0819823 陳子祈 演算法導論 HW1

一、試驗成果

```
41  void Sort::InitialA()
42  {
43
44      srand( time(NULL) );
45      //int up = size, low = 1;
46      int r;
47      for(int i=0; i<size; i++)
48      {
49          r = (int)((rand() / (RAND_MAX+1.0)) * (size - 1.0) + 1.0);
50          A.push_back(r);
51          //cout<<"A["<<i<<"] = "<<A[i]<<" "<<"r = "<<r<<endl;
52      }
53
54      /*for(int i=0; i<size; i++)
55          A.push_back(size-i);    */ //Insertion worst case
56      //PrintA();
57  }
```

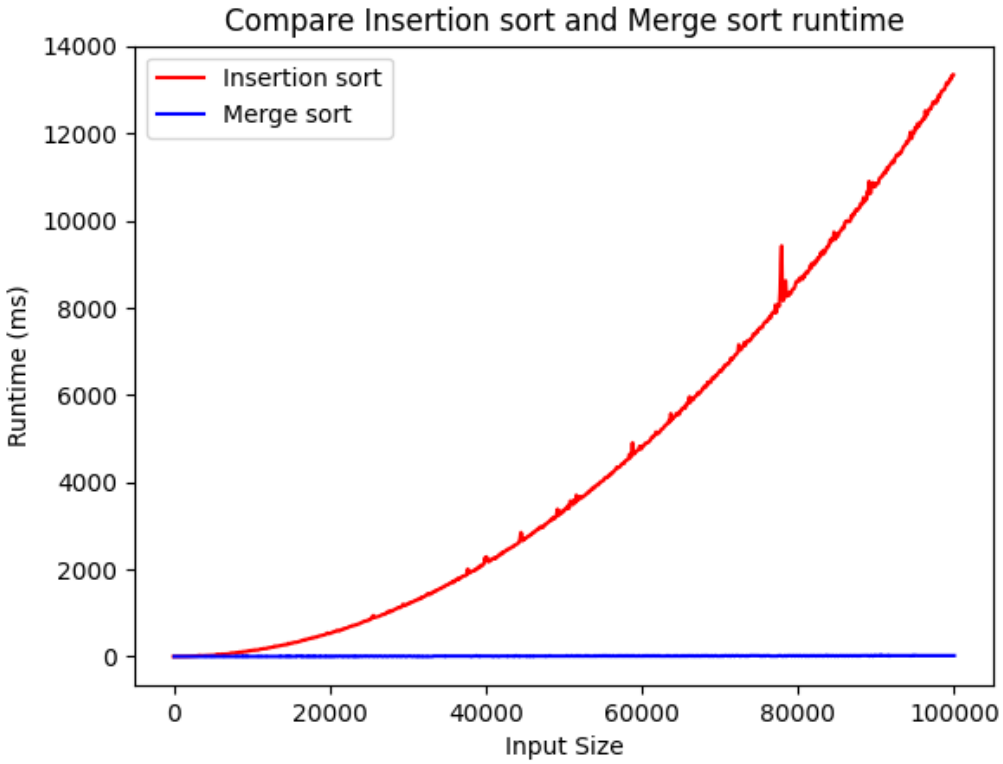
```
161     Sort I;
162     for(I.size = 100; I.size<=100000 ; I.size += 100)
163     {
164         cout<<"Size = "<<I.size<<endl;
165         I.InitialA();
166         I.CountTime_Insertion_sort();
167         I.Outputstring();
168     }
169     I.Fileout("HW1_out1.txt");
```

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:

(一) 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。

(二) 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:

```
171     Sort M;
172     for(M.size = 10000; M.size<=500000 ; M.size += 10000)
173     //for(M.size = 100; M.size<=100000 ; M.size += 100)
174     {
175         cout<<"size = "<<M.size<<" ";
176         M.InitialA();
177         M.CountTime_Merge_sort();
178         M.Outputstring();
179     }
180     //M.Fileout("HW1_out2.txt");
181     M.Fileout("HW1_out2_test.txt");
```

