CO 322 – Data Structures and Algorithms

Sorting Algorithms – Lab Report

KARUNACHANDRA R.H.I.O.

E/17/153

Variation of performance with the input size

Bubble, Selection and Insertion algorithms were implemented in the code. To compare the performances of those three implementations with respect to the Input Size (Size of the array that should be sorted), program was executed under three cases.

They are,

- Average case
- Worst case
- Best case

Execution times were measured in those cases with the change of input size. Here I used input sizes as 5,50,500,5000 and 50000.

Table 1: Variation of Execution time (in nano seconds) **under average case**

Sorting Algorithm	Size 5	Size 50	Size 500	Size 5000	Size 50000
Bubble	4,100	83,900	3,433,866	50,239,100	5,400,576,400
Selection	4,100	35,300	2,366,166	19,582,000	1,445,722,700
Insertion	2,750	19,233	1,879,200	12,782,300	365,527,400

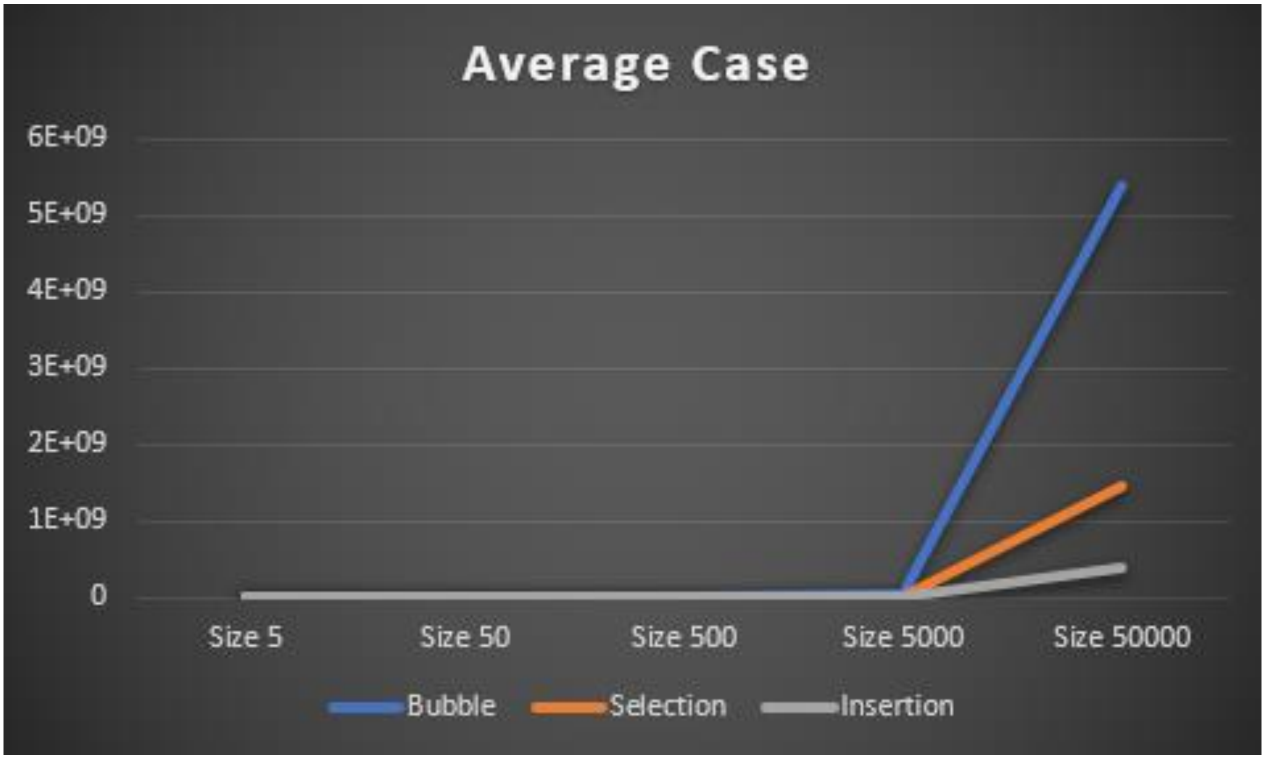


Table 2: Variation of Execution time (in nano seconds) **under Best Case**

Sorting Algorithm	Size 5	Size 50	Size 500	Size 5000	Size 50000
Bubble	2,200	3,100	12,300	107,700	1,062,300
Selection	3,900	33,200	2,291,600	11,491,300	427,076,000
Insertion	2,300	4,000	20,600	192,500	1,902,800

