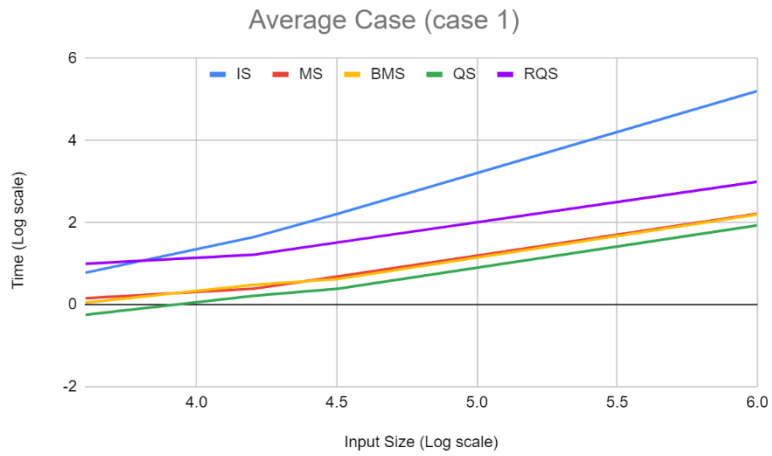
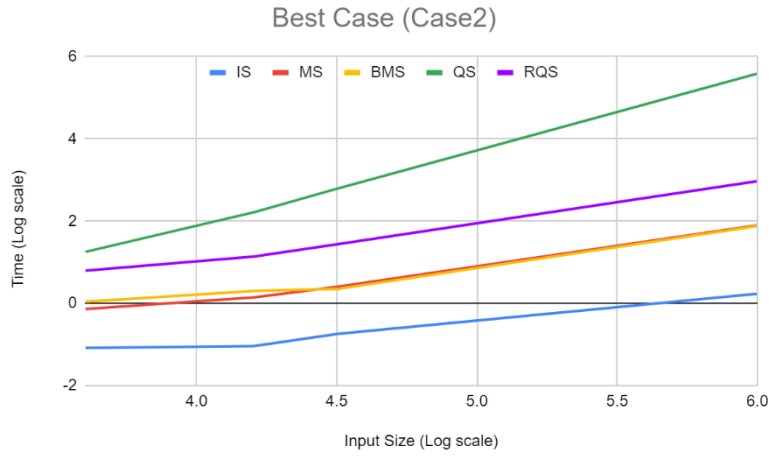
Programming Assignment #1 Report					
姓名	劉又慈	學號	B11901193	系級	電機三

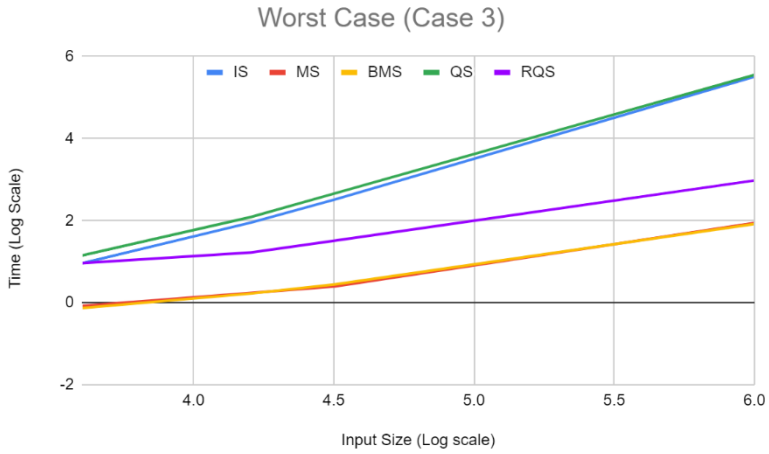
1. Table of runtime and memory usage of five sorting algorithms. (Data obtain from EDA union lab machines, port 40056)

