第十一週

王子銜, 陳毅軒, 吳尚龍

電機通訊程式設計

April 29, 2024

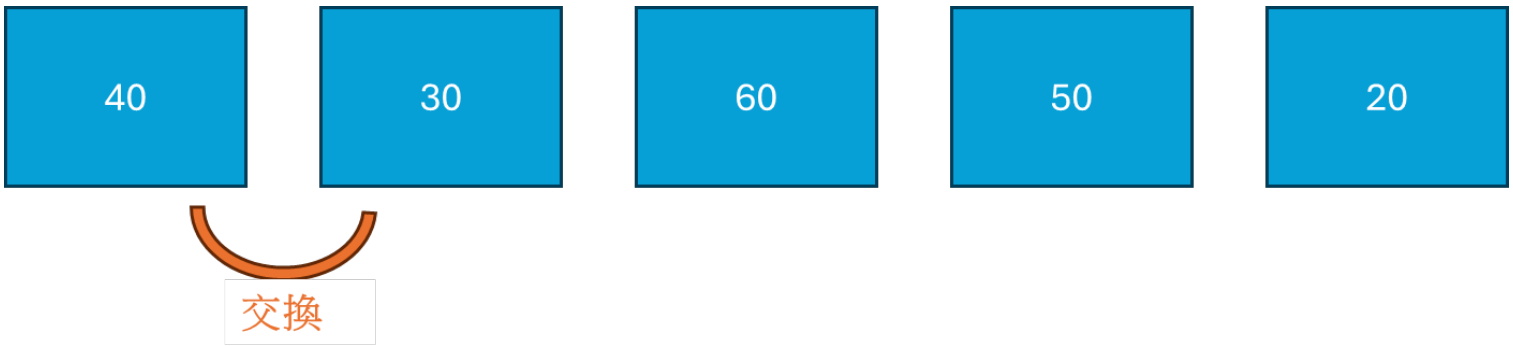
Outline

bubble sort

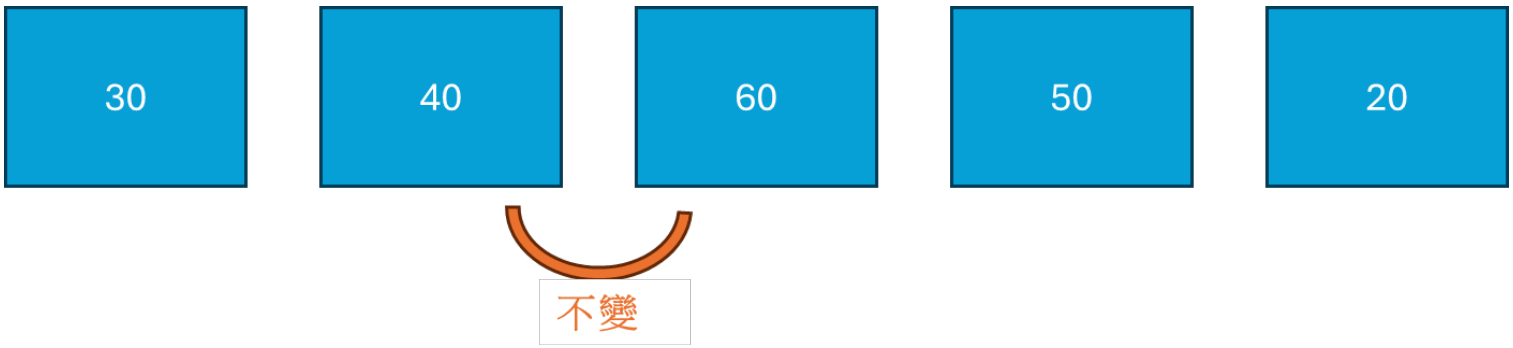
bubble sort

氣泡排序法 (Bubble Sort) 又稱交換排序法，原理是從第一筆資料開始，逐一比較相鄰兩筆資料，如果兩筆大小順序有誤則做交換，反之則不動，接者再進行下一筆資料比較，所有資料比較完第 1 回合後，可以確保最後一筆資料是正確的位置。
因爲一輪只能確認一個位置，所以要做 $n-1$ 輪。