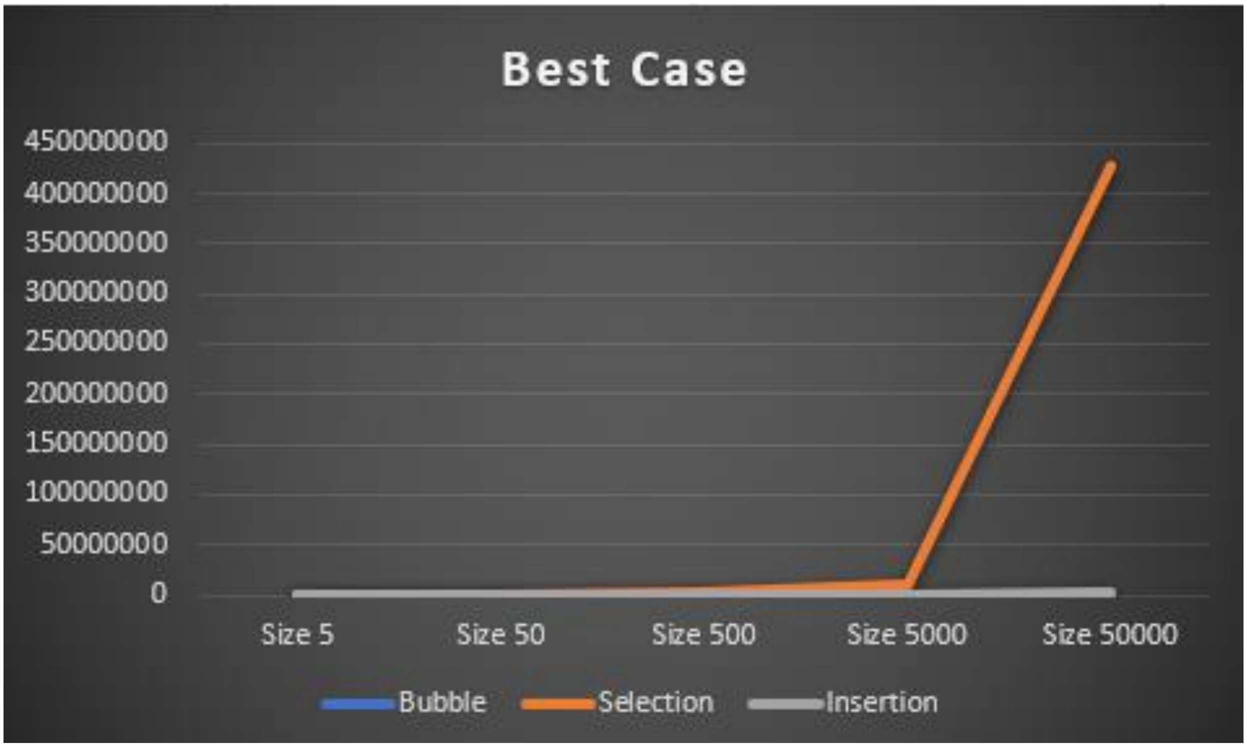
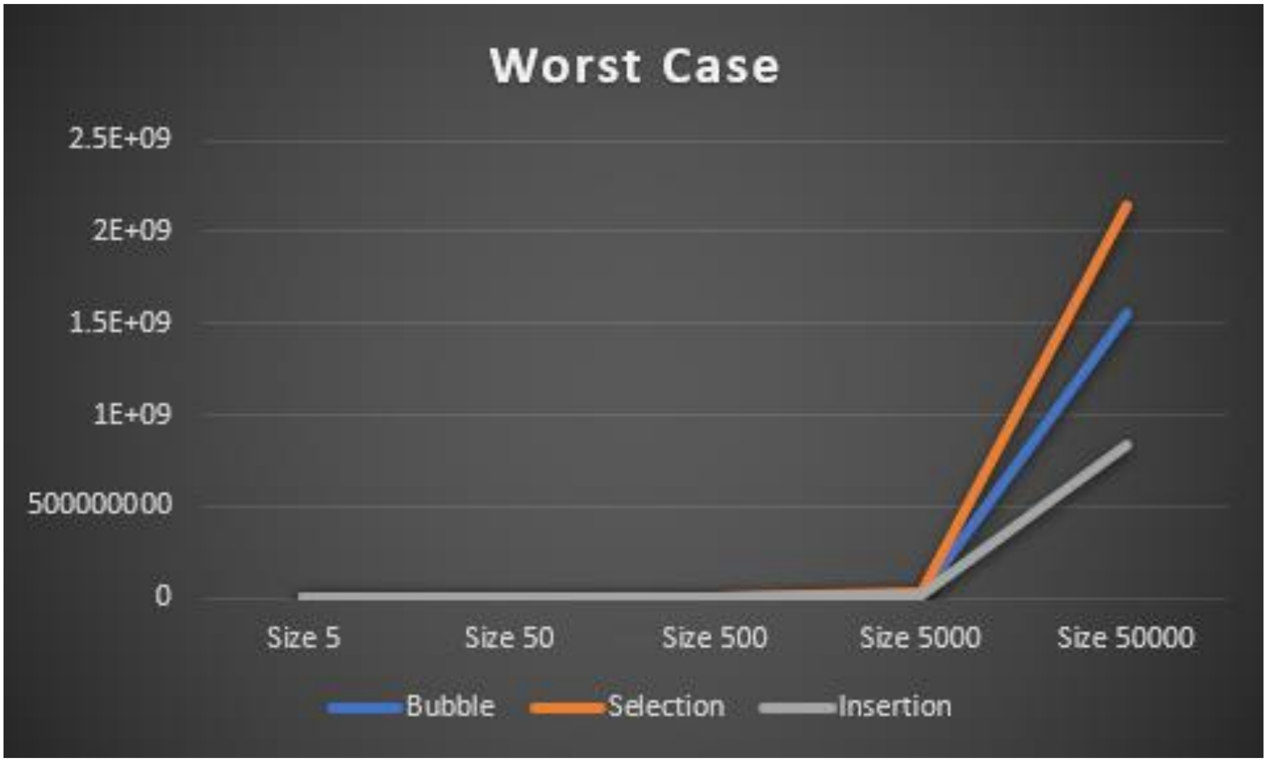