Input size	IS		MS		BMS		QS		RQS	
	CPU time(ms)	Memory(KB)	CPU time(ms)	Memory(KB)	CPU time(ms)	Memory(KB)	CPU time(ms)	Memory(KB)	CPU time(ms)	Memory(KB)
4000.case2	0.082	5904	0.726	5904	1.092	5904	17.699	6032	6.212	5904
4000.case3	9.121	5904	0.832	5904	0.741	5904	14.128	5904	9.162	5904
4000.case1	5.963	5904	1.434	5904	1.116	5904	0.563	5904	9.921	5904
16000.case2	0.091	6056	1.386	6056	1.997	6056	163.387	6924	13.692	6056
16000.case3	89.431	6056	1.723	6056	1.668	6056	122.19	6432	16.75	6056
16000.case1	44.468	6056	2.464	6056	3.043	6056	1.64	6056	16.548	6056
32000.case2	0.182	6188	2.547	6188	2.267	6188	624.566	7968	27.504	6188
32000.case3	328.744	6188	2.494	6188	2.805	6188	463.086	6996	32.488	6188
32000.case1	164.872	6188	4.88	6188	4.253	6188	2.45	6188	33.057	6188
1000000.case2	1.703	12144	79.12	14004	76.216	14000	387249	49464	938.947	12144
1000000.case3	321517	12144	87.372	14004	82.378	14000	356177	34176	950.086	12144
1000000.case1	160851	12144	164.952	14004	158.305	14000	85.883	12144	993.937	12144

Trendline plot with slope calculation for different cases (view more detail in https://docs.google.com/spreadsheets/d/1q3VzviFzNhhS48JpJBPPP47AfD8hYCGcsBBV8BefPoI/edit?usp=sharing)：







Slope calculation(使用 google sheet 趨勢線公式功能)：

	IS	MS	BMS	QS	RQS
Case 1	1.88	0.899	0.915	0.925	0.873
Case 2	0.587	0.882	0.802	1.83	0.937
Case 3	1.92	0.872	0.877	1.86	0.875

Compare your slope with the complexity in the textbook. Please explain why or why not they match：

- Insertion Sort
  課本中 IS 的 time complexity 為$O(n^2)$，將n^2 取log後得到 2，和表格中 Insertion Sort 的斜率相符。而在 case 2 時，由於資料已經排序完畢，因此 IS 運算速度較快(無須做任何 swap)，有較小的斜率，理論上在 case 2 的情況斜率應為 1(因為 IS best case time complexity 為$O(n)$)，推測是因為在資料量較少時主要影響運算速度的可能不是 n 而是由常數項決定，使得斜率與理論值不相符。
- Merge Sort
  課本中 MS 的 time complexity 為$\Theta(nlgn)$，將$nlgn$ 取log後無法得到為定值的斜率(會跟資料量有關)，因此我們藉由 MS 的成長趨勢與其他演算法比較，可以發現與課本中的描述符合。
- Bottom up Merge Sort
  課本中沒有詳細解釋 BMS，但由上表可以發現他的表現和 MS 差不多。
- Quick Sort
  課本中 QS 的 time complexity 為 average case $O(nlgn)$，可由 case 1 的趨勢圖看出他和 MS、BMS 的表現相近，這與課本的描述相符。而 case 2 和 case 3 即為 QS 的 worst case，課本中有提及 QS worst case time complexity 為$O(n^2)$，n^2 取log後得到 2，這與表格中得到的斜率數值相符合。
- Random Quick Sort

RQS 可以有效的避免發生 QS worst case 的情況，並且我們可以從斜率表看出其成長趨勢和 MS 相似，故其 time complexity 亦為$\Theta(nlgn)$，這與課本敘述相同。

2. Comparison between MS and BMS, including runtime difference and explanation.

Input size	MS		BMS	
	CPU time(ms)	Memory(KB)	CPU time(ms)	Memory(KB)
4000.case2	0.726	5904	1.092	5904
4000.case3	0.832	5904	0.741	5904
4000.case1	1.434	5904	1.116	5904
16000.case2	1.386	6056	1.997	6056
16000.case3	1.723	6056	1.668	6056
16000.case1	2.464	6056	3.043	6056
32000.case2	2.547	6188	2.267	6188
32000.case3	2.494	6188	2.805	6188
32000.case1	4.88	6188	4.253	6188
1000000.case2	79.12	14004	76.216	14000
1000000.case3	87.372	14004	82.378	14000
1000000.case1	164.952	14004	158.305	14000

由上圖比較 MS 和 BMS 在不同測資的 running time 可發現，在較小的資料量(4000~16000) MS 的運算時間較短，但在資料量較大的情況(32000~1000000) BMS 的運算速度較 MS 更快，但運算時間的差異不大，兩者的表現非常接近。

推測兩種演算法運算時間有些微差異的原因，由於 MS 使用 recursive method，可能在執行過程中產生 overhead 為了維持遞迴的 call of stack，而 BMS 採用 iterative method，可以減少 overhead 的產生。此外，MS 每次遞迴切分 input data 的作法可能造成較差的 cache locality (主要是 spatial locality)，這使 MS 的執行效率相比於 BMS 較差，以上理由可能是 BMS 在大資料量時表現比較好的原因。