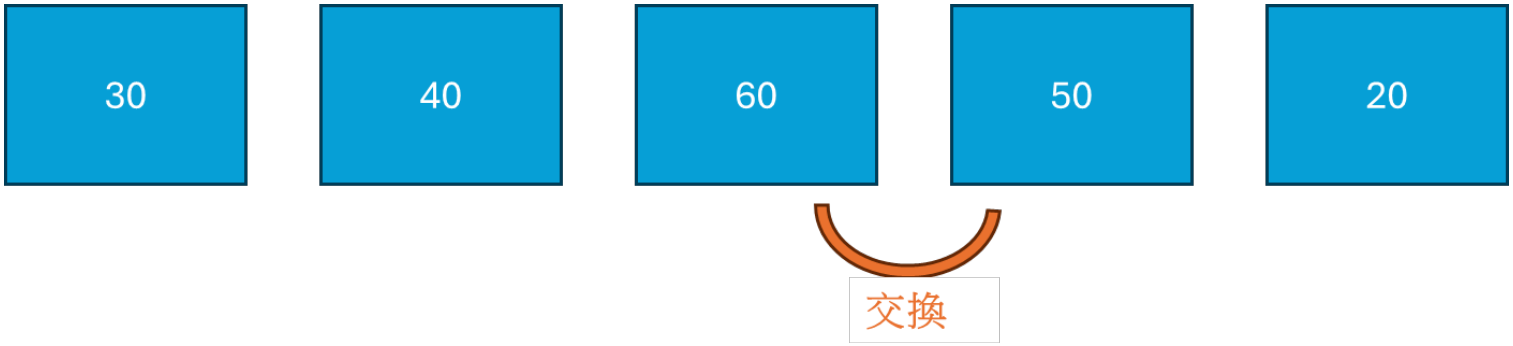
bubble sort



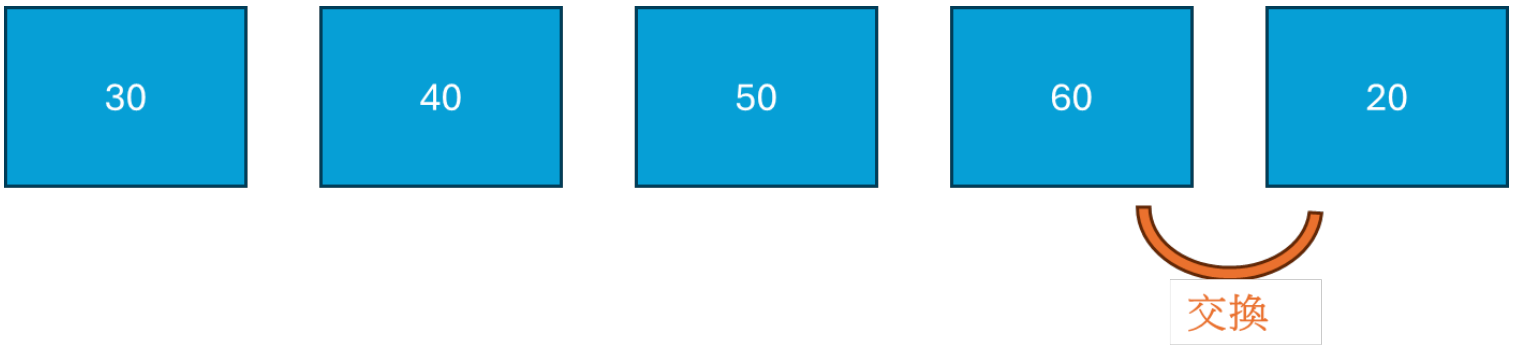
bubble sort



bubble sort



bubble sort



bubble sort

bubble sort

```
1  void bubble_sort(int arr[], int n) {
2      for (int i = n - 1; i > 0; i++) {   // i 是最後綠色在的位置
3          for (int j = 0; j < i - 1; j++) {
4              if (arr[j] > arr[j + 1]) {
5                  int temp = arr[j];
6                  arr[j] = arr[j + 1];
7                  arr[j + 1] = temp;
8              }
9          }
10     }
11 }
```