Table 3: Variation of Execution time (in nano seconds) **under worst case**

Sorting Algorithm	Size 5	Size 50	Size 500	Size 5000	Size 50000
Bubble	4,000	77,400	3,433,666	20,924,000	1,547,910,600
Selection	4,000	35,700	2,501,900	26,051,600	2,148,315,600
Insertion	2,700	35,600	2,562,633	15,711,100	841,990,100



Since those time values in nano seconds are extremely large in big input values there is a vast range for the time values. So, it is not very much clear in the graphs.

However, it can be observed that execution time increases with respect to the input size in every case.

Empirical results VS Theoretical analysis

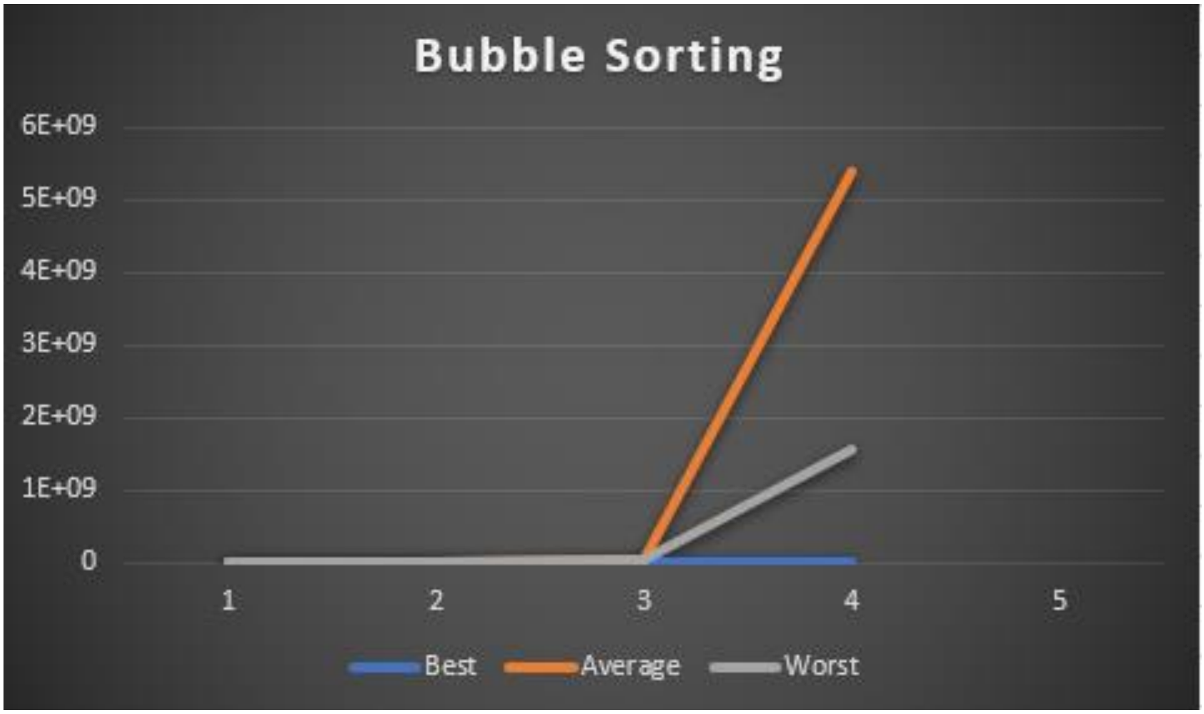
Table 3: Theoretical Analysis

Sorting Algorithm	Best Case	Average Case	Worst Case
Bubble	O(n)	O(n^2)	O(n^2)
Selection	O(n^2)	O(n^2)	O(n^2)
Insertion	O(n)	O(n^2)	O(n^2)

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

1)Bubble Sorting:

Input Size	Best Case	Average Case	Worst Case
5	2200	4100	4000
50	3100	83900	77400
500	12300	3433866	3433666
5000	107,700	50239100	20924000
50000	1062300	5400576400	1547910600

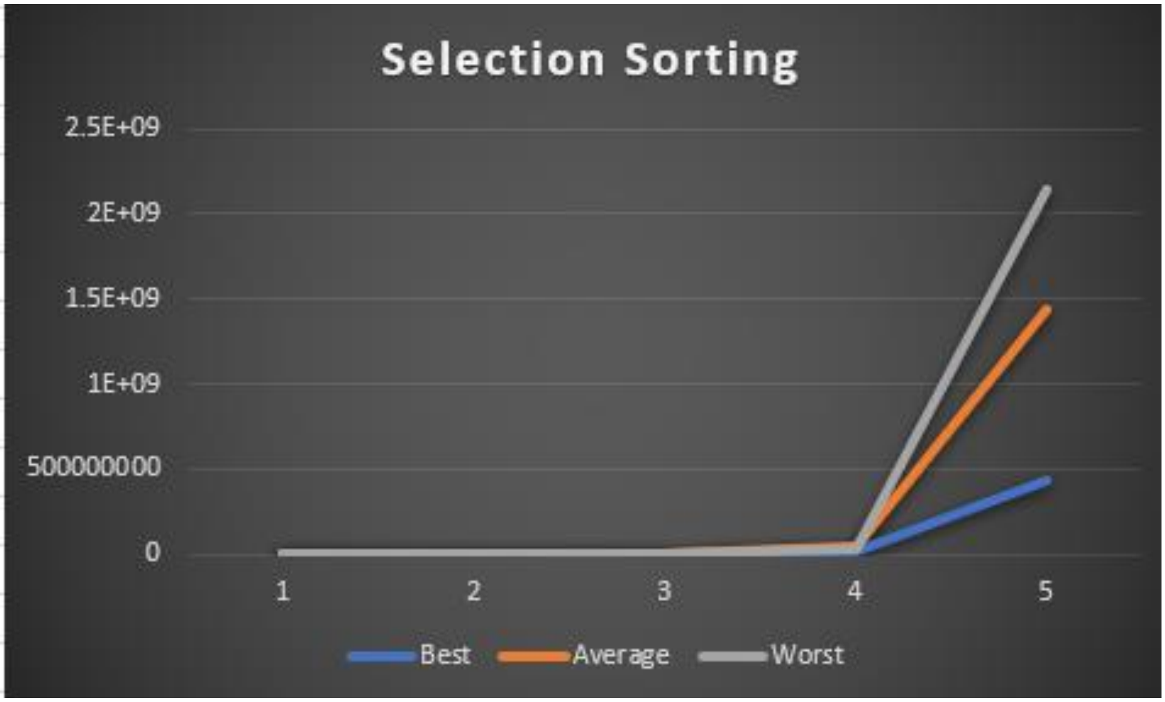


As in the table 3, bubble sorting has O(n) time complexity for best case and O(n^2) time complexity for average and worst cases.

