

# Lesson 3. 排序與搜尋 I

Sorting and Searching I

# 排序與搜尋 Sorting and Searching

排序 ( Sorting ) : 將某些東西以有規則的方式排列

搜尋 ( Searching ) : 利用演算法有效率的查找我們想要找的東西

# std::sort

C++ STL 之下內建的排序工具，使用前必須先導入標頭檔 `algorithm`

用法：

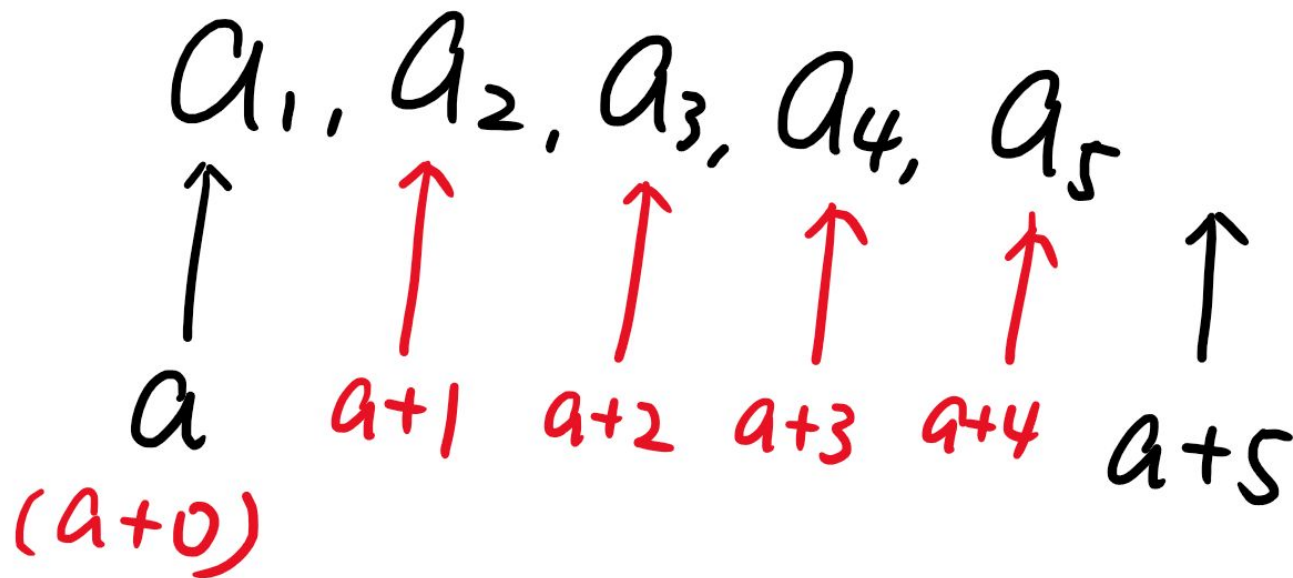
```
sort( array_name + L , array_name + R ); // sorting array's [ L , R )  
elements
```

sort 裡放的第一個參數與第二個參數都為 迭代器 的型態

可以把迭代器想像成一種指標，第四單元的時候會介紹到

std::sort 的排序預設為由小到大，預設複雜度為  $O(N * \log N)$

## 一般陣列的迭代器



# std::sort

```
1 #include <iostream>
2
3 #include <algorithm>
4
5 using namespace std;
6
7 int main(void)
8 {
9     int a[5];
10
11     for(int i=0;i<5;i++) cin >> a[i]; // 4 4 3 2 1
12
13     sort(a,a+5); // sort(a+0,a+5); [0,5)
14
15     for(int i=0;i<5;i++)
16     {
17         cout << a[i] << " "; // 1 2 3 4 4
18     }
19
20     cout << "\n";
21
22     return 0;
23 }
```

