus.com/reference/vector/vector/push\\_back/](https://www.cplusplus.com/reference/vector/vector/push_back/)

- vector :: begin()

<https://www.cplusplus.com/reference/vector/vector/begin/>

- vector :: end()

<https://www.cplusplus.com/reference/vector/vector/end/>

- lower\_bound()

[https://www.cplusplus.com/reference/algorithm/lower\\_bound/](https://www.cplusplus.com/reference/algorithm/lower_bound/)

- upper\_bound()

[https://www.cplusplus.com/reference/algorithm/upper\\_bound/](https://www.cplusplus.com/reference/algorithm/upper_bound/)

# 第十三題

## Lib. Functions

- `<T> * lower_bound(pointerA to the first element, pointerB to the last element, const <T> &val);`
    - Defined in `<algorithm>`
    - Return a pointer to the first element in the range `[pointerA, pointerB)` which **not less** than `val`
    - Cf. `<T> *upper_bound(...)`
      - Return a pointer to the first element which **greater** than `val`
  - `lower_bound (pointer A, pointer B, val)`
- 回傳 vector 中範圍在 **pointerA** 和 **pointerB** 之間的不小於(大於或等於) `val` 值的第一個元素**指標位置**。

### ● 本題使用 vector 容器解題

#### ➤ vector

<https://www.cplusplus.com/reference/vector/vector/>

#### ➤ vector :: push\_back (val)

[https://www.cplusplus.com/reference/vector/vector/push\\_back/](https://www.cplusplus.com/reference/vector/vector/push_back/)

#### ➤ vector :: begin()

<https://www.cplusplus.com/reference/vector/vector/begin/>

#### ➤ vector :: end()

<https://www.cplusplus.com/reference/vector/vector/end/>

#### ➤ lower\_bound()

[https://www.cplusplus.com/reference/algorithm/lower\\_bound/](https://www.cplusplus.com/reference/algorithm/lower_bound/)

#### ➤ upper\_bound()

[https://www.cplusplus.com/reference/algorithm/upper\\_bound/](https://www.cplusplus.com/reference/algorithm/upper_bound/)

# 第十三題

## Lib. Functions

- $\langle T \rangle$  \* lower\_bound(*pointerA* to the first element, *pointerB* to the last element, `const  $\langle T \rangle$  &val`);
    - Defined in `<algorithm>`
    - Return a pointer to the first element in the range [*pointerA*, *pointerB*) which **not less** than *val*
    - Cf.  $\langle T \rangle$  \*upper\_bound(...)
    - Return a pointer to the first element which **greater** than *val*
  - upper\_bound (pointer A, pointer B, val)
- 回傳 vector 中範圍在 **pointerA** 和 **pointerB** 之間的大於 val 值的第一個元素**指標位置**。

- 本題使用 vector 容器解題

- vector

<https://www.cplusplus.com/reference/vector/vector/>

- vector :: push\_back (val)

[https://www.cplusplus.com/reference/vector/vector/push\\_back/](https://www.cplusplus.com/reference/vector/vector/push_back/)

- vector :: begin()

<https://www.cplusplus.com/reference/vector/vector/begin/>

- vector :: end()

<https://www.cplusplus.com/reference/vector/vector/end/>

- lower\_bound()

[https://www.cplusplus.com/reference/algorithm/lower\\_bound/](https://www.cplusplus.com/reference/algorithm/lower_bound/)

- upper\_bound()

[https://www.cplusplus.com/reference/algorithm/upper\\_bound/](https://www.cplusplus.com/reference/algorithm/upper_bound/)



# 第十三題

➤ lower\_bound (pointer A, pointer B, val)

回傳 vector 中範圍在 **pointerA** 和 **pointerB** 之間的不小於(大於或等於) val 值的第一個元素**指標位置**。

➤ upper\_bound (pointer A, pointer B, val)

回傳 vector 中範圍在 **pointerA** 和 **pointerB** 之間的大於 val 值的第一個元素**指標位置**。

[https://www.cplusplus.com/reference/algorithm/lower\\_bound/](https://www.cplusplus.com/reference/algorithm/lower_bound/)

