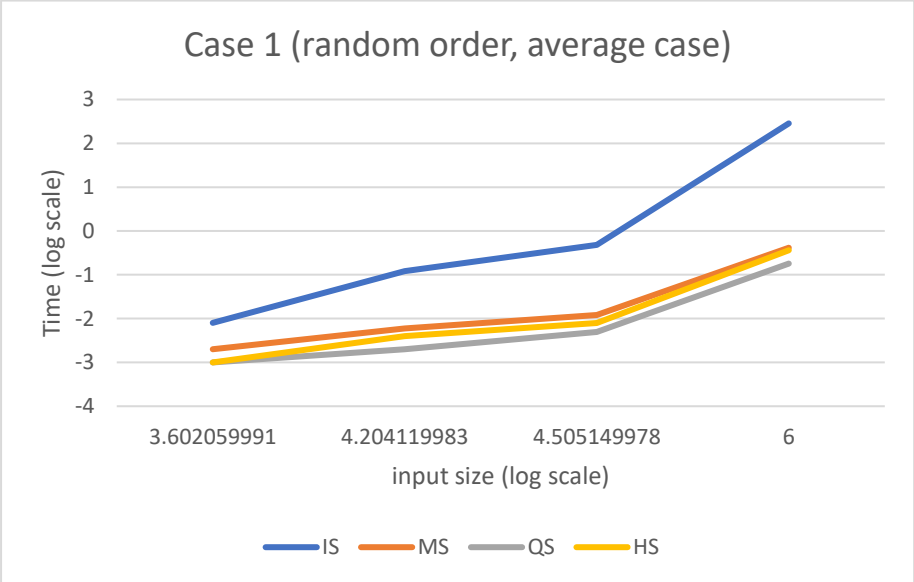
Algorithm PA1 Report

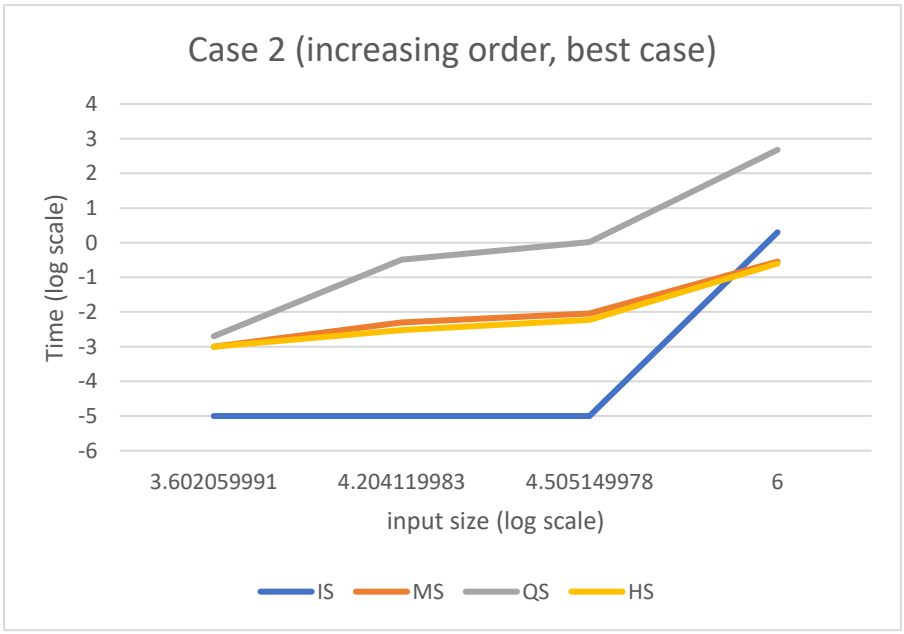
b0670264 / 會計五 / 林聖硯
EDA Union: Port 40051