bubble sort

時間複雜度

比較次數 $(n-1)+(n-2)+(n-3)+...+1$
時間複雜度爲 $O(n^2)$

Insertion sort

Insertion sort

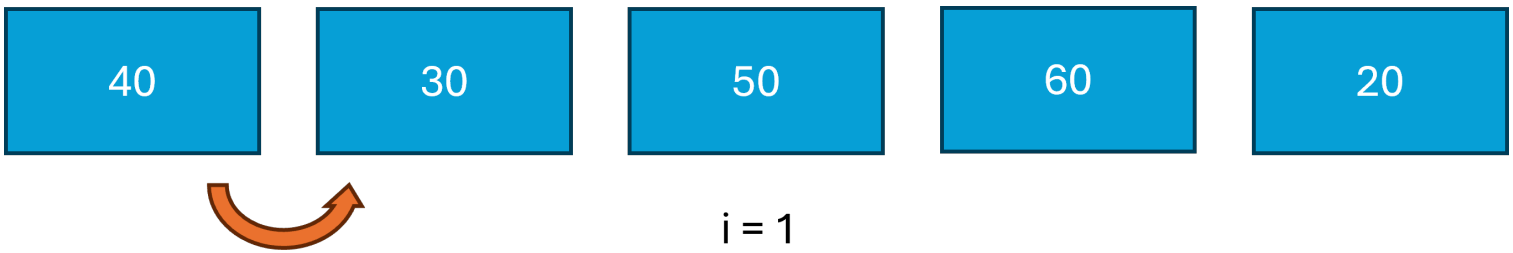
插入排序法 (Insertion Sort) 的原理是：將資料列假設分成已排序和未排序的兩部分，每次從未排序的資料中，挑選出一個元素，插入到已排序的資料中，直到所有的資料都已排序完成。

Insertion sort 的執行流程可以這樣描述：

1. 從資料列的第二個元素開始，逐一取出每一個元素，稱爲目標元素。
2. 將目標元素與已排序的資料列 (目標元素前的資料列) 中的元素逐一比較，直到找到一個比目標元素大的元素或搜尋完整個已排序的資料列。
3. 將目標元素插入到適當的位置。
4. 重複上述過程直到所有的元素都已排序完成。