# 字串的比較

字串在比較上預設都是以 **字典序** 來做為比較的基準

字典序會以 ASCII 的先後順序而定

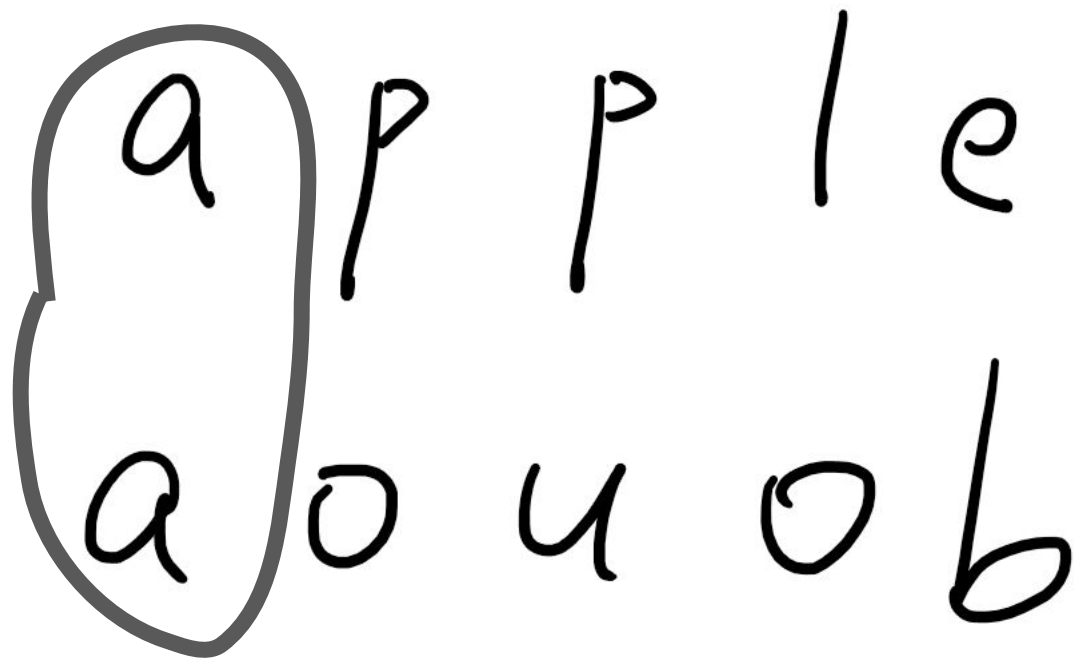
兩個字串在比較時會由左到右依序比較

## 字串比較的例子

a p p l e

a o u o b

由左往右比較



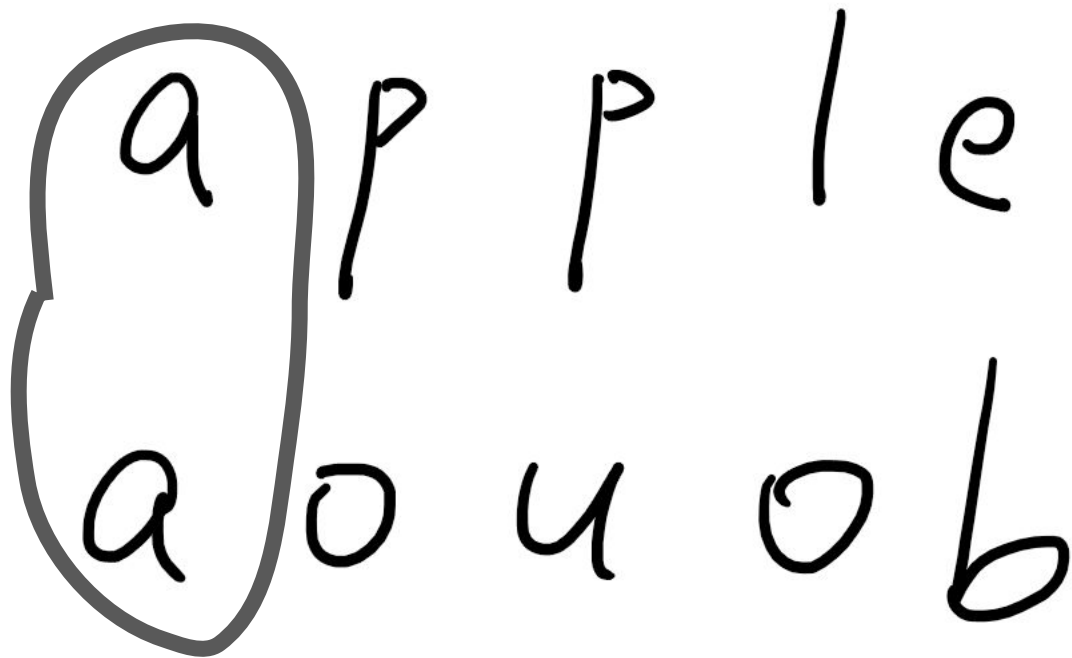
apple  
apob

The image shows a handwritten comparison of the words 'apple' and 'apob'. The first character 'a' in both words is circled with a thick grey line, indicating the starting point of a left-to-right comparison. The words are written in a casual, cursive style.