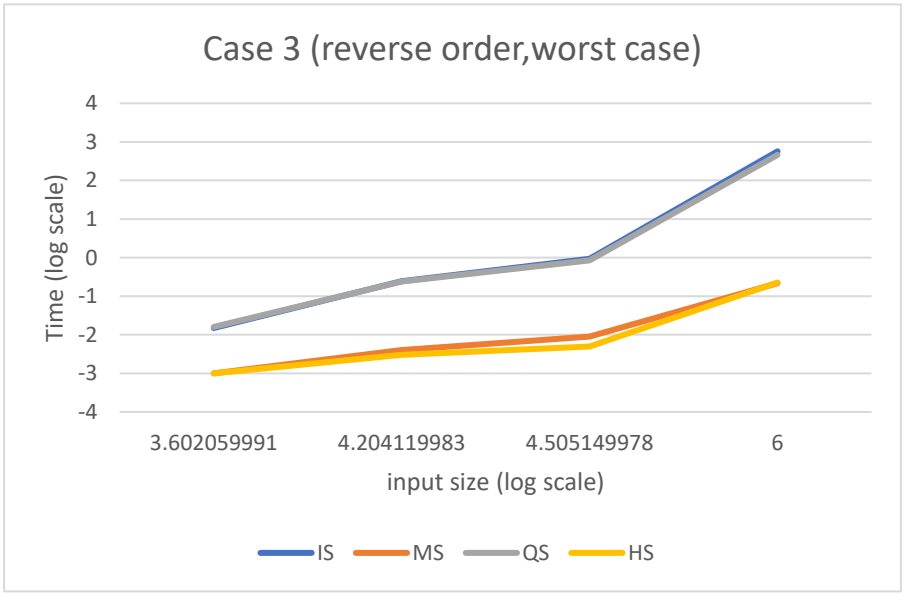
一、不同排序方法對上不同大小資料之 runtime

Input size	IS		MS		QS		HS	
	CPU time (s)	Memory (KB)	CPU time (s)	Memory (KB)	CPU time (s)	Memory (KB)	CPU time (s)	Memory (KB)
4000.case2	0	12500	0.001	12500	0.0019997	12616	0.001	12500
4000.case3	0.014998	12500	0.001	12500	0.015998	12520	0.000999	12500
4000.case1	0.007999	12500	0.002	12500	0.001	12500	0.001	12500
16000.case2	0	12648	0.005	12648	0.320951	13320	0.003	12648
16000.case3	0.242963	12648	0.004	12648	0.241963	12952	0.003	12648
16000.case1	0.120982	12648	0.005999	12648	0.002	12648	0.004	12648
32000.case2	0	12648	0.008998	12648	1.03584	14048	0.005999	12648
32000.case3	0.933857	12648	0.008999	12648	0.84987	13320	0.004999	12648
32000.case1	0.481927	12648	0.011998	12648	0.00499	12648	0.007999	12648
1000000.case2	2	18668	0.285957	18668	479.084	46172	0.250962	18668
1000000.case3	569.140	18668	0.216968	20524	461.334	34700	0.225966	18668
1000000.case1	284.600	18668	0.413937	18668	0.179973	18668	0.362945	18668

二、不同 Case 之下，不同演算法的表現畫成對數圖觀察
(如果 runtime 是 0，我這邊以 log 0.00001 = -5 表示，因為 log 裡面不能為 0)
所有的取 log 都是以 10 為底




Case 2 (increasing order, best case)
Time (log scale)
4
3
2
1
0
-1
-2
-3
-4
-5
-6
3.602059991
4.204119983
4.505149978
6
input size (log scale)
IS
MS
QS
HS


Case 3 (reverse order,worst case)
Time (log scale)
4
3
2
1
0
-1
-2
-3
-4
3.602059991
4.204119983
4.505149978
6
input size (log scale)
IS
MS
QS
HS

三、觀察與發現

由於以上的折線圖有經過 log 轉換，故我們必須將原本的時間複雜度函式經過 log 轉換才能判斷斜率合不合理。

$$O(\mathrm{N}^2) \rightarrow O(2log\mathrm{N})$$
$$O(NlogN) \rightarrow O\big(log(NlogN)\big) = O(log(N) + log(logN))$$
$$O(N) \rightarrow O(log(N))$$

(1) 同 case、不同演算法之比較

1. 在 Case 1(average case)內，MS、QS、HS 很明顯地表現比 IS 好也比較穩定，而 IS 在 4000 筆 input 的表現很明顯就與其他演算法差了一個層級，而到了 1,000,000 筆 input 時，其總耗費秒數更來到 284 秒。很明顯地驗證了 IS 在 average case 之時間複雜度為 O(N^2)，相較於 MS、QS、HS 在 average case 之時間複雜度為 O(N log N)。