因爲一輪只能確認一個目標元素放正確位置，所以要做 $n-1$ 輪。

Insertion sort



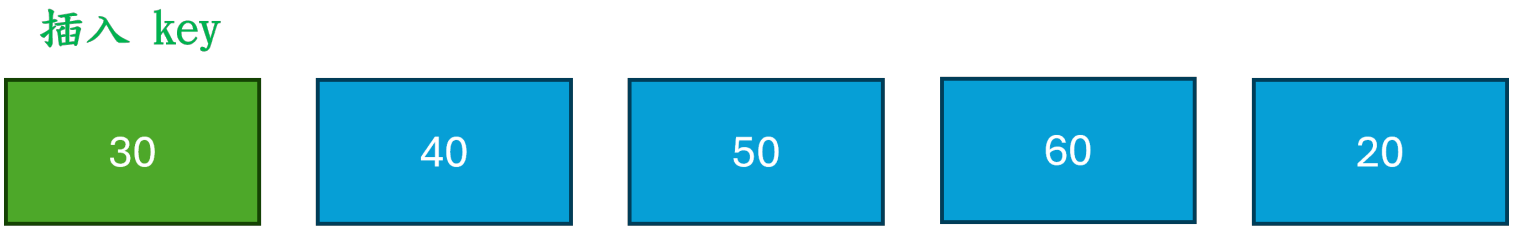
Insertion sort

i = 1

j = -1

key = 30

Insertion sort



i = 1

j = -1

key = 30

Insertion sort

i = 2

j = 1

key = 50

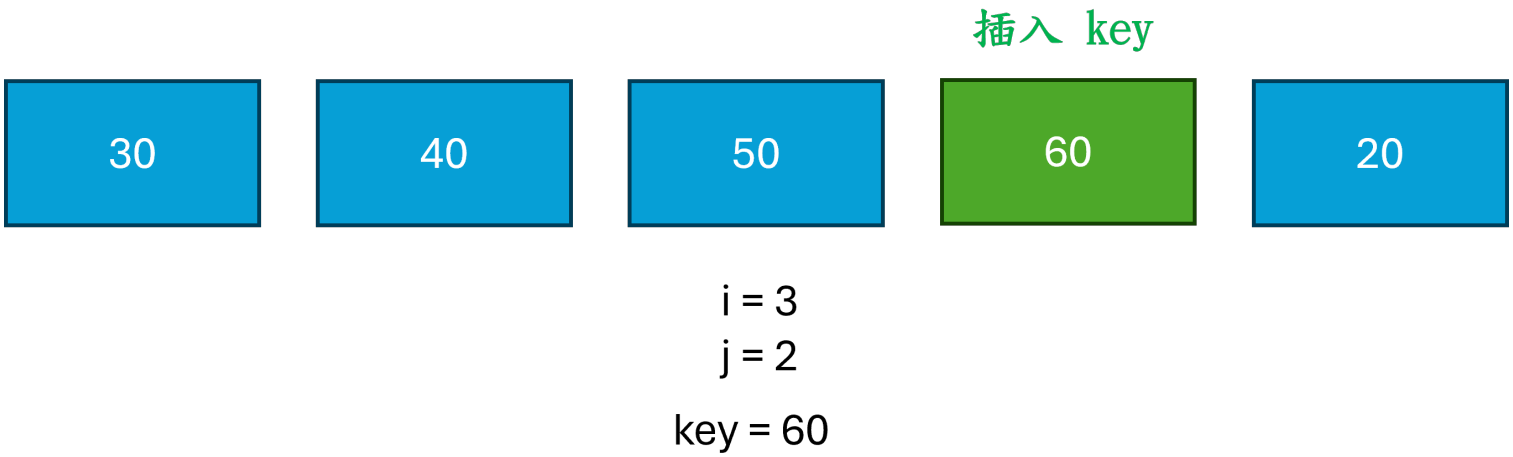
Insertion sort

i = 2

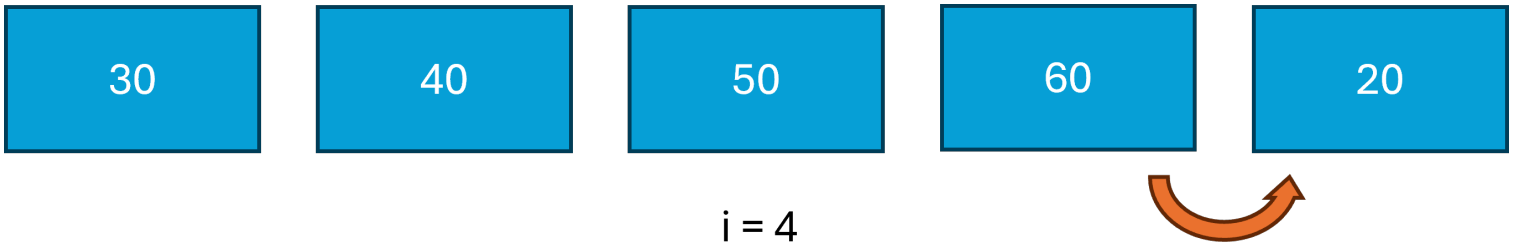
j = 1

key = 50

Insertion sort



Insertion sort

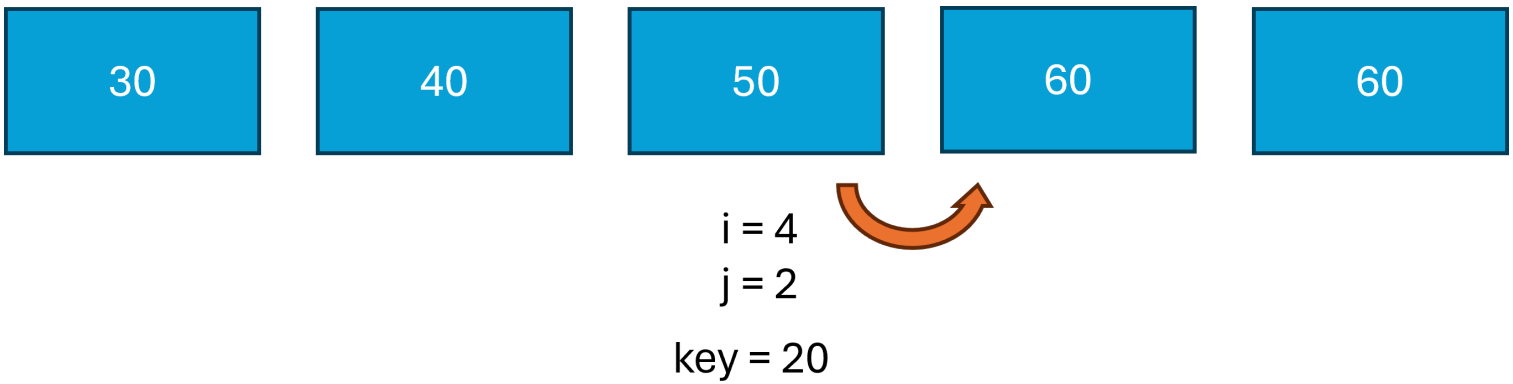


i = 4

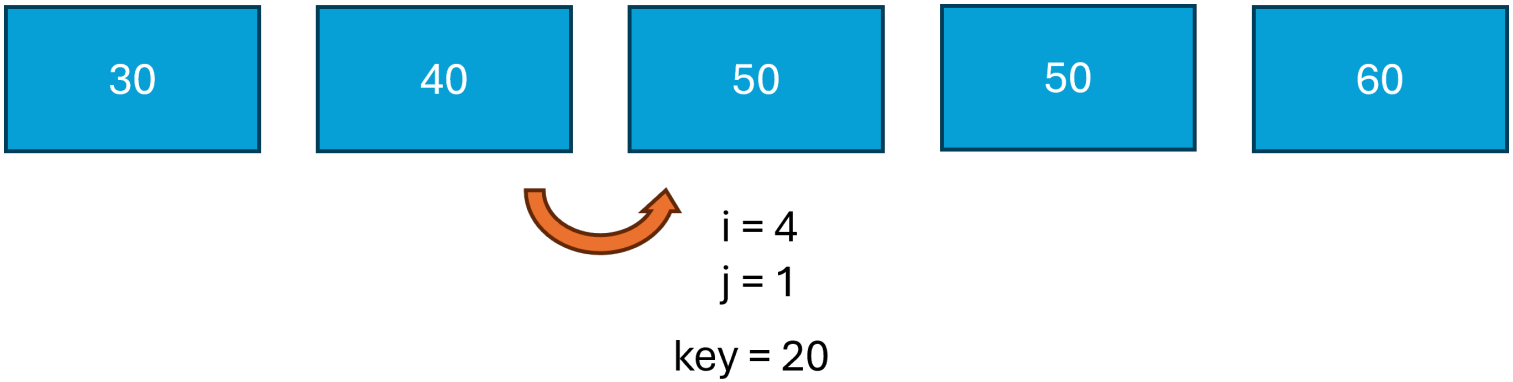
j = 3

key = 20

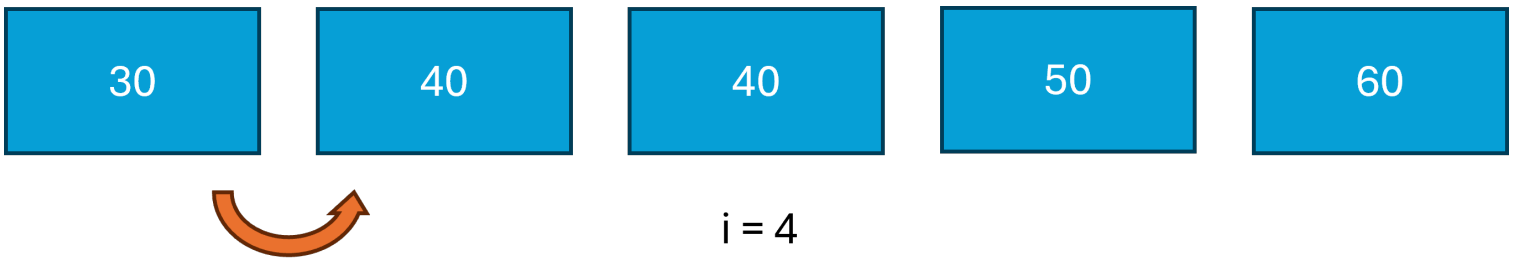
Insertion sort



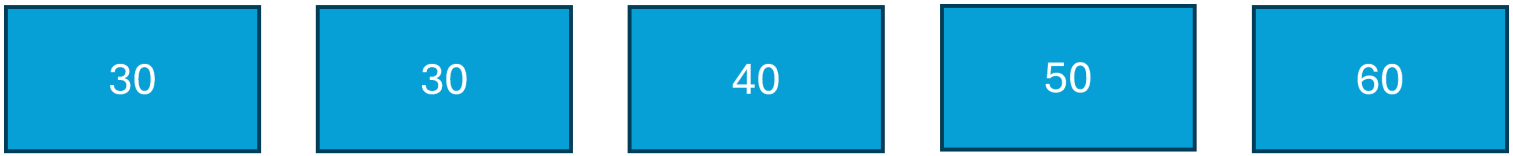
Insertion sort



Insertion sort



Insertion sort



i = 4

j = -1

key = 20

Insertion sort

i = 4

j = -1

key = 20

Insertion sort

```
1  void insertionSort(int arr[], int n){
2      int i, key, j;
3      for (i = 1; i < n; i++) {
4          key = arr[i];
5          j = i - 1;
6          while (j >= 0 && arr[j] > key) {
