Data Structures 2

Lab 1 Report: Sorting Algorithms

Abdelrahman Ahmed Saad ID: 9625

Mohamed Ashraf Kasem ID: 9621

Eyad Amir Beshr ID: 9626

Group 4 — Section 2
(Eyad: Section 1)

1 Algorithms Overview

1.1 Bubble Sort

Bubble Sort repeatedly compares adjacent elements and swaps them if they are out of order. It includes an early exit where if no swaps occur in a full pass, the algorithm stops early.

1.2 Selection Sort

Selection Sort finds the minimum element in the unsorted portion of the array and swaps it into its correct position. It performs exactly one swap per outer loop iteration regardless of the input.

1.3 Insertion Sort

Insertion Sort picks each element and shifts larger elements to the right until it finds the correct position to insert it. It performs well on nearly sorted data due to fewer required shifts.

2 Time Complexity

Table 1: Best and Worst Case Time Complexity

Algorithm	**Best Case**	**Worst Case**
Bubble Sort	O(n)	O(n^2)
Selection Sort	O(n^2)	O(n^2)
Insertion Sort	O(n)	O(n^2)

3 Results

Table 2: Running Time (ms) for Each Sorting Algorithm

Array Size	Bubble (ms)	Selection (ms)	Insertion (ms)
10,000	3,714.4	1,570.7	1,662.0
25,000	23,238.3	9,580.5	10,314.5
50,000	94,680.2	38,514.9	41,869.1
100,000	565,955.8	156,112.2	166,872.0

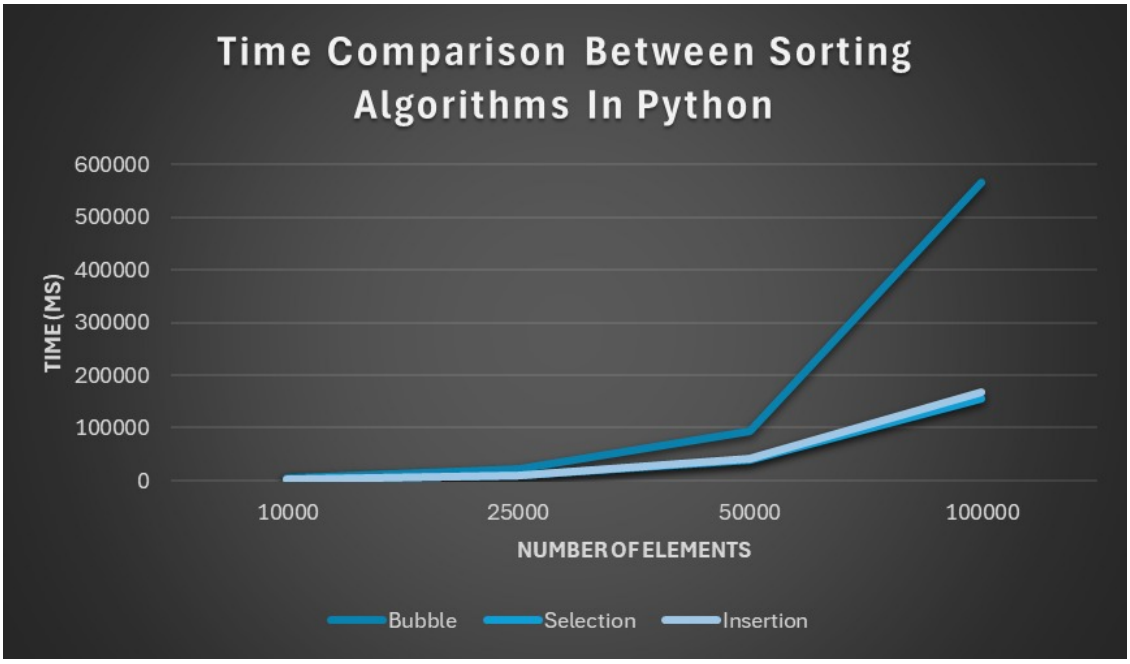

Figure 1: Time Comparison Between Sorting Algorithms in Python

4 Conclusion

All three algorithms confirm their $O(n^2)$ worst case behavior as array size grows. Bubble Sort is the slowest due to its high number of swaps. Selection Sort and Insertion Sort perform similarly, with Selection Sort having a slight edge in Python due to fewer shift operations per iteration.