Running Time Table (using port:40056 edaU6)

Input size	IS		MS		QS		RQS		HS	
	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	0.141	5904	0.96	5904	19.486	6032	21.688	5904	0.792	5904
4000.case3	9.908	5904	1.171	5904	16.189	5908	22.034	5904	0.822	5904
4000.case1	7.195	5904	1.722	5904	0.926	5904	22.665	5904	0.88	5904
16000.case2	0.123	6056	1.884	6056	205.701	6936	63.534	6056	1.533	6056
16000.case3	64.596	6056	2.542	6056	163.426	6428	63.528	6056	2.294	6056
16000.case1	34.328	6056	3.941	6056	1.727	6056	64.17	6056	2.446	6056
32000.case2	0.164	6188	3.146	6188	805.991	8004	122.464	6188	1.948	6188
32000.case3	241.606	6188	2.619	6188	634.409	6988	122.741	6188	1.809	6188
32000.case1	122.961	6188	4.111	6188	3.651	6188	126.755	6188	3.567	6188
1000000.case2	2.27	12144	76.737	14004	time-out	time-out	3833.32	12144	75.802	12144
1000000.case3	254995	12144	81.875	14004	time-out	time-out	3832.61	12144	74.177	12144
1000000.case1	125687	12144	150.667	14004	98.183	12144	3882.25	12144	138.225	12144

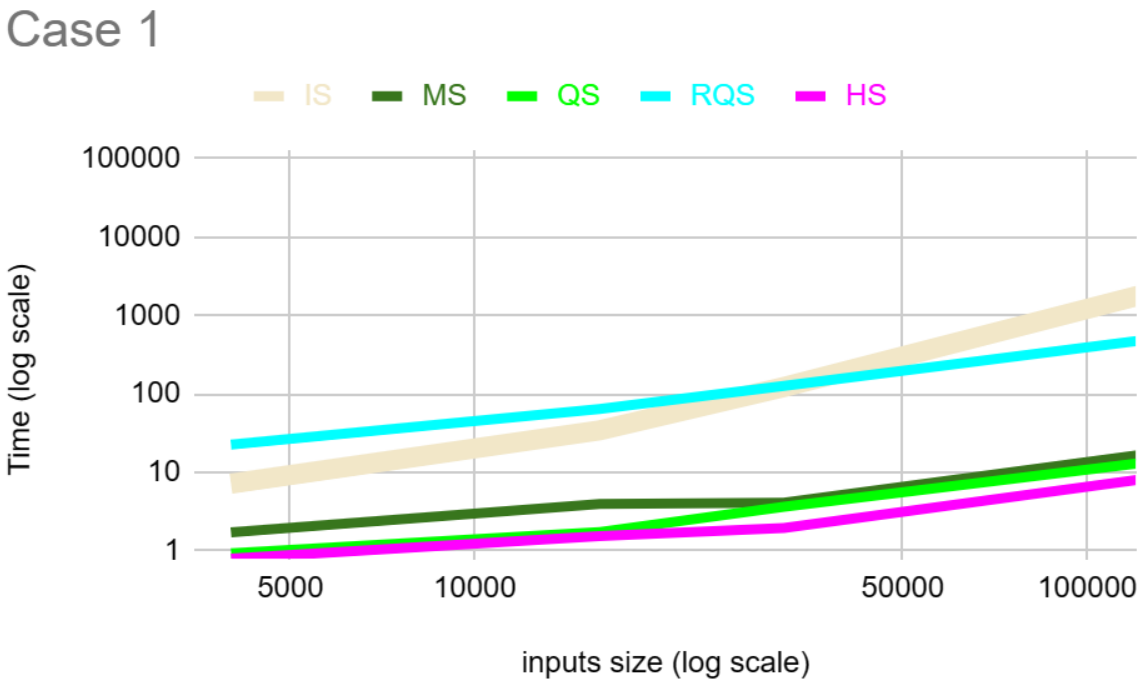
Case 1 / Case 2 / Case 3 separared

case1	QS	HS	MS	RQS	IS
4000	0.926	0.88	1.722	22.665	7.195
16000	1.727	2.446	3.941	64.17	34.328
32000	3.651	3.567	4.111	126.755	122.961
1000000	98.183	138.225	150.667	3882.25	125687

