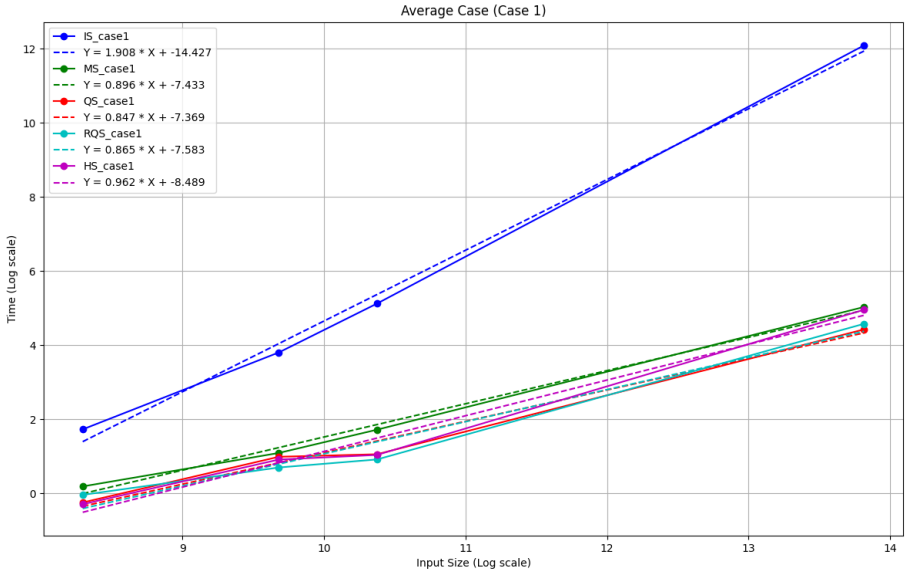
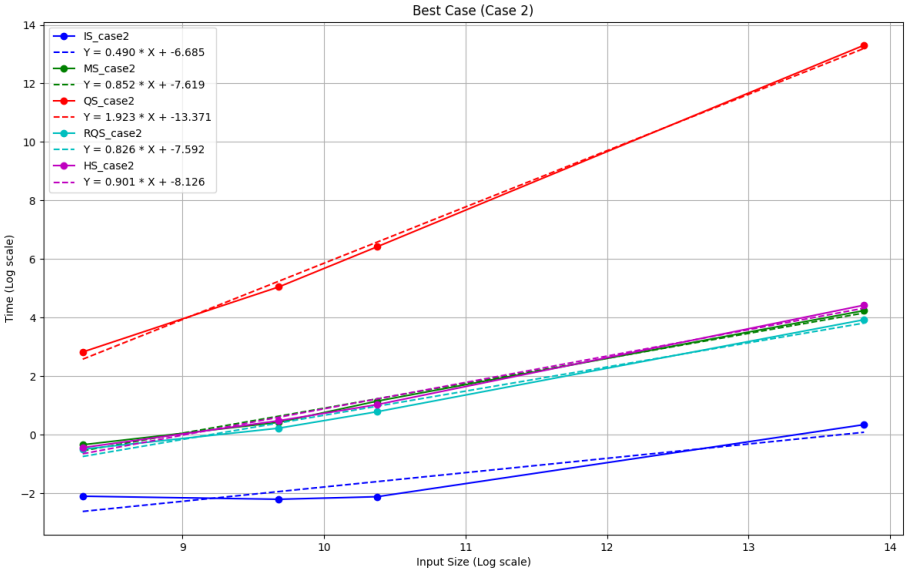
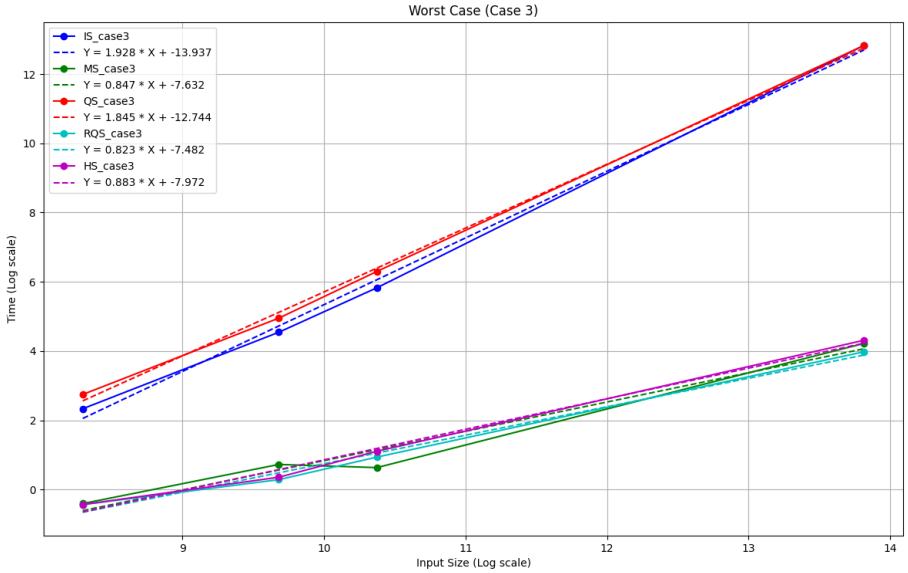
PA1 Report

