CO322: Data Structures and Algorithms

Sorting Algorithms

Lab – 01

E/19/306

Rajakaruna M.M.P.N.

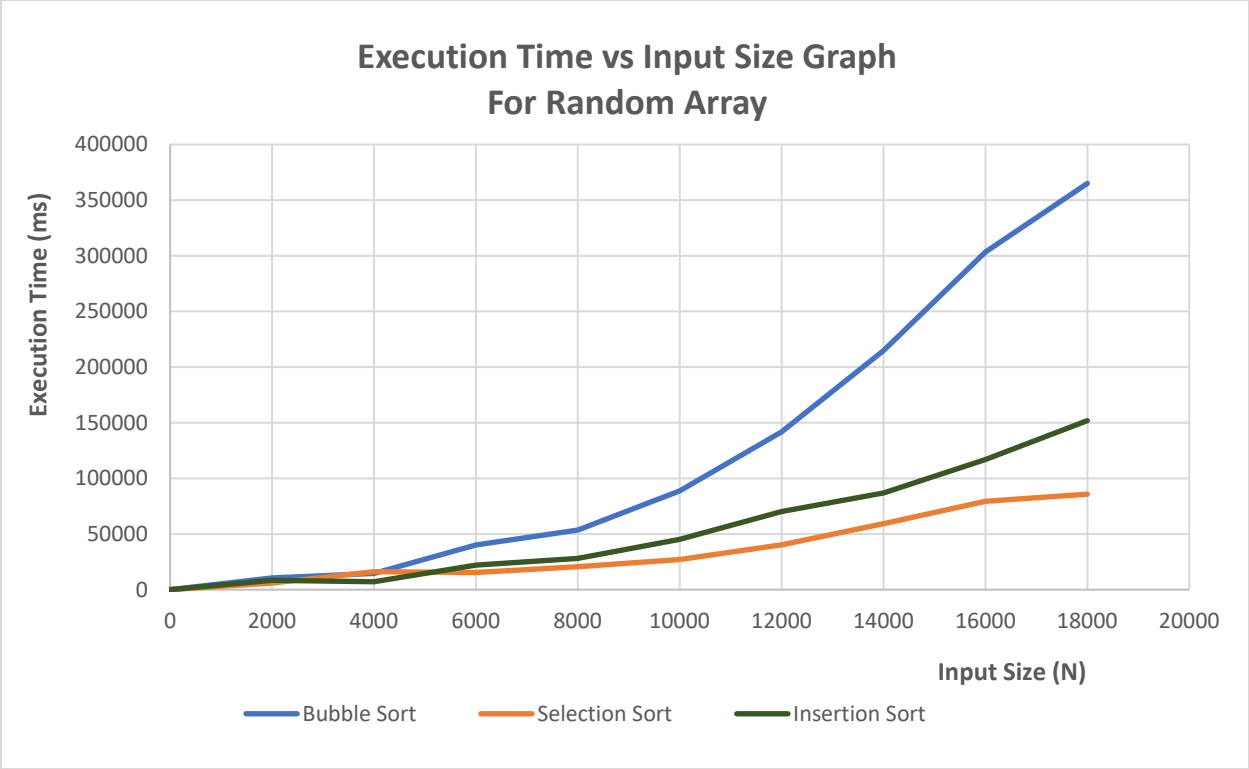
1) How does the performance vary with the input size?

Three Sorting algorithms were implemented in the lab. They were Bubble Sort, Selection Sort and Insertion Sort. Under random array case, the performance of each sorting algorithm was checked while changing the input size of the array.

The execution time varies with the size of the input. When the input is increased, execution time is also increased.

When comparing the bubble sort, selection sort and insertion sort in the random array case, time for execution is higher in bubble sort and lower in selection sort.
The same output can be obtained when considering the worst array and best array cases.

