0819823陳子祈 演算法導論 HW1

一、試驗成果

一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

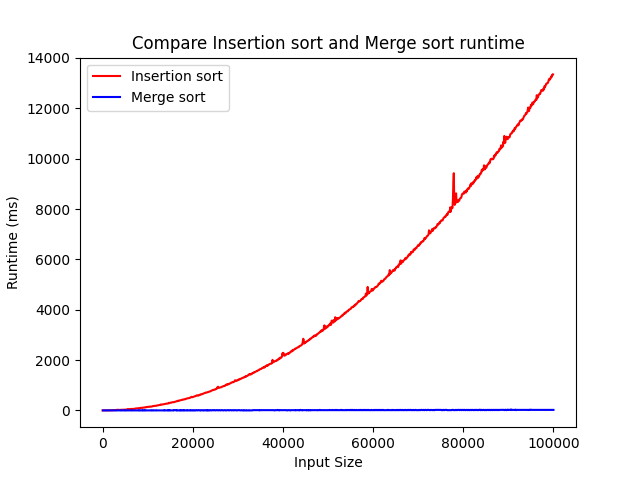
自動產生的描述

根據基本要求1、2，我使用for迴圈設定了不同大小的input array，大小從100逐次增加100到100000，根據實測結果如下:

1. Insertion Sort

|  |  |
| --- | --- |
| Input size | Runtime(ms) |
| 100 | 0 |
| 1000 | 9 |
| 10000 | 140 |
| 100000 | 13505 |

就Insertion sort而言，10倍的input size，執行時間約成長100倍，若n為input size，f(n)為Insertion執行的時間，符合O(f(n)) = n^2的趨勢。



若使用python將Insertion Sort與Merge Sort一同繪製成圖表，可看出Insertion Sort的成長速率比Merge Sort快非常多，因此後面有將Merge Sort的Runtime獨立出來製作成圖表。

1. Merge Sort

|  |  |
| --- | --- |
| Input size | Runtime(ms) |
| 100 | 0 |
| 1000 | 0 |
| 10000 | 0 |
| 100000 | 21 |

就Merge sort而言，由於執行時間實在太快，只能觀察到Input size大於10000以上的執行時間，所以我把設定了不同大小的input array的for迴圈改為:

一張含有 文字 的圖片

自動產生的描述

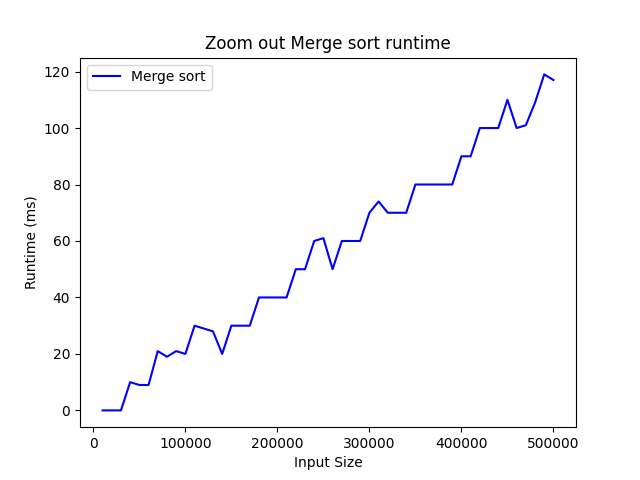
重製表格如下:

|  |  |
| --- | --- |
| Input size | Runtime(ms) |
| 100000 | 20 |
| 200000 | 40 |
| 300000 | 70 |
| 400000 | 90 |
| 500000 | 117 |

就Merge sort而言，2倍的input size，執行時間約成長2倍，3倍的input size，執行時間約成長3.5倍，4倍的input size，執行時間約成長4.5倍，5倍的input size，執行時間約成長6倍，若n為input size，f(n)為Insertion執行的時間，大致符合O(f(n)) = n\*lg(n)的趨勢。

(2lg2 = 2, 3lg3 = 4.75, 4lg4 = 8, 5lg5 = 11)

使用python繪製成圖表:



依此圖表可觀察到Input size越大，Runtime以一定的速率成長。

二、參考實測方法

(一) Sorting algorithm 正確性

1. 第54行將註解消除掉，可檢查未排序過的Input array

一張含有 文字 的圖片

自動產生的描述

2. 第78行將註解消除掉，可檢查經Insertion sort排序過的array

一張含有 文字 的圖片

自動產生的描述

3. 第130行將註解消除掉，可檢查經Merge sort排序過的array。

一張含有 文字 的圖片

自動產生的描述

4. 將第160行的for迴圈改成固定size，可檢查固定size的經Insertion sort排序過的array

一張含有 文字 的圖片

自動產生的描述

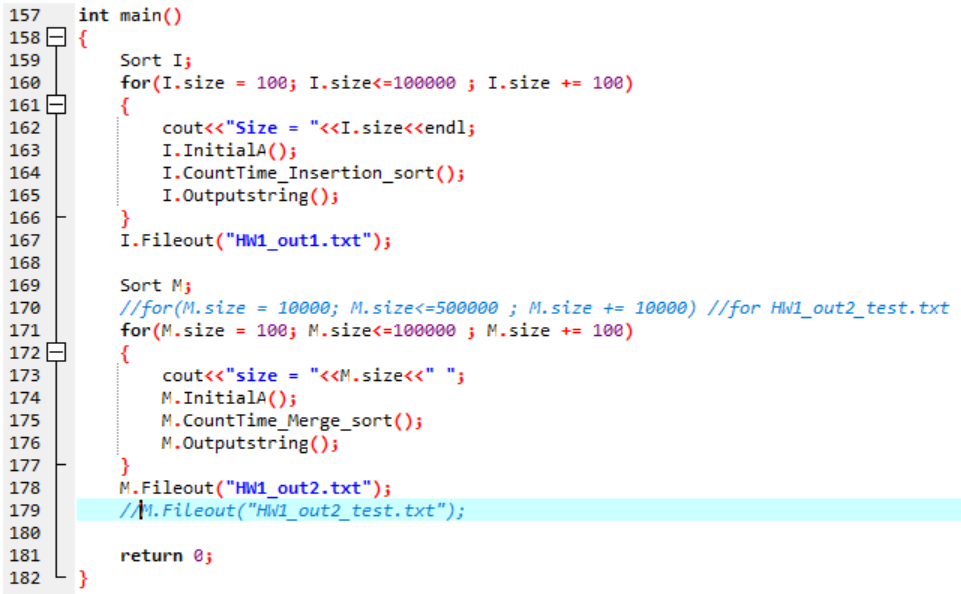
5. 同樣需將第170行的for迴圈改成固定size，可檢查固定size的經Merge sort排序過的array