3. Comparison between QS and RQS, including runtime difference and explanation.

Input size	QS		RQS	
	CPU time(ms)	Memory(KB)	CPU time(ms)	Memory(KB)
4000.case2	17.699	6032	6.212	5904
4000.case3	14.128	5904	9.162	5904
4000.case1	0.563	5904	9.921	5904
16000.case2	163.387	6924	13.692	6056
16000.case3	122.19	6432	16.75	6056
16000.case1	1.64	6056	16.548	6056
32000.case2	624.566	7968	27.504	6188
32000.case3	463.086	6996	32.488	6188
32000.case1	2.45	6188	33.057	6188
1000000.case2	387249	49464	938.947	12144
1000000.case3	356177	34176	950.086	12144
1000000.case1	85.883	12144	993.937	12144

除了 average case 的 case 1，RQS 的執行速度都比 QS 快上不少，原因是 case 2 和 case 3 都是已排序好的資料，QS 取 data array 中最右邊的那個數作為 pivot 時，無法在每一次 partition 將 data array 拆成平均的兩半進行下一輪的遞迴，因此每次遞迴都幾乎要遍歷 data 一次，因此運算效率非常的差，為$O(n^2)$。而 RQS 隨機取 pivot 的作法可以大幅減少每次 partition 時，拆成兩半的資料量不平均的情況，因此有更好的運算效率。

本次作業的 RQS 我使用以下的程式碼實作：

```
int SortTool::RandomizedPartition(vector<int>& data, int low, int high){
        srand(time(NULL));
        int random = low + rand() % (high - low + 1);
        swap(data[random], data[high]);
        return Partition(data, low, high);
}
```

理論上 QS 和 RQS 在 case 1 的運算速度應該非常相近，因為兩者的 pivot 都是隨機選取的，比較不易出現 partition 時分配不平均的情況，但由上方比較表格可以發現，RQS 在 case 1 的運算速度比 QS 差上許多，這是因為我使用 srand 函式取每一次的 pivot，而這個函式的運算在每一次遞迴都會做一次，因此 RQS 比 QS 找 pivot 需要更多的時間。(更多比較將於 4.討論)

4. Data structure used and other findings in this programming assignment.
   - Data structure used
     本次作業中使用 vector 實作資料結構 array。
   - Other findings
     當 RQS 的實作方式改為直接以 data 的中點作為 pivot，以下是實作程式碼：

```
int SortTool::RandomizedPartition(vector<int>& data, int low, int high){
        int random = (low + high)/2;
        swap(data[random], data[high]);
        return Partition(data, low, high);
}
```

     我們可以發現以此方式執行的 RQS 在 case 1 會和 QS 有相似的執行效率，因為 pivot 都是隨機選取的，又因為減少了使用 srand function 找 pivot 的時間，所以可以運算的比原本的 RQS 更快。而 case 2 和 case 3 中，因為資料都排序好的，因此取中點作為 pivot 可以在每一次的 partition 最平均的將資料分成兩部分再進入下一次遞迴，因此執行的

速度相較 QS 快了不少。以下數據取得自 EDA union lab port 40056。

Input size	QS		RQS		RQS (choosing midpoint as pivot)	
	CPU time(ms)	Memory(KB)	CPU time(ms)	Memory(KB)	CPU time(ms)	Memory(KB)
4000.case2	17.699	6032	6.212	5904	0.263	5904
4000.case3	14.128	5904	9.162	5904	0.223	5904
4000.case1	0.563	5904	9.921	5904	0.54	5904
16000.case2	163.387	6924	13.692	6056	0.898	6056
16000.case3	122.19	6432	16.75	6056	0.532	6056
16000.case1	1.64	6056	16.548	6056	2.487	6056
32000.case2	624.566	7968	27.504	6188	0.577	6188
32000.case3	463.086	6996	32.488	6188	1.122	6188
32000.case1	2.45	6188	33.057	6188	2.286	6188
1000000.case2	387249	49464	938.947	12144	26.978	12144
1000000.case3	356177	34176	950.086	12144	33.926	12144
1000000.case1	85.883	12144	993.937	12144	83.668	12144

參考資料：

https://ithelp.ithome.com.tw/m/articles/10263066

https://zh.wikipedia.org/zh-tw/CPU%E7%BC%93%E5%AD%98

https://stackoverflow.com/questions/10153393/mergesort-is-bottom-up-faster-than-top-down