作業內容

上課時老師有提到 Quicksort 可以透過配合 InsertionSort 來增進排序效率。本次的作業內容為透過 Java 實現課本 Pseudocode Quicksort，並透過 InsertionSort 加以改良。

作業要求

※請勿抄襲、複製網路上或同學的程式碼，違者不論主從以零分計算※

- 以 Java 程式撰寫，實現課本 Pseudocode 的 Quicksort 與 InsertionSort 結合進行改良。
  - 本次作業希望同學練習透過 Pseudocode 寫成實體程式碼的實作能力，故須以課本 Pseudocode 為主，若參考非課本的版本將會影響評分。
- 使用第 2 頁所給定的三組陣列內容當作輸入值，並印出和第 3 頁圖 2，輸出結果所示的相同結果。
- 當 Subarray 元素個數小於等於 3 時，改用 InsertionSort 增進排序效率。
  - 若只實作原始 Quicksort，並未透過 InsertionSort 增進排序效率，會影響評分。
- 定義 Quicksort()、HoarePartition()、InsertionSort()三個 function，並將對應的功能實作在 function 當中。
  - 你可以定義其他 Function 協助使用，但必須註解說明其用途。
- 在每一次 HoarePartition()結束或者 InsertionSort()結束之後，印出使用之方法與結束後之陣列內容。
- 請適當註解，以便批改。

Java Project 需求

- 專案名稱「Quicksort_學號」
- Class 名稱「Main」
- Use an execution environment JRE：JavaSE-17（為確保程式能正確呈現結果，請一律使用此版本進行編譯）
- 「Text file encoding」中改為「Other」，並選擇「UTF-8」

（詳細設定可參考公告中另一附件「Eclipse 安裝與輸出說明」）

作業輸入

```
int[] array_1 = {10, 4, 2, 8, 7, 3, 5, 9, 6, 1};
int[] array_2 = {1, 6, 14, 13, 7, 2, 11, 10, 4, 9, 5, 8, 12, 3, 15};
int[] array_3 = {8, 20, 16, 13, 15, 17, 12, 18, 19, 7, 10, 5, 4, 14, 2, 6, 1, 11, 9, 3};
```

```
2
3  public class Main {
4
5      public static void main(String[] args) {
6
7          int[] array_1 = {10, 4, 2, 8, 7, 3, 5, 9, 6, 1};
8          int[] array_2 = {1, 6, 14, 13, 7, 2, 11, 10, 4, 9, 5, 8, 12, 3, 15};
9          int[] array_3 = {8, 20, 16, 13, 15, 17, 12, 18, 19, 7, 10, 5, 4, 14, 2, 6, 1, 11, 9, 3};
10
11         //todo
12         System.out.printf("%n----------------------------------------%n%n");
13
14         //todo
15
16         System.out.printf("%n----------------------------------------%n%n");
17
18         //todo
19     }
20
21     public static void Quicksort() {
22         //todo
23     }
24
25     public static void HoarePartition() {
26         //todo
27     }
28
29     public static void InsertionSort() {
30         //todo
31     }
32
33 }
```

圖 1，程式碼輸入範例

作業輸出

需參照以下範例進行列印，列印內容須與範例一致。

- 在開始之前與結束之後列印陣列內容。
- 於每一次執行完 HoarePartition 或 InsertionSort 之後，列印「Use partition: 」或「Use Insertion: 」+當前陣列內容。
- 每一次列印完記得換行。