	Random Array		
N	**Bubble Sort**	**Selection Sort**	**Insertion Sort**
0	0	0	0
2000	10563	5975	8382
4000	14558	13112	7086
6000	40233	15413	21981
8000	5.35E+04	20330	28023
10000	8.88E+04	27017	45150
12000	1.42E+05	38978	70218
14000	2.15E+05	59317	86892
16000	3.03E+05	7.94E+04	116966
18000	3.65E+05	8.58E+04	151857
20000	4.98E+05	125309	183565



2) Do the empirical results you get agree with theoretical analysis?

Following is the theoretical analysis for each sorting algorithm,

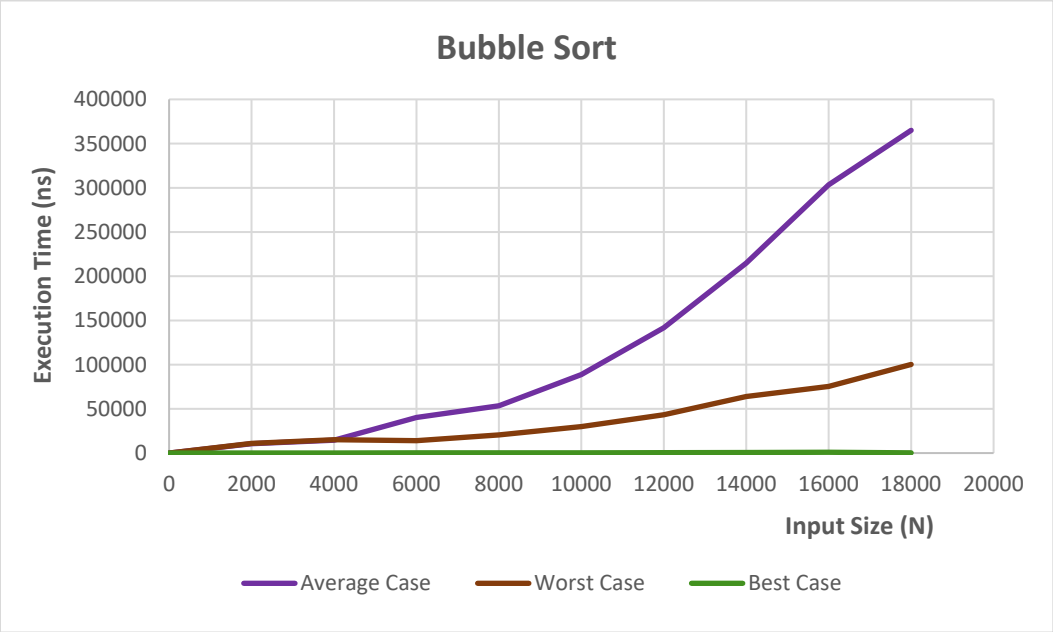
Algorithm	Best case	Worst case	Average case
Bubble sort	O(N)	O(N_2)	O(N_2)
Selection sort	O(N_2)	O(N_2)	O(N_2)
Insertion sort	O(N)	O(N_2)	O(N_2)

Let's compare each sorting algorithm's practical result with theoretical analysis.

Bubble Sort

N	Average Case	Worst Case	Best Case
0	0	0	0

2000	10563	10901	79
4000	14558	15109	158
6000	40233	14003	249
8000	5.35E+04	20623	317
10000	8.88E+04	30015	346
12000	1.42E+05	4.33E+04	431
14000	2.15E+05	6.40E+04	708
16000	3.03E+05	7.54E+04	926
18000	3.65E+05	1.00E+05	243
20000	4.98E+05	1.26E+05	212