7              arr[j + 1] = arr[j];
8              j = j - 1;
9          }
10         arr[j + 1] = key;
11     }
12 }
```

Insertion sort

時間複雜度

比較次數 $1+2+3+...+(n-2)+(n-1)$
時間複雜度爲 $O(n^2)$

quick sort

想法

Quick sort 快速排序演算法是一種 divide and conquer 的陣列排序方法，其過程如下：

1. 先從 array 中選出一個元素當基準 (pivot)，然後讓 pivot 左邊的元素都小於 pivot，pivot 右邊的元素都大於等於 pivot。這個過程稱爲 partition。
2. 再分別對 pivot 左邊和右邊的 array 重複以上過程，就可以達到排序的效果。

例子

假設有個 array，初始狀態 = [9, 4, 1, 6, 7, 3, 8, 2, 5]。
首先，選定 5 作爲 pivot。我們把小於 pivot 的通通擺在左邊，剩下的擺右邊，結果如下：

```
<--小於pivot--|--大於pivot->
[4, 1, 3, 2, 5, 9, 6, 7, 8]
             ^pivot
```

partition()

爲了實作 quick sort，我們需要一個輔助函式 partition()。

partition() 的作用是從 array 中選出一個 pivot 當作標準，用這個 pivot 把 array 分成兩半，使得左半邊元素全部小於等於 pivot，右半邊元素全部大於等於 pivot。注意它會直接修改原本的 array。

partition() - 演算法

這邊介紹一種 partition 的演算法叫 Hoare。

partition 概念如下：

1. 選擇陣列中央的元素作爲 pivot。
