PA1 Report

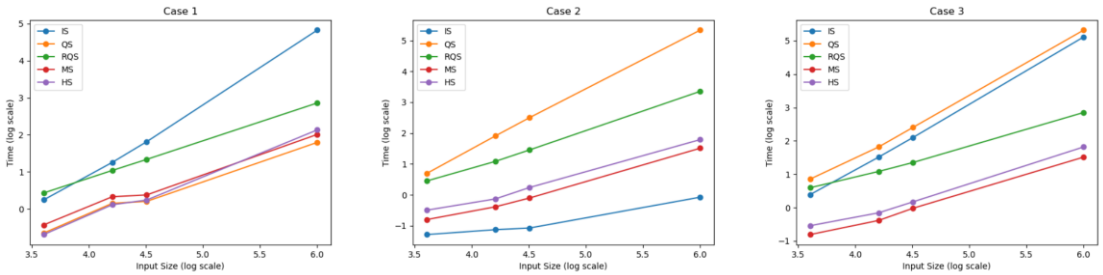
學號 ：B10502076 姓名 ： 金家逸 系級 ： 電機三

Algorithm implementation is using the textbook for reference

data structures used in my program is only array.

The data of the following table is running on my local machine

Input size	IS		MS		QS		RQS		HS	
	CPU time (s)	Memory (KB)	CPU time (s)	Memory (KB)	CPU time (s)	Memory (KB)	CPU time (s)	Memory (KB)	CPU time (s)	Memory (KB)
4000.case.2	0.052	5904	0.159	5904	4.916	5968	2.821	5904	0.32	5904
4000.case.3	2.45	5904	0.156	5904	7.137	5904	3.945	5904	0.288	5904
4000.case.1	1.743	5904	0.368	5904	0.222	5904	2.684	5904	0.202	5904
16000.case.2	0.075	6056	0.412	6056	81.026	6684	12.179	6056	0.741	6056
16000.case.3	32.804	6056	0.415	6056	66.039	6308	12.195	6056	0.702	6056
16000.case.1	18.003	6056	2.125	6056	1.411	6056	10.899	6056	1.292	6056
32000.case.2	0.085	6188	0.796	6188	318.648	7480	28.701	6188	1.745	6188
32000.case.3	126.949	6188	0.961	6188	254.663	6748	22.608	6188	1.495	6188
32000.case.1	64.872	6188	2.378	6188	1.592	6188	21.634	6188	1.731	6188
1000000.case.2	0.845	12144	32.62	13876	217293	39592	2262.4	12144	61.887	12144
1000000.case.3	130479	12144	32.769	13880	209374	28120	711.405	12144	66.101	12144
1000000.case.1	66862.1	12144	102.281	13880	62.277	12144	721.871	12144	135.228	12144



Analyze the graph :

Algorithm	Runtime		
	Best case	Average case	Worst case
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$
Merge sort	$O(nlgn)$	$O(n)$	$O(nlgn)$
Quick sort	$O(nlgn)$	$O(nlgn)$	$O(n^2)$
Randomized Quick sort	--	$O(nlgn)$	--
Heap sort	$O(nlgn)$	$O(nlgn)$	$O(nlgn)$

For Merge sort and Heap sort, the complexity for all three cases is $O(nlgn)$, and we can observe that the running time and slopes in the three graphs are very close.

For Insertion sort, case 1 is the average case, case 3 is the worst case, and both have a complexity of $O(n^2)$. Therefore, the running time and slope are both high for these cases. However, case 2 is the best case with a complexity of $O(n)$, so the running time and slope are the lowest.

For Randomized Quick sort, all three cases are average cases with a complexity of $O(nlgn)$, and the slopes are similar to Heap sort and Merge sort. However, the

constant coefficient may be larger than that of Heap sort and Merge sort, resulting in a longer running time compared to the two.

For Quick sort, case 1 is the average case with a complexity of $O(nlgn)$, so the trend is similar to Merge sort and Heap sort. It can be observed that as the input size increases, Quick sort's running time is slightly faster than Heap sort and Merge sort. However, for case 2 and case 3, which are worst-case scenarios, their complexity is $O(n^2)$. In case 2, Quick sort's running time and slope are the highest, while in case 3, Quick sort's running time and slope are similar to Insertion sort.

Explain:

1. why linear for some cases?
   Some complexity is $O(n^2)$. After taking log for input size and running time, the curve is linear. As for $O(nlgn)$, after taking log, loglog(n) term is very small, which can be neglected.

2. Quicksort may have the same complexity with insertion sort , why? How to solve it? :
   Quicksort's performance heavily depends on the choice of pivot elements. If the pivot selection strategy consistently chooses poor pivots (selecting the minimum or maximum element), quicksort can degrade to $O(n^2)$ complexity.

   Solution: Implement a randomized pivot selection strategy, such as selecting a pivot element at random, to mitigate the chances of worst-case behavior.

3. Comparing the difference between the QS and RQS
   (1) QS:
   the pivot element is typically chosen as either the first, last, or middle element of the array. This deterministic pivot selection can lead to worst-case behavior when sorting already sorted or nearly sorted data. In the worst-case scenario, when poor pivot elements are consistently chosen, it can degrade to $O(n^2)$ time complexity. However, on average, it has an $O(nlgn)$ time complexity.
   (2) RQS:

RQS improves the pivot selection by choosing a pivot element at random from the array. This randomness helps mitigate the worst-case scenarios associated with deterministic pivot selection. provide an average-case time complexity of $O(nlgn)$by reducing the likelihood of encountering worst-case scenarios.

4. Randomized quicksort may have the same complexity with insertion sort, why?
   (1) When the input size is very small, the constant factors in the time complexity of both RQS and IS can make them appear to have similar performance. In such cases, the overhead of QS's partitioning and recursive calls might outweigh its advantages.
   (2) If the input data is nearly sorted, RQS can still perform additional comparisons and swaps, making it less efficient than expected. In such cases, the average-case time complexity increases.