case2	IS	HS	MS	RQS	QS
4000	0.141	0.792	0.96	21.688	19.486
16000	0.123	1.533	1.884	63.534	205.701
32000	0.164	1.948	3.146	122.464	805.991
1000000	2.27	75.802	76.737	3833.32	—---

case3	HS	MS	RQS	IS	QS
4000	0.822	1.171	22.034	9.908	16.189
16000	2.294	2.542	63.528	64.596	163.426
32000	1.809	2.619	122.741	241.606	634.409
1000000	74.177	81.875	3832.61	254995	---------

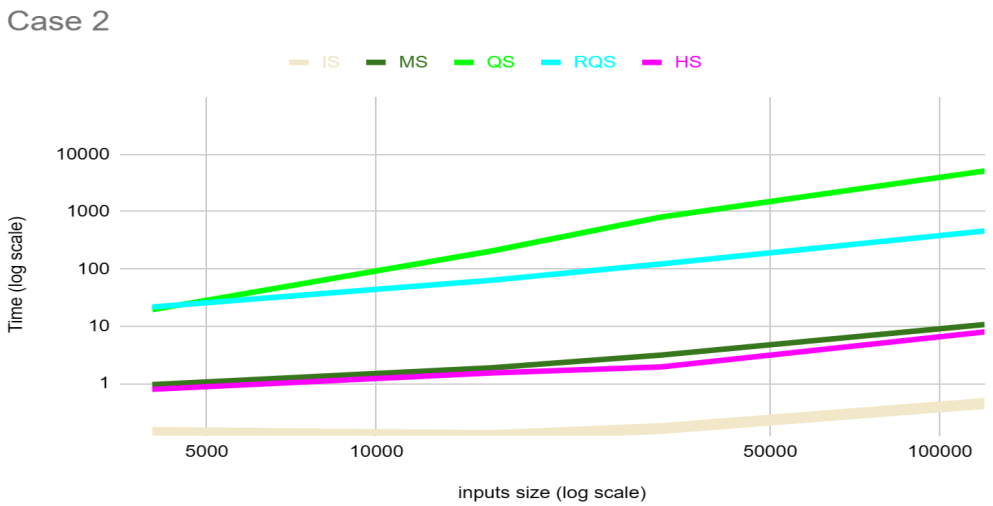
Ploting



觀察到

1. MS QS HS 差不多小 (理論約nlogn)
2. HS 最好
3. RQS IS 最慢 (IS 除了best case 都很慢)

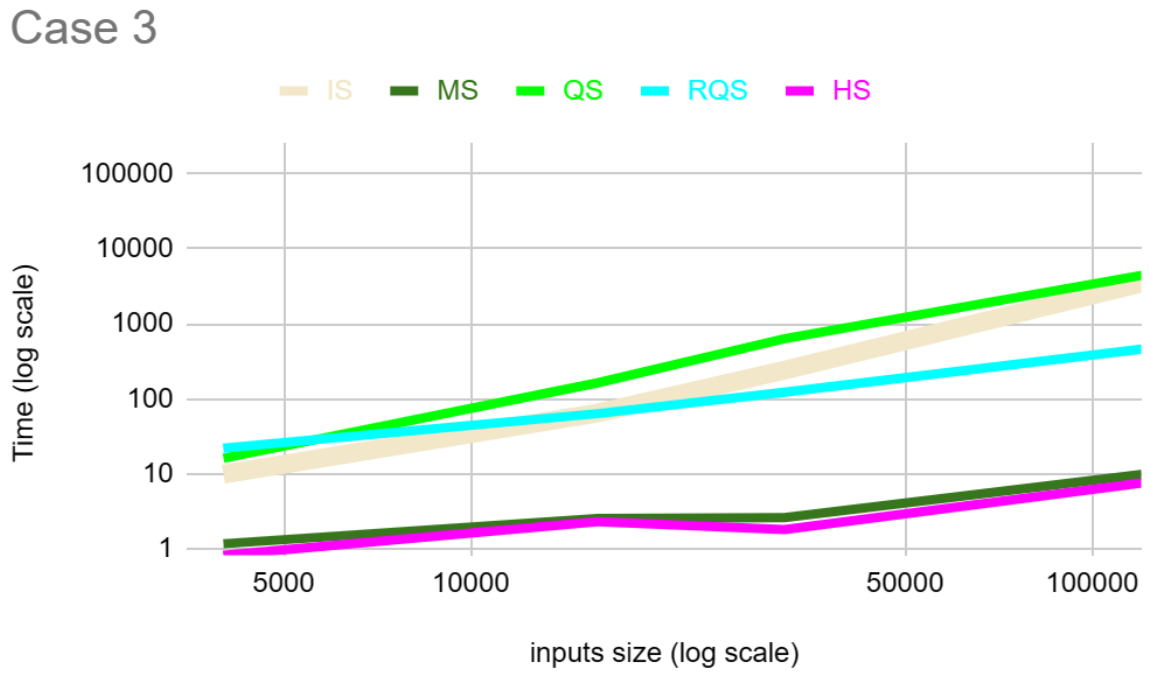
在avg時



觀察到

1. IS 在best case中是最快到 (因為不用交換)
2. HS 還是有第二 幾乎和merge同條
3. RQS QS 家族的都比較慢 (可能是因為即使分好了 也要一直partition 一直partition 遞迴下去)

在best時



觀察到

在worst時

1. IS 是最慢的 (因為要一直交換到整個數列翻轉)
2. RQS QS 緊跟在IS後, 甚至超過 (可能因為每次分割都非常不均衡導致)
3. RQS QS 家族的都比較慢 (可能是因為即使分好了 也要一直partition 一直partition 遞迴下去)

analysis

1. why linear for some cases? :

可能是因為輸入的性質。像是對於近乎有序的列表，IS就會是線性的。另一個原因可能是函數調用、就是初始化和其他常數時間操作的開銷，這可能會掩蓋實際複雜度在較小輸入時的效果。

2. what is the slope supposed to be?