2. 從最前面開始掃描大於 pivot 的元素，從最後面開始掃描小於 pivot 的元素，找到之後交換。
3. 重複以上步驟，直到兩邊掃描線相遇。

實作 - partition

做完 partition，j 左邊的元素都會小於等於 pivot，右邊都會大於等於 pivot。實作如下：

```
int partition(int arr[], int lo, int hi) {
    int pivot = arr[(lo + hi) / 2];
    int i = lo - 1, j = hi + 1;
    while (true) {
        do {
            i++;
        } while (arr[i] < pivot);
        do {
            j--;
        } while (arr[j] > pivot);

        if (i >= j) {
            return j;
        }
        swap(arr, i, j);
    }
}
```

實作 - quick sort

quick sort 實作如下：

```
1 void quickSort(int arr[], int lo, int hi) {
2     if (lo >= 0 && hi >= 0 && lo < hi) {
3         int pivot = partition(arr, lo, hi);
4         quickSort(arr, lo, pivot); // 注意 pivot 有包含在內
5         quickSort(arr, pivot + 1, hi);
6     }
7 }
```

quick sort

上面教的 Hoare 演算法可以隨邊挑 pivot，不一定要是中間。
其實還有一種演算法叫做 Lomuto，實作會比較容易且直觀，但受限於只能挑當前陣列的最右邊爲 pivot，有興趣可以去了解看看。