# 程式設計作業 5 報告

**407510062** 吳翔詠

關於三種排序法的比較:

1. 測試環境: CPU=>Intel Core i7-7700HQ@ 2.8GHz *2
   **RAM 8.0GB**

   使用系上工作站測試

2. 各個程式碼:

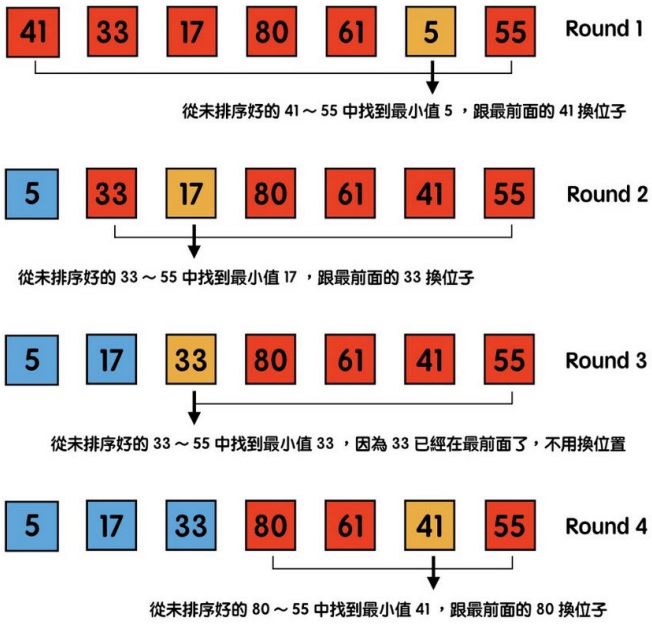
   **Bubblesort:**重複比較後一個數字，如果比較大就交換，時間複雜度平均為 **O($n^2$)**，是一個穩定的排列。

```
void BubbleSort(int array[], int len)
{
    int i, j, temp;
    for (i = 0; i < len; i++)
    {
        for (j = 0; j < len - 1 - i; j++)
        {
            if (array[j] > array[j + 1])
            {
                temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
            }
        }
    }
    return;
}
```

   **SelectionSort:**從第一個數字開始，比較後面數中的最小值，如果第一個數比他大，他們交換，以此類推。時間複雜度平均為 **O($n^2$)**，由於會放到位排序資料的最前面可能會破壞原先的順序(即使有兩

個相同大小的數字)，所以不是穩定排序。

簡單圖例:

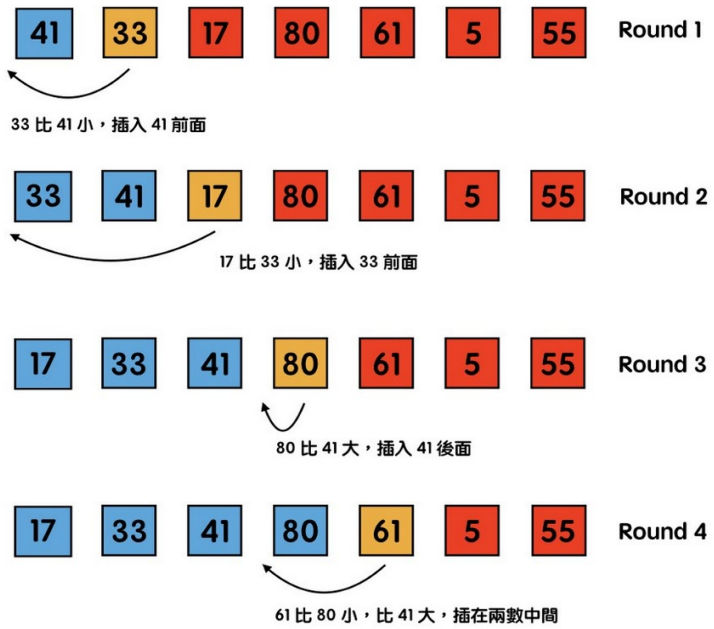



```c
void selection_sort(int list[], int n)
{
    int i, j, min_id, temp;
    for (i = 0; i < n - 1; i++)
    { //固定i比較
        min_id = i;
        for (j = i + 1; j < n; j++)
        { //跟i換位置
            if (list[j] < list[min_id])
                min_id = j;
        }
        temp = list[i];
        list[i] = list[min_id];
        list[min_id] = temp;
    }
    return;
}
```

