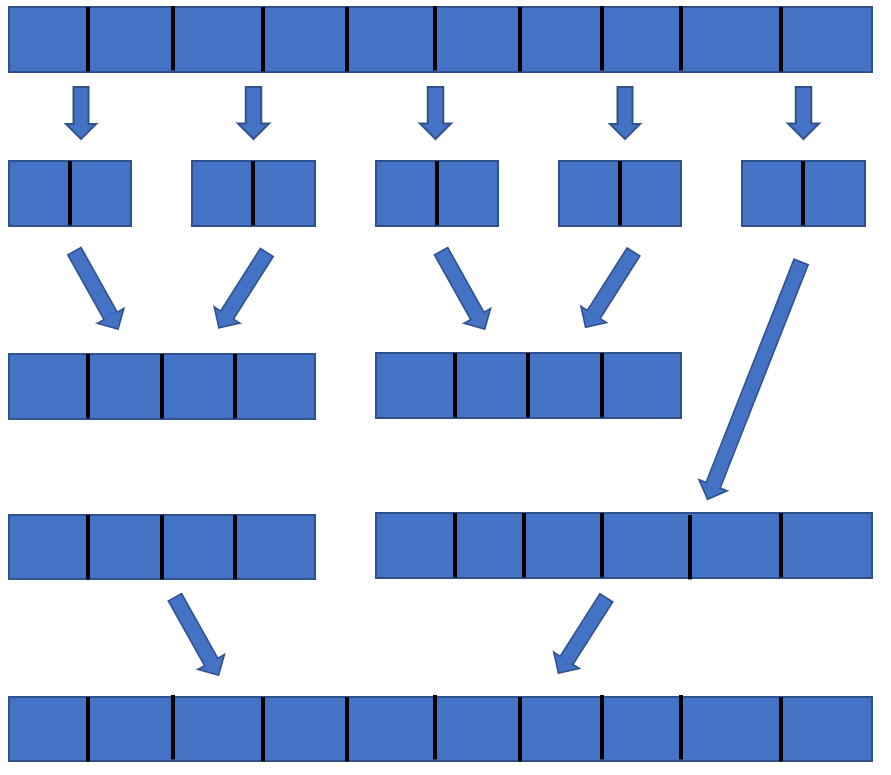
電機系 資料結構緒論 作業四

姓名：郭庭維

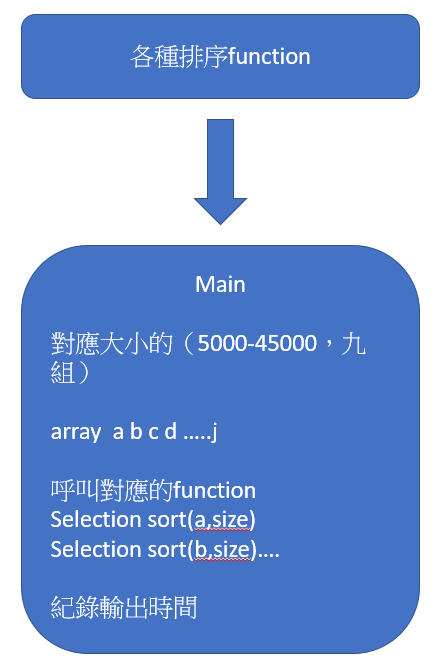
系級：會計 112

學號：H14086030

1. 實作並簡述(1) ~ (6) sort的運作方式
   1. Insertion sort：從數列前面開始排序，確保前n個數已經排好序後，將第n+1個數加入數列對應的位子。
   2. Selection sort：掃描整個陣列把第一小的數字跟數列第一個數字交換，接著把第二小的跟數列第二個數字交換
   3. Quick sort：
      1. 以某個數字為分界點（e.g.第一個數）
      2. 左邊與右邊各有個index，左邊確保每個數都小於分界點，右邊則確保都大於分界點，如果有左邊有不符合的就和也不符合右邊的交換位子。
      3. 重複以上步驟直到左index在右index的右邊（交換位置）
      4. 左index左側的所有數字都小於分界點，右側則都大於分界點
      5. 將小於與大於的部分重新做一次quick sort
   4. Merge sort
      1. 把數列拆成兩個數一組
      2. 將這兩個數字排序
      3. 與旁邊其他數組合併，合併成由小到大的數列
      4. 持續步驟三直到所有都被合併回一個數列