$$\frac{\log(y)}{\log(x)} = \log_x(y)$$ 就是y是x的幾次

例如斜率=2 代表 $\log_x(y) = 2$ => $y=x^2$
又例如 斜率 = 1/2 => 複雜度是 根號x

3. Quicksort may have the same complexity with insertion sort, why?

這是因為如果在快速排序中經常選擇到最小或最大的元素作為基準元素，那麼分割會非常不均勻。在最壞的情況下，這會使快速排序的時間複雜度降至O(n^2)。這是因為每次分割只能將數列減少一個元素，所以需要進行 n 次分割，每次分割都要掃描 n 個元素

How to solve it?

解決方法就是利用RQS，其中的R是隨機的意思，通過隨機選擇privot，我們可以期望分割是較為均勻的，從而使得平均時間複雜度維持在O(nlogn)。
而且從圖形來看，RQS 幾乎不受case1 case2 case3的影響 都是在很穩定的時間內完成(如下圖 在case case 間 幾乎沒什麼波動)

RQS	case1	caes2	case3
4000	21.688	22.665	22.034
8000	63.534	64.17	63.528
32000	122.464	126.755	122.741
10000000	3833.32	3833.32	3832.61

4. Comparing the difference between the QS and RQS:

快速排序是確定性的(即，給定相同的輸入，它將始終具有相同的行為)，而隨機快速排序通過隨機基準選擇引入了概率性組件。
快速排序在特定輸入(如排序或近乎排序的列表)上更容易達到其O(n^2) 的最壞情況時間複雜度。隨機快速排序通過隨機選擇基準元素來減少這種情況，使最壞情況出現的可能性較小。

5. Randomized quicksort may have the same complexity with insertion sort, why ?

儘管隨機快速排序中的基準元素選擇減少了遇到最壞情況的可能性，但它並沒有消除這種可能性。仍然存在非零的概率，使得隨機快速排序可能連續選擇不好的基準，導致O(^2)的時間複雜度。
不然還有一種可能就是，如果你的inputs 很少，那取得隨機數的過程就不容忽瑞