一張含有 文字 的圖片

自動產生的描述

(二) 使用Python將兩種Sorting演算法的Runtime繪製成圖表

1. 請將167、178行輸出檔案檔名照下面這樣寫，並編譯、執行程式

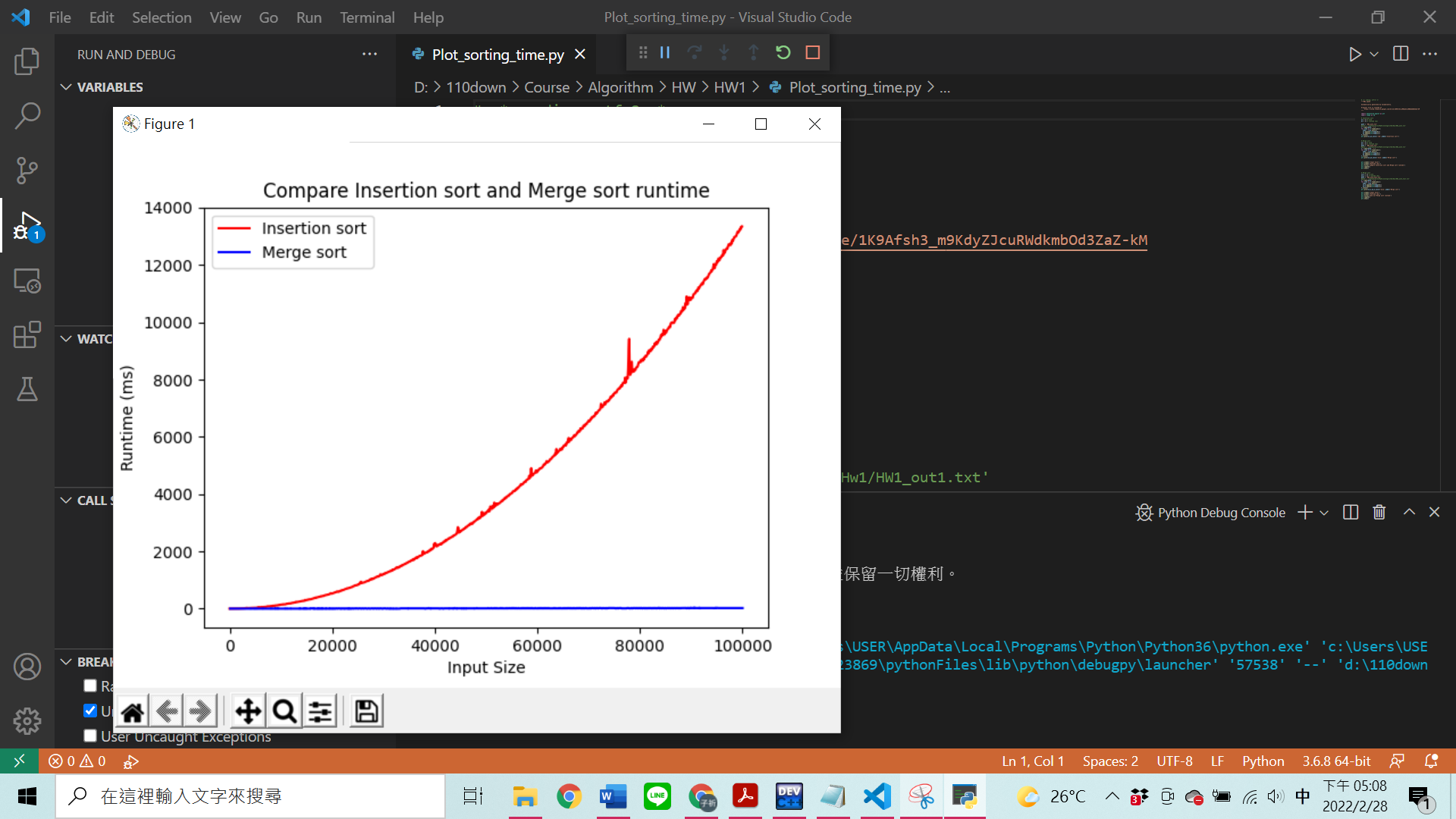


2. 若有python IDE，打開Plot\_sorting\_time.py

一張含有 桌 的圖片

自動產生的描述

3. 執行Python程式



(三) 使用Python將Merge sort在Input size超過10000的Runtime繪製成圖表

1. 請拿掉第170行註解，註解171行，注意輸出檔案名需改成179行那樣，編譯並執行程式一張含有 文字 的圖片

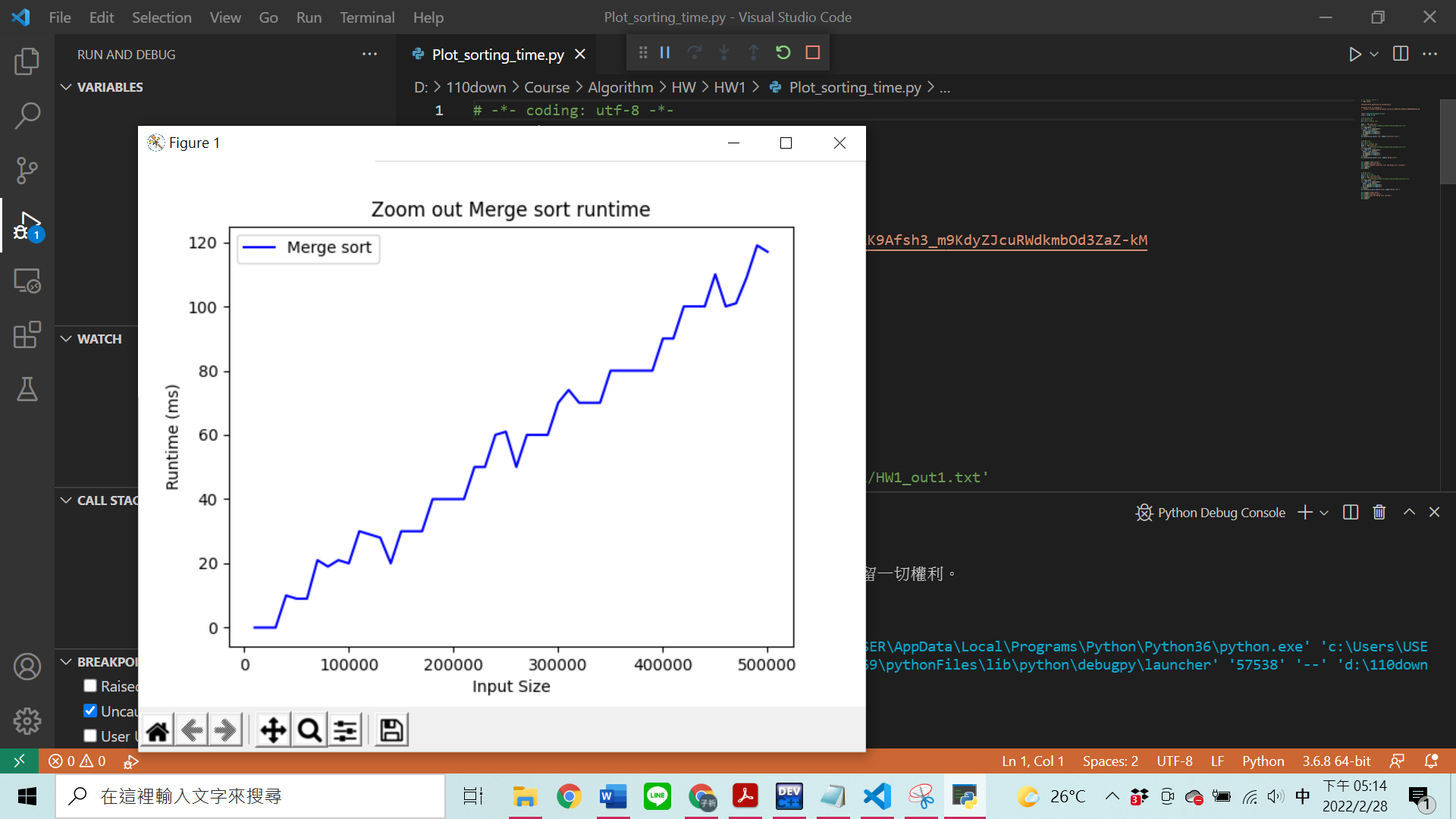
自動產生的描述

2. 若有python IDE，打開Plot\_sorting\_time.py

一張含有 桌 的圖片

自動產生的描述

3. 執行Python程式，將第一個圖表關掉之後，第二個圖表出現即為成功



三、心得

這次是第一個演算法的作業，題目雖不難，演算法也可直接參考課本上提供的方法，不過把整個程式實作、包裝成可以跑且比較好看懂的程式需要花很多時間，甚至我一開始想直接在C++實現Python繪製圖表的功能，後來發現實在太難，於是改成輸出到txt檔，再交給Python讀取txt檔並繪製成圖表比較簡單。繪製成圖表之後，雖然Insertion sort的曲線很漂亮，但Insertion sort與Merge sort的執行時間實在差異太大到完全無法比較，且在Input array size小於10000時Merge sort的執行時間小到連測都測不出來，因此只好測超過10000時Merge sort的執行時間。總之這次作業讓我對這兩個排序演算法的方法、執行時間複雜度、實作、繪製成圖表等更加熟悉，我覺得獲益良多，希望下次還可以做這種比較開放性、不會太難的題目。