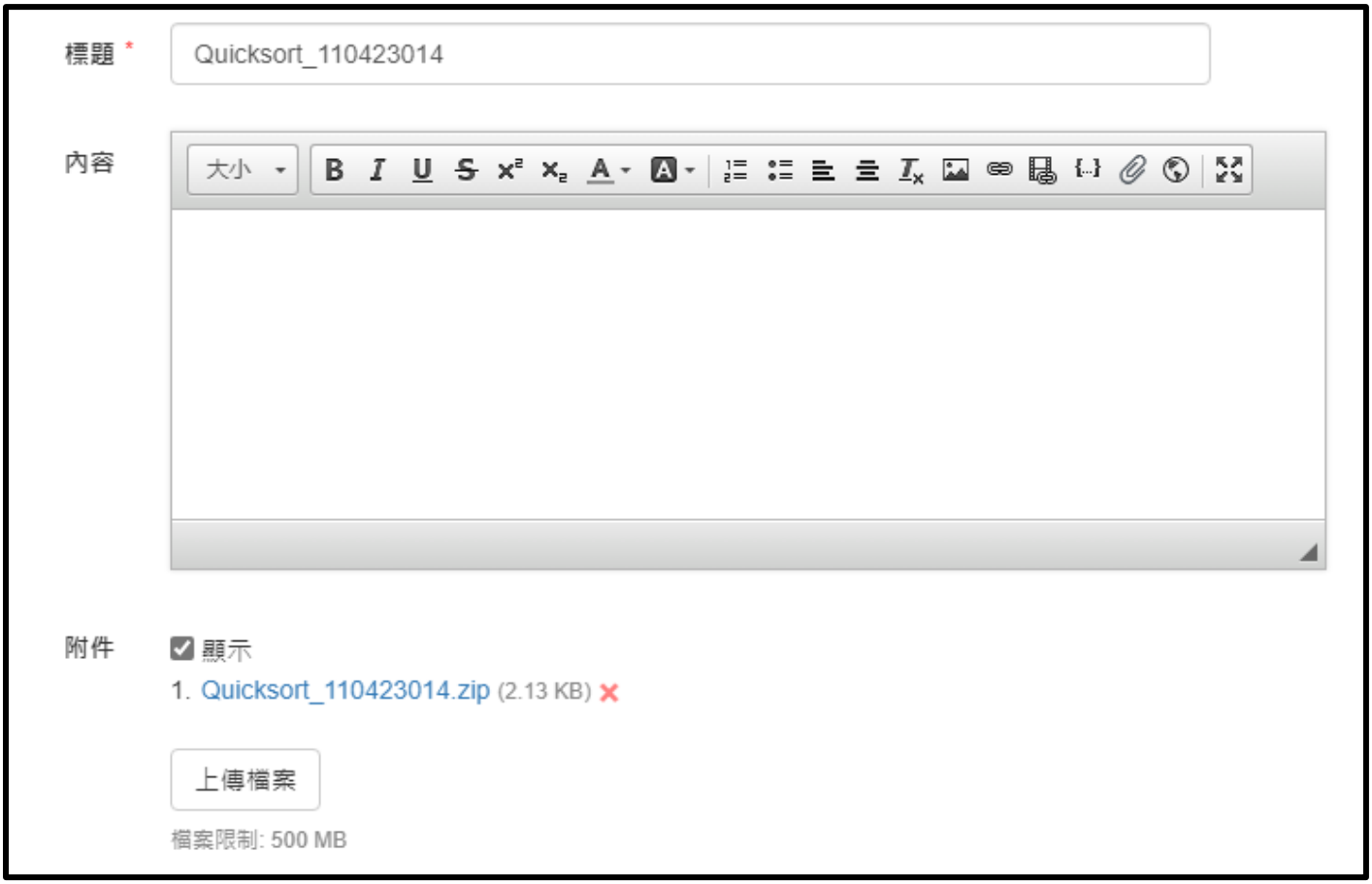
```
before sorting is [10, 4, 2, 8, 7, 3, 5, 9, 6, 1]
Use partition:[1, 4, 2, 8, 7, 3, 5, 9, 6, 10]
Use partition:[1, 4, 2, 8, 7, 3, 5, 9, 6, 10]
Use partition:[1, 3, 2, 4, 7, 8, 5, 9, 6, 10]
Use Insertion:[1, 2, 3, 4, 7, 8, 5, 9, 6, 10]
Use partition:[1, 2, 3, 4, 5, 6, 7, 9, 8, 10]
Use Insertion:[1, 2, 3, 4, 5, 6, 7, 9, 8, 10]
Use Insertion:[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
After sorting is  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

----------------------------------------

before sorting is [1, 6, 14, 13, 7, 2, 11, 10, 4, 9, 5, 8, 12, 3, 15]
Use partition:[1, 6, 14, 13, 7, 2, 11, 10, 4, 9, 5, 8, 12, 3, 15]
Use partition:[1, 2, 3, 5, 4, 6, 11, 10, 7, 9, 13, 8, 12, 14, 15]
Use partition:[1, 2, 3, 5, 4, 6, 11, 10, 7, 9, 13, 8, 12, 14, 15]
Use Insertion:[1, 2, 3, 4, 5, 6, 11, 10, 7, 9, 13, 8, 12, 14, 15]
Use partition:[1, 2, 3, 4, 5, 6, 8, 10, 7, 9, 11, 13, 12, 14, 15]
Use partition:[1, 2, 3, 4, 5, 6, 7, 8, 10, 9, 11, 13, 12, 14, 15]
Use Insertion:[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 12, 14, 15]
Use partition:[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
Use Insertion:[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
After sorting is  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

----------------------------------------

before sorting is [8, 20, 16, 13, 15, 17, 12, 18, 19, 7, 10, 5, 4, 14, 2, 6, 1, 11, 9, 3]
Use
Use
Use
Use
Use
Use
Use
Use
Use
Use
Use
Use
Use
After sorting is  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

防雷

圖 2，輸出結果

EE-Class 繳交

將輸出的專案上傳至 EE-Class 作業區。



- 檔案名稱「Quicksort_學號.zip」
- 繳交期限至 5/29 23:59 分，繳交期限截止時，會自動關閉作業區，系統關閉後一律不再補交，請同學盡早完成上傳作業

課本虛擬碼

ALGORITHM HoarePartition(A[l…r])

// Partitions a subarray by Hoare's algorithm, using the first element as pivot

// Input : Subarray of array A[0…n-1], defined by its left and right indices l and r (l< r)

// Output: Partition of A[left…right], with the split position returned as this function's value

p←A[l]

i←l; j←r+1

repeat

    repeat i←i+1 **until A**[i] ≥ p

    repeat j←j-1 **until** A[j] ≤ p

    swap(A[i], A[j])

until i ≥ j

swap(A[i], A[j]) //undo last swap when i ≥ j

swap(A[l], A[j])

return j

ALGORITHM Quicksort(A[l…r])

// Sorts a subarray by quicksort

// Input : Subarray of array A[0...n-1], defined by its left and right indices l and r

// Output : Subarray A[l...r] sorted in nondecreasing order

if l < r

    s←HoarePartition(A[l…r]) //s is a split position

    印出指定內容() // Hint 請在 HoarePartition 或者 InsertionSort 之後印出當前陣列內容。

    Quicksort(A[l…s-1])

    Quicksort(A[s+1…r])

ALGORITHM InsertionSort(A[0…n-1])

//Sorts a given array by insertion sort

//Input: An array A[0…n-1] of n orderable elements

//Output: Array A[0…n-1] sorted in nondecreasing order

For i←1 **to** n-1 **do**

    v←A[i]

    j←i-1

    while j≥0 **and** A[j] > v **do**

        A[j+1]←A[j]

        j←j-1

    A[j+1]←v