## Example

```
1 // lower_bound/upper_bound example
2 #include <iostream>      // std::cout
3 #include <algorithm>     // std::lower_bound, std::upper_bound, std::sort
4 #include <vector>        // std::vector
5
6 int main () {
7     int myints[] = {10,20,30,30,20,10,10,20};
8     std::vector<int> v(myints,myints+8);           // 10 20 30 30 20 10 10 20
9
10    std::sort (v.begin(), v.end());                 // 10 10 10 20 20 20 30 30
11
12    std::vector<int>::iterator low,up;
13    low=std::lower_bound (v.begin(), v.end(), 20); //      ^
14    up= std::upper_bound (v.begin(), v.end(), 20); //      ^
15
16    std::cout << "lower_bound at position " << (low- v.begin()) << '\n';
17    std::cout << "upper_bound at position " << (up - v.begin()) << '\n';
18
19    return 0;
20 }
```

Edit  
&  
Run

Output:

```
lower_bound at position 3
upper_bound at position 6
```

# 第十三題

```

1  #define _CRT_SECURE_NO_WARNINGS
2  //[大理石在那兒?/Where is the Marble?](1/2)
3  #define IN "P13IN.txt"
4  #define OUT "P13OUT.txt"
5  //*****
6  #include <iostream>
7  #include <ctime>
8  using namespace std;
9  void redir(void);
10
11 //*****
12 /* Work Space*/
13 #include <vector>
14 #include <algorithm>
15 //*****
16
17 int main(void) {
18     redir(); //redirection
19

```

```

42
43     freopen("CON", "r", stdin); //取消重新導向
44     freopen("CON", "w", stdout);
45     printf("Time used = %.2f\n", (double)clock() / CLK_TCK);
46     system("pause");
47     return 0; //the end...
48 }
49
50 void redir(void) {
51     freopen(IN, "r", stdin);
52     freopen(OUT, "w", stdout);
53 }
54

```

## 需要修改或自行撰寫的部分

- 初行加入crt secure no warnings。
- 記得修改輸入和輸出txt檔案名稱。
- #define為C++直接定義，變數型態會依照定義自動判定。
- #include <vector> 內建函式。
- ✓ vector.push\_back()、vector.begin()、vector.end()。
- #include <algorithm> 內建函式。
- ✓ 氣泡排序法sort()、下限搜尋lower\_bound()、上限搜尋upper\_bound()。

# 第十三題

```

20 //*****
21 /* Work Space*/
22     int kase = 0, N, Q, i, Z, p;
23
24     while (scanf("%d %d", &N, &Q) == 2 && N) {
25         vector<int> a; //block scope
26         printf("CASE# %d:\n", ++kase);
27         for (i = 0; i < N; i++) {
28             scanf("%d", &Z);
29             a.push_back(Z);
30         }
31         sort(a.begin(), a.end());
32         while (Q--) {
33             scanf("%d", &Z);
34             //lower_bound() 尋找大於或等於Z的第一個位置
35             p = int(lower_bound(a.begin(), a.end(), Z) - a.begin());
36             if (p < a.size() && a[p] == Z) {
37                 printf("%d found at %d\n", Z, p + 1);
38             }
39             else {
40                 printf("%d not found\n", Z);
41             }
42         }
43     }
44 //*****

```

## 需要修改或自行撰寫的部分

➤ 解題思考：

- (1) 變數宣告、讀取輸入資訊
- (2) 排序
- (3) 搜尋
- (4) 顯示輸出

# 第十三題

```

20 //*****
21 /* Work Space*/
22 int kase = 0, N, Q, i, Z, p;
23
24 while (scanf("%d %d", &N, &Q) == 2 && N) {
25     vector<int> a; //block scope
26     printf("CASE# %d:\n", ++kase);
27     for (i = 0; i < N; i++) {
28         scanf("%d", &Z);
29         a.push_back(Z);
30     }
31     sort(a.begin(), a.end());
32     while (Q--) {
33         scanf("%d", &Z);
34         //lower_bound() 尋找大於或等於Z的第一個位置
35         p = int(lower_bound(a.begin(), a.end(), Z) - a.begin());
36         if (p < a.size() && a[p] == Z) {
37             printf("%d found at %d\n", Z, p + 1);
38         }
39         else {
40             printf("%d not found\n", Z);
41         }
42     }
43 }
44 //*****