* 1. Heap sort
     1. 把數列建構成max heap tree
     2. Pop a[0]，把它移到末端，例如max heap tree長度為8，則把a[0]，移動到a[8]
     3. 重新建構max heap tree，重複步驟一二
  2. Radix sort
     1. 把數列中個位數字為0 1 2 3…依序放到名稱0 1 2 3…的array暫存
     2. 依序把0 1 2 3 Array拿出來（個位數字是1的會比2還要前面）
     3. 重複步驟一、二 依序用 個十百千萬位數做為排序標準
     4. 執行到最高位數的後，最後排列結果即為排序後的結果

1. 紀錄每次的CPU運行時間並輸出
   1. 測試環境說明：  
      作業系統：win 10 pro  
      CPU：i5-9400F
   2. 測試方法：
      1. 寫好各種sort function後，把透過其他資源建立的隨機、不重複、大小相同的十個數列放在main中（數字範圍為1-200萬）
      2. 呼叫對應的sort function紀錄執行十次所需要的時間
      3. 時間則使用這個時間做為執行時間，因為沒有使用print function印出排列後的數列，所以我相信這個時間還算精確。一張含有 文字 的圖片

         自動產生的描述
   3. 執行時間統計結果如下表，備註：merge sort 與radix在資料量50000筆時會”with return value 3221225725”，推測是遞迴過多的緣故，所以資料從缺，因為認為九組數據也足夠了，因此沒有再測試下去。下面的時間都是執行十次排序的總時間，沒有另外除10求平均。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| size | Insertion sort | selection sort | quick sort | merge sort | heap sort | radix sort |
| 5000 | 0.3365 | 0.2942 | 0.02766 | 0.03126 | 0.03742 | 0.03023 |
| 10000 | 1.3 | 1.17 | 0.03299 | 0.03379 | 0.04652 | 0.03771 |
| 15000 | 2.78 | 2.42 | 0.04337 | 0.04779 | 0.06745 | 0.04687 |
| 20000 | 5.01 | 4.351 | 0.0491 | 0.05402 | 0.08545 | 0.05398 |
| 25000 | 7.83 | 6.912 | 0.05724 | 0.0644 | 0.1159 | 0.05706 |
| 30000 | 11.38 | 9.632 | 0.06739 | 0.07177 | 0.1202 | 0.06637 |
| 35000 | 15.1 | 13.1 | 0.06921 | 0.08348 | 0.1358 | 0.07753 |
| 40000 | 19.85 | 17.12 | 0.07732 | 0.09199 | 0.1619 | 0.08202 |
| 45000 | 25.66 | 22.23 | 0.08562 | 0.1009 | 0.178 | 0.08365 |
| 50000 | 30.93 | 26.65 | 0.09306 |  | 0.192 |  |

1. 建立圖表
2. (1) ~ (6) sort的優缺點
   1. 從上面圖上可以看到 selection sort跟 insertion sort很明顯是效率最差的兩種排序方法，這也跟我們在課本上學到的複雜度是吻合的，20000筆資料所需的時間幾乎是10000的4倍，四萬筆資料也是兩萬筆的4倍左右
   2. 相較之下其他種算法對於資料量大小的影響還是有，但少了很多，資料量十倍之下，大概只有三倍左右的差距
   3. 優缺點：
      1. Selection sort／Insertion sort：  
         優點：直觀、實作容易  
         缺點：但是效率最差
      2. Quick sort：  
         優點：這次幾種算法中，是第二快的排序算法  
         缺點：unstable sort、worse case O(n^2)
      3. Merge sort  
         優點：stable sort，效率也不錯，可以支援external sort  
         缺點：可能會需要額外空間，這次測試中發現過大的資料會出現錯誤，可能要改成external sort
      4. Heap Sort  
         優點：如果用Array來建構heap，不需要額外空間，效率不差  
         缺點：nonstable sort。效率不及其他排序法
      5. Radix sort  
         優點：這次測試中，屬於效率最高的算法  
         缺點：需要大量的空間來存放，而且測試過程會發現無法應付過大的資料
3. 心得與討論：  
   　　我覺得有這次的機會可以實際測試每一種排序算法的效率是一件很有趣的事情，看到複雜度的算法跟其他排序法比起來效率簡直是天壤之別，如此具體的差異讓我以後在寫程式的時候會更加重視演算法複雜度的問題。但是比較可惜的是這次的作業時間上有點趕，加上自己同時還有3-4份期末報告要交沒有辦法花太多時間在處理程式上的bug。有的排序法如果時間到沒有寫出來，卡住的部分只能先參考課本。暑假後會自己重新地把這些算法再練習一遍打好自己的基礎。  
   　　另外看中山大學的影片時候有提到radix sort的效率應該要是最高的，但我自己實驗看來，在前面幾次資料量比較小的時候，quick sort會比 radix sort還快，但是25,000筆資料後，確實radix sort會比較快。