Algorithm Analysis and Implementation

Mariam Ali M H Al Obaidly
MSc Artificial Intelligence
University of Essex Online

This report examines the Bubble Sort algorithm and provides a comparison with QuickSort in order to evaluate their efficiency and functionality. Bubble Sort is defined as a simple comparison-based sorting algorithm that repeatedly steps through a list, compares adjacent elements, and swaps them when they are in the wrong order (Cormen et al., 2022). The algorithm performs multiple passes through the dataset, and after each pass, the largest unsorted element is moved to its correct position at the end of the array. In contrast, QuickSort follows a divide-and-conquer strategy, where a pivot element is selected to partition the dataset into smaller sub-arrays that are then sorted recursively.

Time complexity is commonly expressed using Big-O notation, which describes the asymptotic growth rate of an algorithm as the input size increases (Knuth, 1997). The efficiency of the two algorithms differs significantly when analyzing their theoretical time complexity and practical performance. Bubble Sort has an average and worst-case time complexity of $O(n^2)$ due to its nested-loop structure, which requires repeated comparisons across the dataset. However, in the best-case scenario, where the dataset is already sorted and an early termination condition is implemented, its time complexity can approach $O(n)$. In contrast, QuickSort achieves an average time complexity of $O(n \log n)$, making it significantly more scalable for larger datasets. Although QuickSort has a worst-case complexity of $O(n^2)$, this scenario is less common when appropriate pivot selection strategies are applied. The benchmark graph clearly reflects these theoretical differences, as the execution time of Bubble Sort increases rapidly with larger input sizes, whereas QuickSort demonstrates a more gradual growth pattern. As illustrated in Figure 1, Bubble Sort demonstrates a rapid increase in

execution time as input size grows, while QuickSort exhibits a more scalable growth pattern.

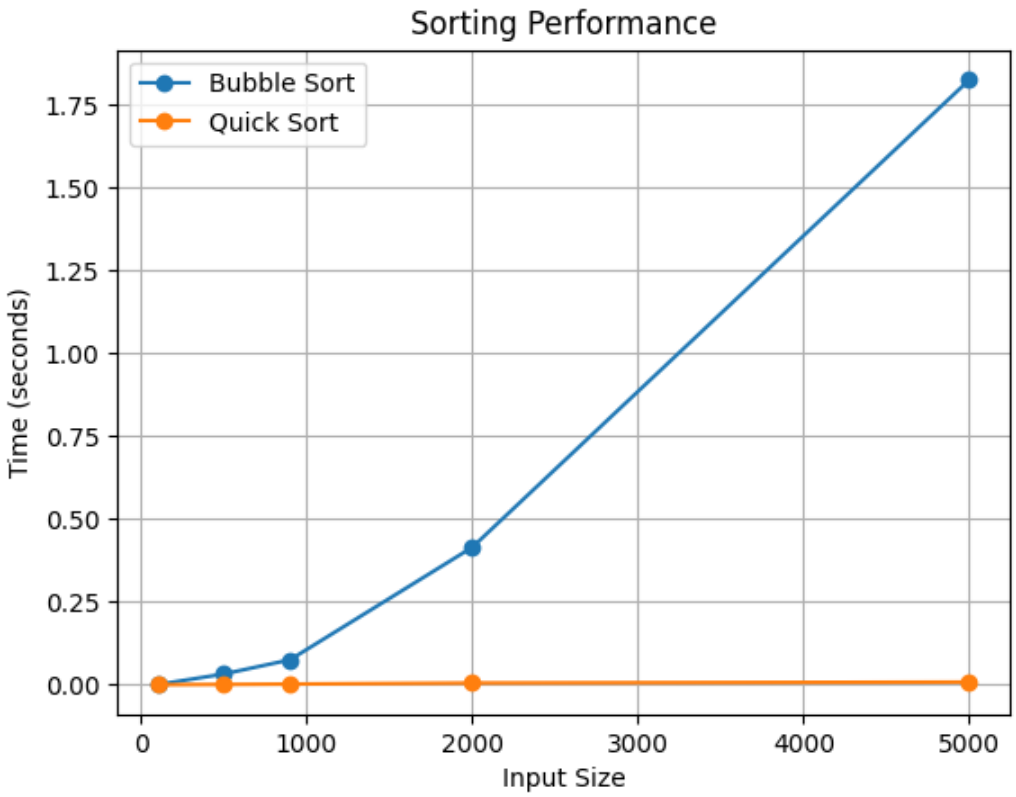

Figure 1: Performance comparison between Bubble Sort and QuickSort across different input sizes.

A direct comparison between the two algorithms highlights their suitability for different practical scenarios. Bubble Sort may still be suitable for educational purposes or for very small datasets, where simplicity is prioritised over performance. In contrast, QuickSort is far more appropriate for large datasets and real-world systems that require efficient and scalable sorting performance. However, QuickSort presents a potential performance risk due to its $O(n^2)$ worst-case complexity when pivot selection results in highly unbalanced partitions. In practice, this risk is commonly mitigated through improved pivot selection strategies, such as randomised pivot selection or median-of-three techniques.

References:

Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C. (2022) *Introduction to Algorithms*. 4th edn. Cambridge, MA: MIT Press.

Knuth, D.E. (1997) *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. 3rd edn. Boston: Addison-Wesley.