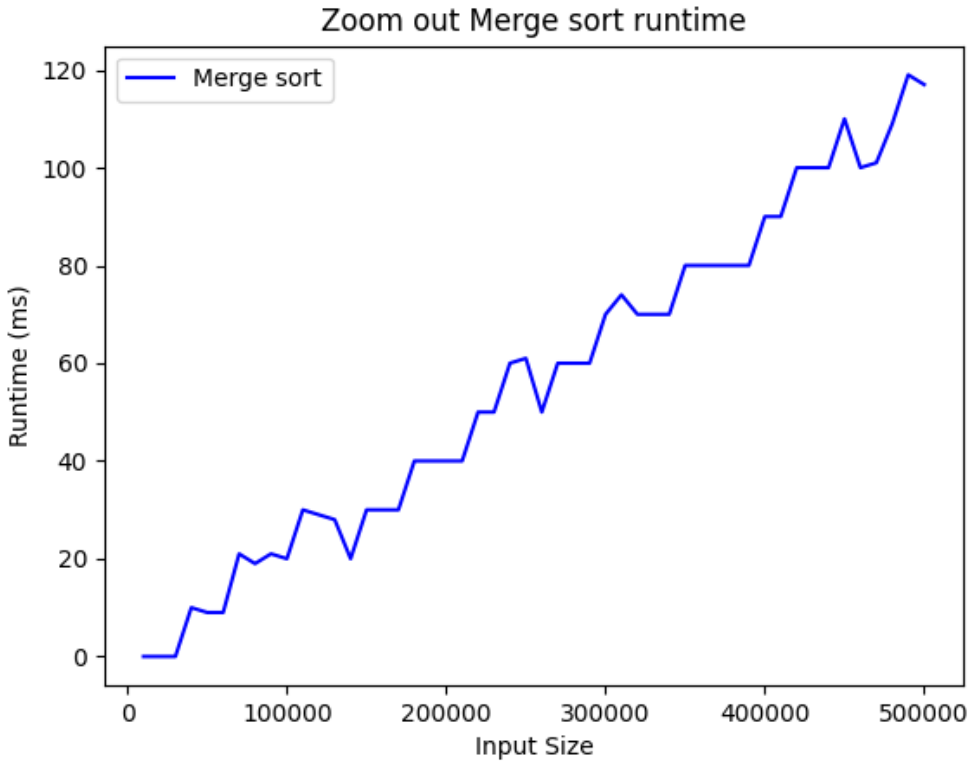
重製表格如下:

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

```
41  void Sort::InitialA()
42  {
43
44      srand( time(NULL) );
45      //int up = size, low = 1;
46      int r;
47      for(int i=0; i<size; i++)
48      {
49          r = (int)((rand() / (RAND_MAX+1.0)) * (size - 1.0) + 1.0);
50          A.push_back(r);
51      }
52      /*for(int i=0; i<size; i++)
53          A.push_back(size-i);    */ //Insertion worst case
54      //PrintA();
55  }
```

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

```
73  void Sort::CountTime_Insertion_sort()
74  {
75      Begin = clock(); //Start counting time
76      Insertion_sort();
77      End = clock(); //Finish counting time
78      //PrintA(); //Use this to test the correctness of the sorting function
79      A.clear();
80      duration = (double)(End - Begin) ; // Evaluating ticking times
81      cout << "duration = " << duration <<" ms."<< endl;
82  }
```

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

```
125 void Sort::CountTime_Merge_sort()
126 {
127     Begin = clock();//開始計時
128     Merge_sort(0,size-1);
129     End = clock(); //結束計時
130     //PrintA(); //Use this to test the correctness of the sorting function
131     A.clear();
132     duration = (double)(End - Begin) ; // Evaluating ticking times
133     cout << "duration = " << duration <<" ms."<< endl;
134 }
```

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

```
159     Sort I;
160     for(I.size = 100; I.size<=100000 ; I.size += 100)
161     {
162         cout<<"Size = "<<I.size<<endl;
163         I.InitialA();
164         I.CountTime_Insertion_sort();
165         I.Outputstring();
166     }
167     I.Fileout("HW1_out1.txt");
```

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

```
169     Sort M;
170     for(M.size = 10000; M.size<=500000 ; M.size += 10000)
171     //for(M.size = 100; M.size<=100000 ; M.size += 100)
172     {
173         cout<<"size = "<<M.size<<" ";
174         M.InitialA();
175         M.CountTime_Merge_sort();
176         M.Outputstring();
177     }
178     //M.Fileout("HW1_out2.txt");
179     M.Fileout("HW1_out2_test.txt");
```

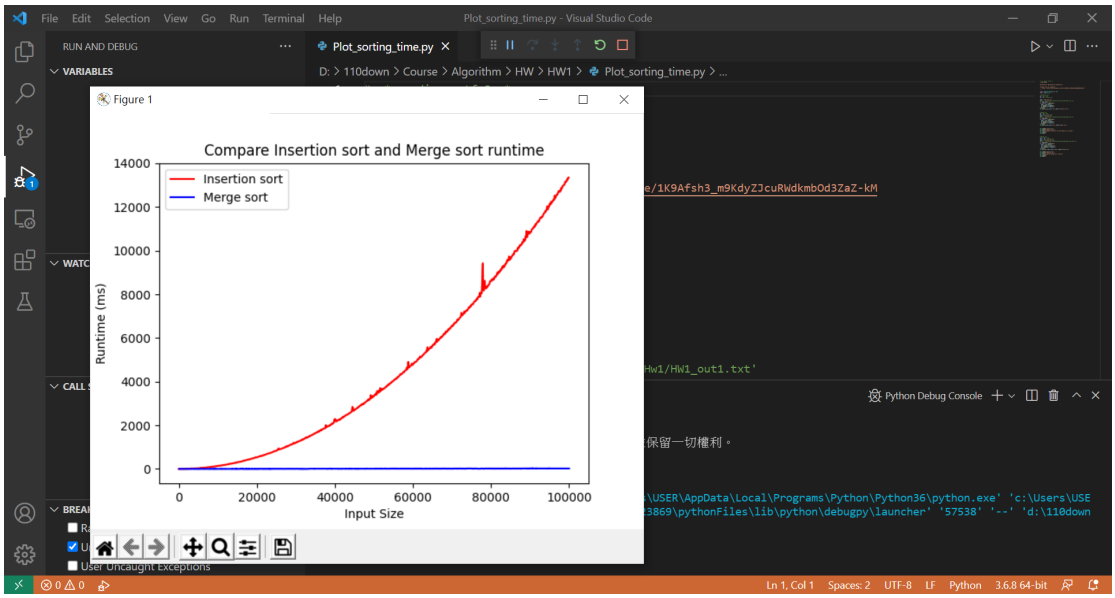
(二) 使用 Python 將兩種 Sorting 演算法的 Runtime 繪製成圖表