**Insertsort:**從第二個數字開始，跟他前面的數字比較，當位於兩個數中間時，插進該位置，否則是最小或最大，時間複雜度平均為

**$O(n^2)$**，是一個穩定的排列。

簡單圖例:




```c
void InsertSort(int A[], int n)
{
    int i, j, Temp;
    for (i = 1; i <= n; i++)
    {
        Temp = A[i];
        j = i - 1;
        while (Temp < A[j])
        {
            A[j + 1] = A[j];
            j--;
            if (j == -1)
                break;
        }
        A[j + 1] = Temp;
    }
    return;
}
```

**3. 建立 Data 方法及數量:**

```
int main()
{
    int i;
    int test_data[TEST_DATA_CNT + 5];
    srand(1);
    for (i = 0; i < TEST_DATA_CNT; i++)
    {
        test_data[i] = rand();
    }
```

使用 rand()建立隨機亂數，亂數種子設為 1。

測試數量分為 10000 跟 50000

**4. 測量排序時間方式:**

助教提供的方法

```
#include <sys/time.h>
```

```
struct timeval start;
struct timeval end;
unsigned long diff;

gettimeofday(&start, NULL);
selection_sort(test_data, TEST_DATA_CNT);
gettimeofday(&end, NULL);

diff = 1000000 * (end.tv_sec - start.tv_sec) + end.tv_usec - start.tv_usec;
```

**5. 實驗結果:**

兩次結果分別為 **0.475786、0.27204、0.124654**

**11.940627、5.179708、3.125063** 秒。

```
u07510062@csie2:~ % gcc ./BubbleSort.c
u07510062@csie2:~ % ./a.out
Sorting performance 475786 us (equal 0.475786 sec)
u07510062@csie2:~ % gcc ./SelectionSort.c
u07510062@csie2:~ % ./a.out
Sorting performance 207204 us (equal 0.207204 sec)
u07510062@csie2:~ % gcc ./InsertionSort.c
u07510062@csie2:~ % ./a.out
Sorting performance 124654 us (equal 0.124654 sec)
u07510062@csie2:~ %
```

```
u07510062@csie2:~ % gcc ./BubbleSort.c
u07510062@csie2:~ % ./a.out
Sorting performance 11940627 us (equal 11.940627 sec)
u07510062@csie2:~ % gcc ./SelectionSort.c
u07510062@csie2:~ % ./a.out
Sorting performance 5179708 us (equal 5.179708 sec)
u07510062@csie2:~ % gcc ./InsertionSort.c
u07510062@csie2:~ % ./a.out
Sorting performance 3125063 us (equal 3.125063 sec)
u07510062@csie2:~ %
```

**6. 總結:**

從結果來看插入排序是最快的，泡沫則最慢，當資料為 **10000** 筆時差異還不是很大，但當資料到 **50000** 筆時，泡沫排序時間明顯倍增，由此可見泡沫排序還是比較適合小型資料跟另外兩種排序相比。由時間複雜度來看，三筆資料是相同的，也意味著，若要真的運用在實務上面時，還是必須選擇更快的排序法。

**7. 參考資料:**


http://spaces.isu.edu.tw/upload/18833/3/web/sorting.htm

https://medium.com/appworks-school/%E5%88%9D%E5%AD%B8%E8%80%85%E5%AD%B8%E6%BC%94%E7%AE%97%E6%B3%95-%E6%8E%92%E5%BA%8F%E6%B3%95%E5%85%A5%E9%96%80-%E9%81%B8%E6%93%87%E6%8E%92%E5%BA%8F%E8%88%87%E6%8F%92%E5%85%A5%E6%8E%92%E5%BA%8F%E6%B3%95-23d4bc7085ff