It can be clearly observed in the above graph with practical values.

2)Selection Sorting:

Input Size	Best Case	Average Case	Worst Case
5	3900	4100	4000
50	33200	35300	35700
500	2291600	2366166	2501900
5000	11,491,300	19,582,000	26,051,600
50000	427076000	1445722700	2148315600

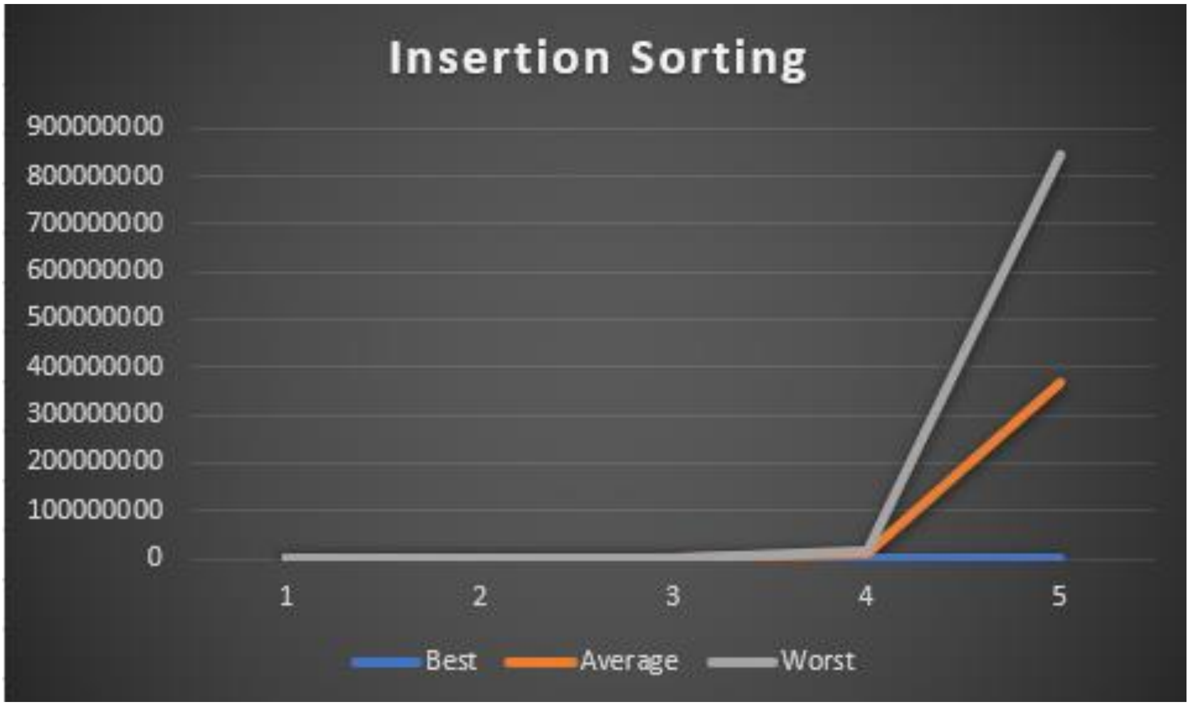


In Selection sorting, all best, average and worst cases have O(n^2) time complexity theoretically. It is fairly visible in the above graph with practical values.

3)Insertion Sorting:

Input Size	Best Case	Average Case	Worst Case
5	2300	2750	2700
50	4000	19233	35600

500	20600	1879200	2562633
5000	192,500	12,782,300	15,711,100
50000	1902800	365527400	841990100



In Insertion sort, best case has the time complexity O(n) and average and worst cases have O(n^2). That can be seen in the above graph clearly.

Hence, as an overall conclusion regarding empirical and theoretical execution time values it can be said that they are almost same in the pattern.

Test cases and performance measuring

In testing and performance measuring I used all three cases which are average case, best case and worst case. In those three cases I used input sizes as 5,50,500,5000 and 50000.

I could observe that for same input size, same algorithm and the same case it would give different execution times when I executed the same program again and again. This may be due to other programs running in the background, performance changing in the pc and caching effects etc. In that case, to be more precise and accurate I took time values three times in most of the cases and got the average of them. But in some cases where the execution times are extremely different from each other (Like 10,000ns and 500,000ns) I dismiss some time values and took nearly equal values and got the average.