B09502158 詹宜昇

input size	IS	IS	MS	MS	QS	QS	RQS	RQS	HS	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	0.122ms	5904kb	0.709ms	6040kb	16.849ms	5968kb	0.609ms	5904kb	0.642ms	5904kb
4000.case3	10.306ms	5904kb	0.666ms	6040kb	15.58ms	5904kb	0.646ms	5904kb	0.648ms	5904kb
4000.case1	5.642ms	5904kb	1.211ms	6040kb	0.785ms	5904kb	0.964ms	5904kb	0.749ms	5904kb
16000.case2	0.11ms	6056kb	1.526ms	6056kb	155.85ms	6694kb	1.249ms	6056kb	1.612ms	6056kb
16000.case3	94.025ms	6056kb	2.063ms	6056kb	141.342ms	6300kb	1.33ms	6056kb	1.427ms	6056kb
16000.case1	44.94ms	6056kb	2.987ms	6056kb	2.693ms	6056kb	2.018ms	6056kb	2.5ms	6056kb
32000.case2	0.12ms	6188kb	3.142ms	6188kb	612.151ms	7504kb	2.188ms	6188kb	2.79ms	6188kb
32000.case3	339.849ms	6188kb	1.884ms	6188kb	544.611ms	6740kb	2.553ms	6188kb	3.022ms	6188kb
32000.case1	167.714ms	6188kb	5.591ms	6188kb	2.883ms	6188kb	2.505ms	6188kb	2.831ms	6188kb
1000000.case	1.408ms	12144kb	69.308ms	15956kb	600255ms	56848kb	50.793ms	12144kb	83.498ms	12144kb
1000000.case	372999ms	12144kb	68.0693ms	15956kb	372093ms	27252kb	53.47ms	12144kb	74.575ms	12144kb
1000000.case	177469ms	12144kb	153.148ms	15956kb	83.51ms	12144kb	97.293ms	12144kb	142.144ms	12144kb
edaunion	U10									
port	40060									







Insertion sort's time complexity is linear in best case where the input array is already sorted. Since there is no inversion in the input array, it doesn't need any swap during execution, therefore it simply loops through the array once, which is why the time complexity is linear to the input size.

The slope of each case is already shown in the plots.

Quicksort have the same time complexity with insertion sort in the worst case, where quicksort cannot effectively partition the original array into two subarray. In such cases it has the time complexity of O(n^2) which is same as insertion sort. To solve

the problem, we can randomly select the pivot which is the algorithm Randomized Quicksort, by doing so, the pivots can effectively partition the array on average and it has the time complexity of O(n*logn).

As mentioned in previous part, the main difference between QS and RQS is their partition method, which it results to their performance difference when encounter a sorted or reversely sorted array, where QS cannot effectively partition the array.

RQS may perform poorly when the input array's elements are all same, in such condition it cannot partition the input array into two balanced subarray therefore it will have the same time complexity of insertion sort which is O(n^2).