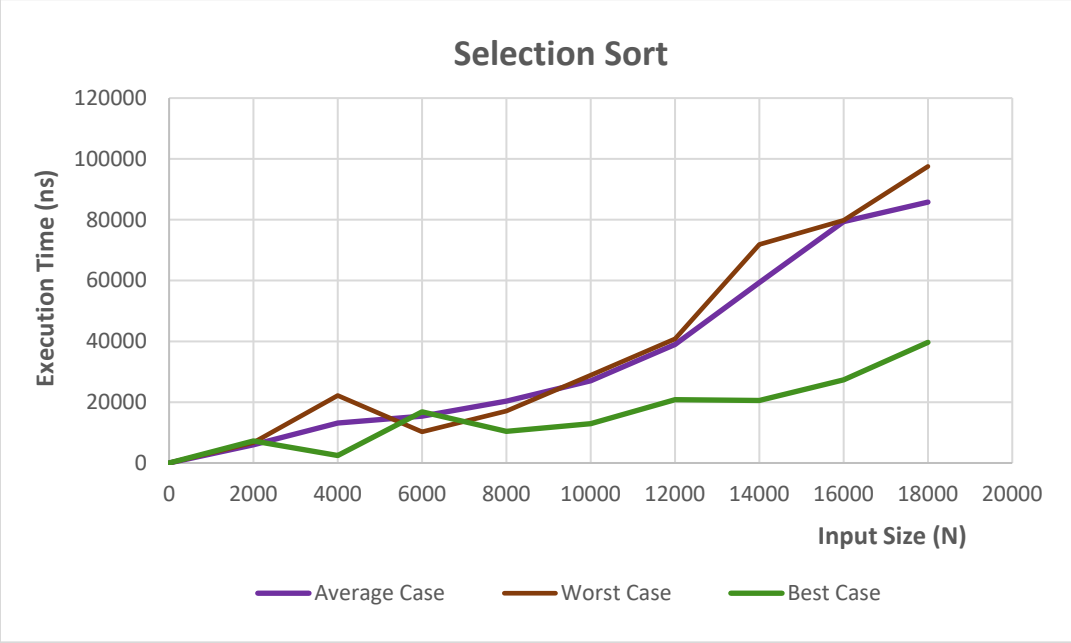


According to the above table we can clearly see that the bubble sort algorithm has the O(n) time complexity for the Best Case and for the other two cases the time complexity is $O(n^2)$. The same result can be observed from the above graph with the practical values. Therefore, we can say that the empirical behavior of the bubble sort algorithm agrees with the theoretical behavior for the above data set.

Selection Sort

N	Average Case	Worst Case	Best Case
0	0	0	0
2000	5975	6723	7251
4000	13112	22165	2464

6000	15413	10224	16837
8000	20330	17091	10422
10000	27017	28861	12930
12000	38978	40816	20821
14000	59317	7.19E+04	20565
16000	7.94E+04	7.98E+04	27355
18000	8.58E+04	9.75E+04	39676
20000	125309	1.24E+05	55655



According to the above table we can see that the Selection Sort has $O(n^2)$ time complexity for all three cases. From the above graph also, we can fairly see that behavior.

Insertion Sort

N	Average Case	Worst Case	Best Case
0	0	0	0
2000	8382	8399	86
4000	7086	12980	175
6000	21981	17276	268
8000	28023	28356	450
10000	45150	47524	452
12000	70218	66382	695
14000	86892	89296	631
16000	116966	121662	886

18000	151857	147510	191
20000	183565	185239	198

According to the above table we can see that the Insertion Sort algorithm has the O(n) time complexity for the Best Case while for the other two cases having the $O(n^2)$ time complexity. From the above graph also, we can observe the same behavior.

Overall, then, we can say that the three sorting algorithms' empirical and theoretical behaviors are consistent with one another.

3) How did/should you test your code. Explain the test cases you used and the rationale for use them?

The input size of the array was taken by running a for loop from 0 to 100 000 with step size of 2000. After that, three arrays were created according to the three separate cases which are Average Case, Worst Cases and the Best Case. For creating these separate arrays, the given three functions were used.

Then each array was sorted under three sorting algorithms which are Bubble Sort, Selection Sort and Insertion Sort and the time taken for each sorting algorithm was observed. After that, a

graph was plotted by using the input size as the independent variable and the execution time in nano seconds as the dependent variable.