由左往右比較

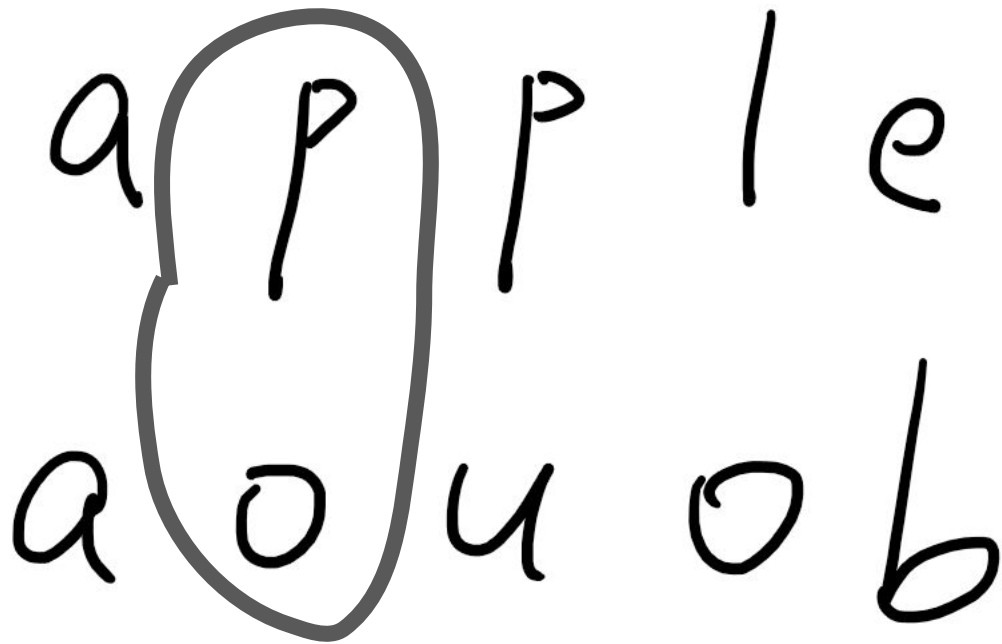
ascii 大小一樣, 繼續比較



Handwritten text showing a comparison between two strings: 'apple' and 'apob'. The first character 'a' in both strings is circled, indicating the current step in a left-to-right comparison process.

| String 1 | String 2 |
|----------|----------|
| a        | a        |
| p        | o        |
| p        | u        |
| l        | o        |
| e        | b        |