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

```
157 int main()
158 {
159     Sort I;
160     for(I.size = 100; I.size<=100000 ; I.size += 100)
161     {
162         cout<<"Size = "<<I.size<<endl;
163         I.InitialA();
164         I.CountTime_Insertion_sort();
165         I.Outputstring();
166     }
167     I.Fileout("HW1_out1.txt");
168
169     Sort M;
170     //for(M.size = 10000; M.size<=500000 ; M.size += 10000) //for HW1_out2_test.txt
171     for(M.size = 100; M.size<=100000 ; M.size += 100)
172     {
173         cout<<"size = "<<M.size<<" ";
174         M.InitialA();
175         M.CountTime_Merge_sort();
176         M.Outputstring();
177     }
178     M.Fileout("HW1_out2.txt");
179     //M.Fileout("HW1_out2_test.txt");
180
181     return 0;
182 }
```

2. 若有 python IDE，打開 Plot_sorting_time.py

3. 執行 Python 程式



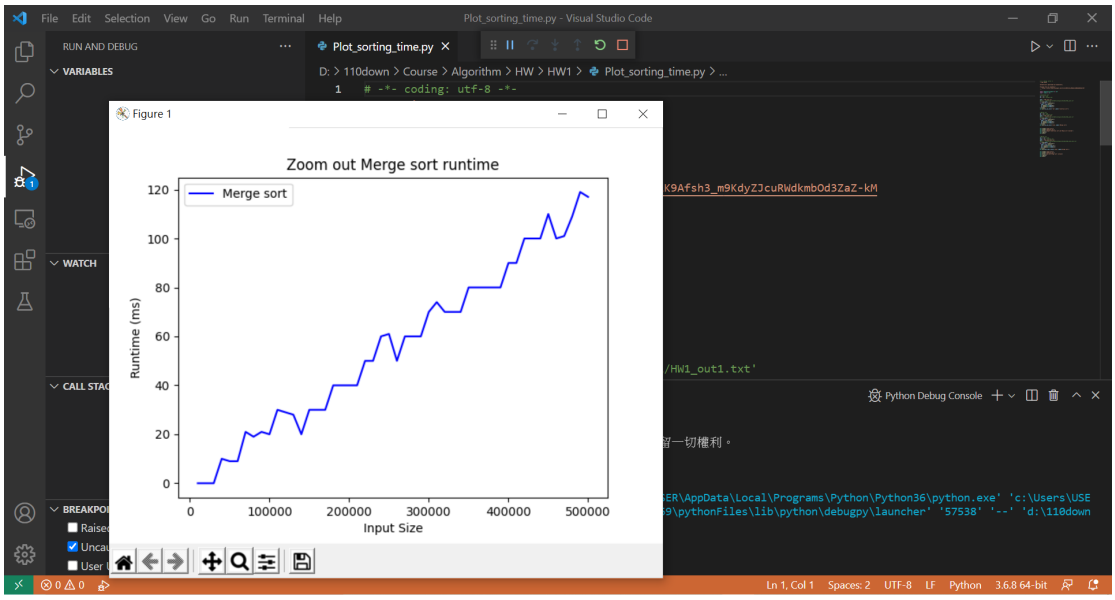
(三) 使用 Python 將 Merge sort 在 Input size 超過 10000 的 Runtime 繪製成圖表

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

```
169     Sort M;
170     for(M.size = 10000; M.size<=500000 ; M.size += 10000) //for HW1_out2_test.tx
171     //for(M.size = 100; M.size<=100000 ; M.size += 100)
172     {
173         cout<<"size = "<<M.size<<" ";
174         M.InitialA();
175         M.CountTime_Merge_sort();
176         M.Outputstring();
177     }
178     //M.Fileout("HW1_out2.txt");
179     M.Fileout("HW1_out2_test.txt");
```

2. 若有 python IDE，打開 Plot_sorting_time.py

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。