在斜率的部分，QS、MS、HS 的時間複雜度皆為 O(NlogN)，經過 log 轉換後變成 O(logN+log(logN))；而 IS 的時間複雜度 O(N^2)，在經過 log 轉換之後為 O(2logN)。從以上的圖能夠看出 QS、MS、HS 的確在不同 input size 之下表現類似，斜率也接近 1，而 IS 之斜率相較其他三個演算法約為 2 倍，正好驗證了以上的公式。

2. 在 Case 2(best case)內，IS 在 4000、16,000、32,000 筆數之資料明顯地表現比其他演算法還要好上許多，正好驗證了他在 best case 的時間複雜度為 O(N)，而 QS 表現地相較其他演算法來的差。但在筆數來到一百萬時，MS 及 QS 相較比較穩定，但 IS 的秒數來到了兩秒，而 QS 的秒數卻飆升至 479 秒，與我在 QS 內 partition 的 implementation 細節有關，在(2)中詳述。

在斜率的部分，MS、HS 的折線圖正好反應了經過 log 轉換之後，時間複雜度為 O(logN+loglogN)的情形，斜率大約為 1(但不會剛好是 1，因為有後項的 loglogN)，而 QS 則因為我 implementation 的關係導致時間複雜度變成 O(N^2) (轉換後為 2logN)，從圖上也能看出其斜率相較 MS 與 HS 約為兩倍。

3. 在 Case3(worst case)內，不同資料筆數之下，IS 與 QS 之表現差不多，正好驗證了他們倆個在 worst case 的時間複雜度都是 O(N^2)；但相較 MS 與 HS 表現明顯差上許多，因為 MS 與 HS 在 worst case 的時間複雜度為 O(N logN)。尤其是在筆數來到一百萬時 IS 與 QS 之秒數分別來到了 569 以及 461 秒。

在斜率的部分，可以看出 IS 與 QS 大約為 MS 與 HS 的兩倍再少一點，正好驗證了各自的時間複雜度在經過 log 轉換後分別為 O(2logN)與 O(logN + loglogN)。

(2) 同演算法、不同 Case 之比較

1. IS 只有在 increasing order(Case2)時表現地比其他兩個 Case 還要好，顯示他在 best case 時能夠有效地處理雜亂的資料，但若資料確定很亂，選擇 IS 就沒那麼有效率。

2. MS 以及 HS 在不同 Case 之下表現，在樣本大小增加時，所耗費的時間也不會有明顯地上升，表現得非常穩定，正驗證了他們不論在任何情況之下，時間複雜度皆為 O(N log N)。也因為我這邊 MS implement 的是 in-place 的版本，所以 time complexity 只需要 O(N logN)，跟 PA0 的做法比起來快上許多，在 space complexity 的表現上也比較好。

3. QS 中，我實作 pivot 的方法為老師投影片上的做法(將最右邊的數字視為 pivot、最後再安插進 array 中)。在 case2(increasing order)中，用此方法實作，每跑一次迴圈就需要將 pivot 的位置往後移一個，但其實最右邊的數字要安插的地方就正好時在最右邊，所以每次在找 pivot 位置的時候都在做白工。而且在這個情況下，partition 總是把一個 input size 為 n 的問題，分成 n-1:0 的問題，導致 QS 的表現明顯都比其他三個演算法差。在 case3(reversing order)中，情況與上述類似，只是子問題變成 0:n-1 的比例，導致在這兩種情形之下，QS 時間複雜度為 O(N^2)，比其他演算法慢上許多。但在 case1 (average case)中，QS 反而能有效地排列 array，而不論在哪種資料筆數之下，都能夠勝過其他三種演算法。

(3) 總結

如果在已知資料不會太混亂的狀況下，我會選擇以 IS 作為我的排序演算法，但在其他情況，我會選擇相較穩定的 MS 或 HS 作為我的排序演算法。

QS/MS/HS/IS 之演算法比較圖

	Quick Sort	**Merge Sort**	**Heap Sort**	**Insertion Sort**
best case	$N \log N$	$N \log N$	$N \log N$	N
average case	$N \log N$	$N \log N$	$N \log N$	N^2
worst case	N^2	$N \log N$	$N \log N$	N^2