繼續比較，一旦分出大小則停止比較

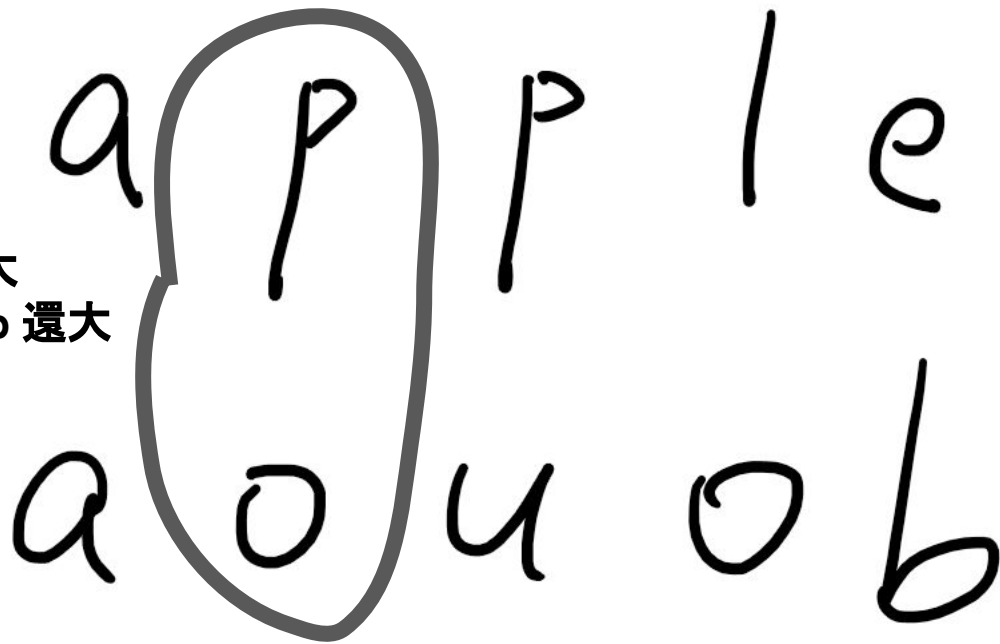


Handwritten text showing a comparison between two strings: 'apple' and 'apob'. The first two characters, 'a' and 'p', are circled together, indicating the point where the strings are compared and found to be equal, leading to the next step in the comparison process.

|   |   |   |   |   |
|---|---|---|---|---|
| a | p | p | l | e |
| a | o | u | o | b |

繼續比較，一旦分出大小則停止比較

'p' 比 'o' 的 ascii 還大  
因此 apple 比 aouob 還大



Handwritten text showing the comparison of 'apple' and 'aouob'. The word 'apple' is written above 'aouob'. A large circle is drawn around the second characters, 'p' in 'apple' and 'o' in 'aouob', indicating the point of comparison.

那麼如果兩個字串一長一短而且前半部都一樣呢

apple  
appleouob

由於第一個字串比到最後沒東西了，所以比第二個字串還小

apple  
appleouob

## 小細節：字串陣列的排序

如果排序的東西為字串陣列，那麼複雜度必須多考慮字串的長度（因為要比較字元）

有時候忽略這個常常會誤以為自己的程式效率很好！務必特別注意

```
1 #include <iostream>
2
3 #include <algorithm>
4
5 using namespace std;
6
7 int main(void)
8 {
9     string a[5] = {"abc", "bbc", "TNGS", "HHSH", "ouob"};
10
11     sort(a, a+5); // sort(a+0, a+5); [0, 5)
12
13     for(int i=0; i<5; i++)
14     {
15         cout << a[i] << " ";
16     }
17
18     cout << "\n";
19
20     return 0;
21 }
```

sort 例題：APCS 2016 3 月 第一題 成績指標

一般排序與變數紀錄答案的應用

題外話：~~這題應該是歷屆 APCS 考試最難的第一題~~

## 輸入處理

```
12     int n , a[25];  
13  
14     cin >> n;  
15  
16     for(int i=0;i<n;i++) cin >> a[i];  
17  
18     sort(a,a+n); // 將 a 序列排序
```



## 利用變數紀錄答案

```
18 // 將 a 列為初值  
19  
20 int mx_low60 = -1 , mn_high60 = 101; // mx_low60 為最高不及格分數 ; 另則為最低及格分數  
21
```

## 利用大家的分數更新答案

```
22  for(int i=0;i<n;i++)
23  {
24      if( a[i] ≥ 60 ) mn_high60 = min(mn_high60,a[i]); // 有人及格，對最低及格分數比較
25
26      else mx_low60 = max(mx_low60,a[i]); // 有人不及格，對不及格分數比較
27  }
```

## 輸出答案

```
29     for(int i=0;i<n;i++) cout << a[i] << " ";
30
31     cout << "\n";
32
33     if( mx_low60 == -1 ) cout << "best case\n"; // mx_low60 都沒變表示大家都及格
34
35     else cout << mx_low60 << "\n";
36
37     if( mn_high60 == 101 ) cout << "worst case\n"; // mn_high60 都沒變表示大家都不及格
38
39     else cout << mn_high60 << "\n";
```

練習題：TNGS IRC Final Exam pE. zhu 的竹子序列

我當初出這一題的時候有特別設計成不會排序也可以做  
不過作法特別暴力

現在你會排序了，使用排序做看看這一題吧！

## 輸入與前置

```
12     int a[105] , index = 10; // 下一個元素的索引值為 index
13
14     for(int i=0;i<10;i++) cin >> a[i];
```

## 操作一

```
28         int num;
29
30         cin >> num;
31
32         a[index] = num; // num 要插入 index 這個索引值
33
34         index++; // 下一個元素要插入在 index + 1 的位置
```

