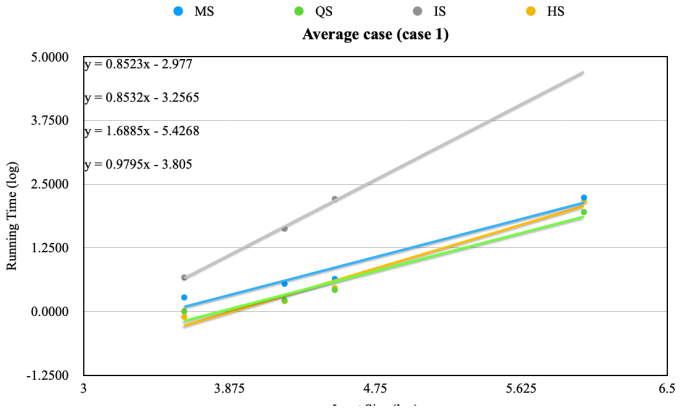
林桓鈺 B10502013

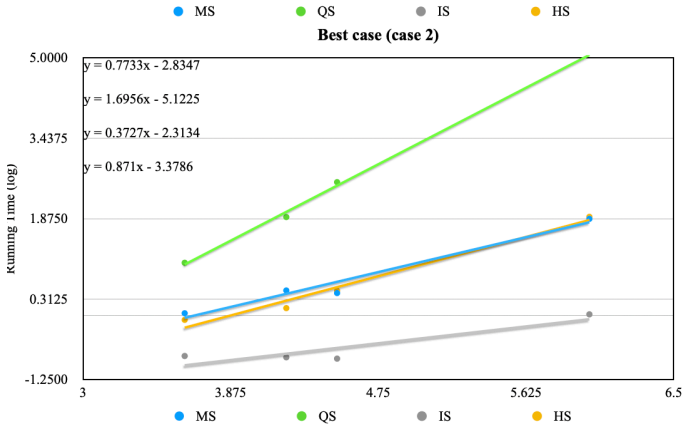
Algorithm-PA1

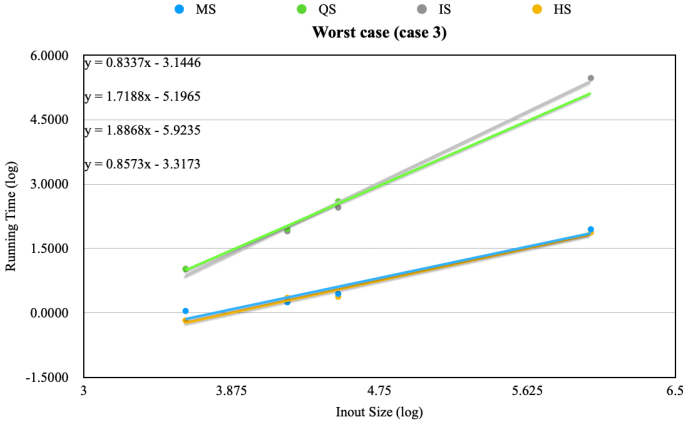
CPU Time and Memory

using EDA union lab machines

Input size	IS		MS		QS		HS	
	CPU time (ms)	Memory (KB)	CPU time (ms)	Memory (KB)	CPU time (ms)	Memory (KB)	CPU time (ms)	Memory (KB)
4000.case2	0.162	5908	1.093	5908	10.435	6036	0.814	5908
4000.case3	10.676	5908	1.11	5908	10.49	6036	0.671	5908
4000.case1	4.68	5908	1.897	5908	1.005	5908	0.787	5908
16000.case2	0.153	6060	3.024	6060	80.611	6940	1.376	6060
16000.case3	79.9	6060	1.773	6060	89.44	6928	2.249	6060
16000.case1	42.333	6060	3.498	6060	1.656	6060	1.599	6060
32000.case2	0.144	6192	2.697	6192	382.858	8004	2.991	6192
32000.case3	284.764	6192	2.789	6192	397.216	7976	2.383	6192
32000.case1	162.234	6192	4.376	6192	2.654	6192	2.891	6192
1000000.case2	1.042	12148	74.319	14008	297808	33408	81.296	12148
1000000.case3	294943	12144	88.706	14008	287637	24656	74.719	12148
1000000.case1	146660	12144	172.226	140008	89.6	12148	142.645	12148







Findings:

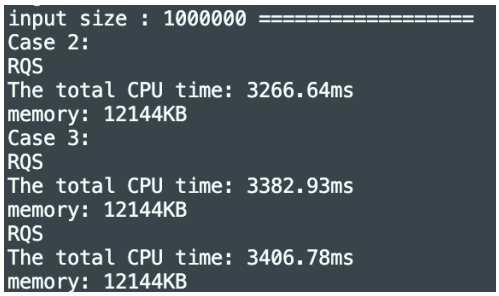
I. In average case (case 1):

A. The time tendencies of MergeSort (MS), QuickSort (QS), and HeapSort (HS) are very similar. I believe the main reason for this similarity is that in the average case, the time complexity of all three algorithms is $O(nlogn)$, while the time complexity of Insertion Sort (IS) is $O(n^2)$. This leads to differences in their time tendencies.

B. The memory usage of these four sorting methods is generally similar, except that when the input size is 100,000, QuickSort and HeapSort use less memory.

II. In best case (case 2):

A. IS requires the least amount of time, with MS and HS following closely behind, while QS consumes the most time. However, when analyzing based on time complexity, IS indeed should take less time because its time complexity is $O(n)$, whereas MS, HS, and QS should all have time complexities of $O(nlogn)$. I'm not entirely sure why this is happening; I consider that the input data to be sorted is already sorted or nearly sorted, resulting in each partitioning step only dividing the array into a smaller part and a larger part, thus leading to a decline in Quick Sort's performance. it could be due to errors in my code or other reasons. I'm unsure why QS takes so much more time compared to MS and HS. If QS is modified to RQS (Randomized Quick Sort), this problem can be resolved. However, in the average case, it may take more time. ◦



III. In worst case (case 3):

A. From the graph, it can be observed that QS and IS exhibit similar time tendencies, while their time complexities are $O(n^2)$. Similarly, MS and HS also show similar time tendencies, with time complexities of $O(nlogn)$. This aligns with the analysis based on time complexity.