```

## 需要修改或自行撰寫的部分

➤ 解題思考：

(1) 變數宣告、讀取輸入資訊

(2) 排序

(3) 搜尋

(4) 顯示輸出

● Line 22:

- ✓ 宣告 **整數** 變數 kase：第幾個 case。
- ✓ 宣告 **整數** 變數 N：大理石個數。
- ✓ 宣告 **整數** 變數 Q：查詢的次數。
- ✓ 宣告 **整數** 變數 Z：查詢的數字。
- ✓ 宣告 **整數** 變數 p：查詢的數字的 index。

● Line 24-30:

- ✓ scanf N和Q，如果兩個變數皆成功輸入則回傳2。
- ✓ 如果N為0，經由&&邏輯運算子，while(0)不進入迴圈結束。

淡江大學電機工程學系 106 學年度韌體實驗題目

13. 【大理石在那兒？/Where is the Marble?】

現有 N 個大理石，每個大理石上寫了一個非負整數。首先把各數從小到大排序，然後回答 Q 個問題，每個問題問是否有一個大理石寫著某個整數 x？如果是，還要回答哪個大理石上寫著 x（如果有兩個以上時，回答排序後算來的第一個）。排序後的大理石從左到右編號為 1~N。



# 第十三題

```

20 //*****
21 /* Work Space*/
22 int kase = 0, N, Q, i, Z, p;
23
24 while (scanf("%d %d", &N, &Q) == 2 && N) {
25     vector<int> a; //block scope
26     printf("CASE# %d:\n", ++kase);
27     for (i = 0; i < N; i++) {
28         scanf("%d", &Z);
29         a.push_back(Z);
30     }
31     sort(a.begin(), a.end());
32     while (Q--) {
33         scanf("%d", &Z);
34         //lower_bound() 尋找大於或等於Z的第一個位置
35         p = int(lower_bound(a.begin(), a.end(), Z) - a.begin());
36         if (p < a.size() && a[p] == Z) {
37             printf("%d found at %d\n", Z, p + 1);
38         }
39         else {
40             printf("%d not found\n", Z);
41         }
42     }
43 }
44 //*****

```

## 需要修改或自行撰寫的部分

➤ 解題思考：

(1) 變數宣告、讀取輸入資訊

(2) 排序

(3) 搜尋

(4) 顯示輸出

● Line 25:

✓ vector<int> a：宣告名稱為a的整數型態vector。

● Line 26:

✓ 列印CASE，++kase為先加一再執行。

● Line 27-30:

✓ for迴圈依序scanf大理石數字至變數Z，再使用push\_back()存入vector容器中。

淡江大學電機工程學系 106 學年度韌體實驗題目

13. 【大理石在那兒？/Where is the Marble?】

現有  $N$  個大理石，每個大理石上寫了一個非負整數。首先把各數從小到大排序，然後回答  $Q$  個問題，每個問題問是否有一個大理石寫著某個整數  $x$ ？如果是，還要回答哪個大理石上寫著  $x$ （如果有兩個以上時，回答排序後算來的第一個）。排序後的大理石從左到右編號為  $1 \sim N$ 。

# 第十三題

```

20 //*****
21 /* Work Space*/
22 int kase = 0, N, Q, i, Z, p;
23
24 while (scanf("%d %d", &N, &Q) == 2 && N) {
25     vector<int> a; //block scope
26     printf("CASE# %d:\n", ++kase);
27     for (i = 0; i < N; i++) {
28         scanf("%d", &Z);
29         a.push_back(Z);
30     }
31     sort(a.begin(), a.end());
32     while (Q--) {
33         scanf("%d", &Z);
34         //lower_bound() 尋找大於或等於Z的第一個位置
35         p = int(lower_bound(a.begin(), a.end(), Z) - a.begin());
36         if (p < a.size() && a[p] == Z) {
37             printf("%d found at %d\n", Z, p + 1);
38         }
39         else {
40             printf("%d not found\n", Z);
41         }
42     }
43 }
44 //*****

```

## 需要修改或自行撰寫的部分

➤ 解題思考：

(1) 變數宣告、讀取輸入資訊

(2) 排序

(3) 搜尋

(4) 顯示輸出

● Line 31:

✓ 氣泡排序法sort，參數(vector中第一個元素指標位置，vector中最後一個元素指標位置)。

● Line 32:

✓ while(Q--) 執行每次搜尋。

● Line 35:

✓ lower\_bound() 尋找 a.begin() 和 a.end()之間不小於(大於或等於) Z的**第一個元素指標位置**。

✓ 將找到的位置與起始位置相減，並轉換成整數型態存入變數 p。

## 淡江大學電機工程學系 106 學年度韌體實驗題目

### 13. 【大理石在那兒？/Where is the Marble?】

現有  $N$  個大理石，每個大理石上寫了一個非負整數。首先把各數從小到大排序，然後回答  $Q$  個問題，每個問題問是否有一個大理石寫著某個整數  $x$ ？如果是，還要回答哪個大理石上寫著  $x$ （如果有兩個以上時，回答排序後算來的第一個）。排序後的大理石從左到右編號為  $1 \sim N$ 。

# 第十三題

```

20 //*****
21 /* Work Space*/
22 int kase = 0, N, Q, i, Z, p;
23
24 while (scanf("%d %d", &N, &Q) == 2 && N) {
25     vector<int> a; //block scope
26     printf("CASE# %d:\n", ++kase);
27     for (i = 0; i < N; i++) {
28         scanf("%d", &Z);
29         a.push_back(Z);
30     }
31     sort(a.begin(), a.end());
32     while (Q--) {
33         scanf("%d", &Z);
34         //lower_bound() 尋找大於或等於Z的第一個位置
35         p = int(lower_bound(a.begin(), a.end(), Z) - a.begin());
36         if (p < a.size() && a[p] == Z) {
37             printf("%d found at %d\n", Z, p + 1);
38         }
39         else {
40             printf("%d not found\n", Z);
41         }
42     }
43 }
44 //*****

```

## 需要修改或自行撰寫的部分

➤ 解題思考：

- (1) 變數宣告、讀取輸入資訊
- (2) 排序
- (3) 搜尋
- (4) 顯示輸出

● Line 36-42:

- ✓ 如果搜尋到的指標位置小於vector大小且(&&)搜尋的數字為正確，則列印找到，p+1為依題意。
- ✓ 否則列印沒有找到。

## 淡江大學電機工程學系 106 學年度韌體實驗題目

### 13. 【大理石在那兒？/Where is the Marble?】

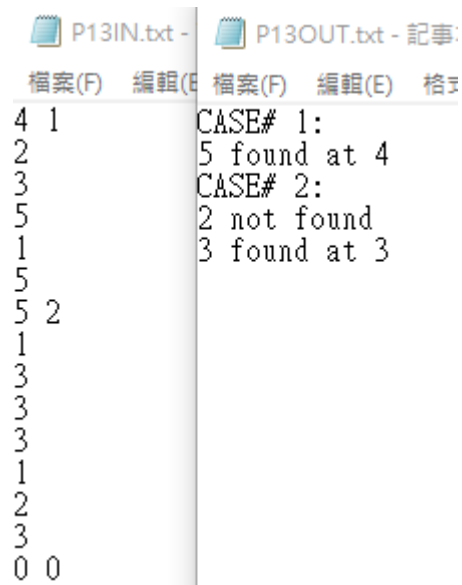
現有  $N$  個大理石，每個大理石上寫了一個非負整數。首先把各數從小到大排序，然後回答  $Q$  個問題，每個問題問是否有一個大理石寫著某個整數  $x$ ？如果是，還要回答哪個大理石上寫著  $x$ （如果有兩個以上時，回答排序後算來的第一個）。排序後的大理石從左到右編號為  $1 \sim N$ 。

題目  
解析

程式  
詳解

輸出  
結果

## 第十三題



The screenshot shows two text files side-by-side. The left file, 'P13IN.txt', contains the following input data:

```
4 1
2
3
5
1
5
5 2
1
3
3
3
1
2
3
0 0
```

The right file, 'P13OUT.txt', contains the following output data:

```
CASE# 1:
5 found at 4
CASE# 2:
2 not found
3 found at 3
```





# Q&A