## 操作二

```
38         sort(a,a+index); // 排序
39
40         int k;
41
42         cin >> k;
43
44         cout << a[index-k] << "\n"; // 輸出第 k 大元素（別忘記排序預設是由小到大）
```

sort 只能由小到大排序嗎

當然不是！如果你想要自定義規則的話就必須要自己寫 compare  
compare function 的型態為 布林 ( bool )，且包含兩個參數

參數的型態依照排序的序列型態而定，序列是什麼型態參數型態就是什麼

至於回傳的值當然只有 true 或 false



# 什麼時候回傳 true 什麼時候回傳 false

compare 回傳的規則定義為：

第一個參數是否有比第二個參數小

因此如果當兩個參數相同大小時，切記！！！！一定要回傳 false

不然你將會得到 Runtime Error

一切就緒！！將 sort 函式加入第三個參數即可

第三個參數要放什麼取決於你的 compare 的函式名稱

名稱叫 apple 就放 apple, banana 就放 banana

```
sort(a, a+5, cmp);
```

## compare 範例 ( 由大到小 )

```
1 #include <iostream>
2
3 #include <algorithm>
4
5 using namespace std;
6
7 bool cmp(int a,int b)
8 {
9     return a > b;
10 }
11 int main(void)
12 {
13     int a[5];
14
15     for(int i=0;i<5;i++) cin >> a[i];
16
17     sort(a,a+5,cmp);
18
19     for(int i=0;i<5;i++) cout << a[i] << " ";
20
21     cout << "\n";
22
23     return 0;
24 }
```

例題：Ten Point Round #4 (Div. 2) pB 三個數字比大小

compare 就可以水過去的題目

備註：~~這一題也可以不要用排序，只是你要寫 11 個 if~~

## 讀入資料後排序

```
19 int main(void)
20 {
21     cin >> a[0] >> a[1] >> a[2];
22
23     sort(a, a+3, cmp);
24
25     cout << a[1] << "\n";
26
27     return 0;
28 }
```

## compare 函式

```
11 bool cmp(int a,int b)
12 {
13     if( a % 2 == 0 && b % 2 == 1 ) return true; // 如果 a 是偶 b 是奇 ; 則 a < b
14
15     if( a % 2 == 1 && b % 2 == 0 ) return false; // 如果 a 是奇 b 是偶 ; 則 a > b
16
17     return a < b; // a == b return false → 奇偶性相同，利用原本數字的大小進行比較
18 }
```

經典問題：

Moon Festival Ten Point Round 2021 (Div. 1 + Div. 2) pH. 柚子編號排列問題

很不一樣的角度思考 compare，建議大家都一定要學會這一題

字典序的特性也可以直接拿來比較數字大小

如果將整數轉成字串，那麼字典序越大的整數表示這個整數越大

這題必須要先知道這一個特性 ouob

因為字串字典序是由左到右比較，當其中一位比較小的時候

這個整數就可以比較出大小了



很多人的直覺（包括一開始看到這題的我）

：那就把字串全部排序然後由大到小輸出就好了啊

Judge : Wrong Answer

反例：

2、221 由大到小排序會是 221、2 組合出來會是 2212

但這題 2221 的字典序是比 2212 還要大的

這題該怎麼做（先將整數轉成字串）

思考看看，我們現在有兩個字串  $a$  ,  $b$

這兩個字串可以組合出  $a + b$  與  $b + a$  兩種字串的可能

那麼我們可以很確定的事情是：

如果  $a + b$  的字串  $>$   $b + a$  的字串，那麼  $a$  一定放在  $b$  的左邊比較好

既然知道這樣了

那我們就利用排序，並自己寫一個 `compare` 來完成排序！

排序的規則就如剛剛我們推出來的：

如果  $a + b$  的字串  $>$   $b + a$  的字串，那麼  $a$  一定放在  $b$  的左邊比較好

# 實作

```
9 bool cmp(string a,string b)
10 {
11     return a + b > b + a;
12 }
```

```
17 signed main(void)
18 {
19     int n;
20
21     cin >> n;
22
23     for(int i=0;i<n;i++)
24     {
25         cin >> a[i];
26     }
27
28     sort(a.begin(),a.end(),cmp); // 排序
29
30     for(int i=0;i<n;i++)
31     {
32         cout << a[i]; // 依序輸出排序完的字串
33     }
34
35     cout << "\n";
36
37     return 0;
38 }
```