Task “Algorithm Analysis and Implementation” Unit 3

Introduction

The role of algorithms, in general, is to solve problems, both theoretical and practical or real-world ones. The primary purpose of sorting algorithms is to arrange numbers in a specific numerical order. Additionally, sorting algorithms, specifically bubble sort, are used to help students understand the basics of computer programming. The selected algorithm in this document is the bubble sort algorithm, and the chosen implementation is in Python. This document aims to explain some of the challenges, benefits, use cases, and other aspects of sorting algorithms.

Submitted Work

1. Explain algorithm efficiency for small vs. large datasets:

A comparative testing of bubble sort and quick sort algorithm’s time complexity (Big-O notation) efficiency:

- In the **best-case** scenario, the time complexity is O(n), and the space complexity is O(1). The efficiency is fast if list is already sorted.
- On **average-case** scenario, the time complexity is $O(n^2)$ and the space complexity is O(1), which reduces the efficiency for most lists.
- In the **worst-case** scenario, the time complexity is $O(n^2)$ and the space complexity is O(1), making it very slow for large, unsorted lists.

2. A comparison of the two algorithms and their real-world use cases:

A comparison between bubble sort and quick sort algorithms and their real-world use cases:

- The **bubble sort** algorithm space complexity is O (1) and the average time complexity is $O(n^2)$. In real world cases the bobble sort algorithm is used mainly for educational purposes, tiny datasets, or nearly sorted lists. It is mainly used for teaching sorting concepts or small, nearly sorted lists.
- The **quick sort** algorithm space complexity is O (log n) and the average time complexity is O (n log n). In real world cases the quick sort algorithm will be used for general-purpose sorting, for example in databases, libraries and large datasets quick sort is often used in practice to sort large sets of data because it is fast and efficient also in use of resources.

(Bakare, Okewu, Abiola, Jaji, and Muhammed, 2024)

3. A Bubble Sort code with a working result “bubble sort.py:

3.1. Result:

```
PS C:\Users\AmnonMalka\Documents\Code> &
C:/Users/AmnonMalka/AppData/Local/Programs/Python/Python313/python.exe
"c:/Users/AmnonMalka/Documents/Code/bubble sort.py"
before sorting: [4, 7, 2, 8, 6, 3, 9, 5, 1, 0, 12, 14, 11, 10]
After pass 1 : [4, 2, 7, 6, 3, 8, 5, 1, 0, 9, 12, 11, 10, 14]
After pass 2 : [2, 4, 6, 3, 7, 5, 1, 0, 8, 9, 11, 10, 12, 14]
After pass 3 : [2, 4, 3, 6, 5, 1, 0, 7, 8, 9, 10, 11, 12, 14]
After pass 4 : [2, 3, 4, 5, 1, 0, 6, 7, 8, 9, 10, 11, 12, 14]
After pass 5 : [2, 3, 4, 1, 0, 5, 6, 7, 8, 9, 10, 11, 12, 14]
After pass 6 : [2, 3, 1, 0, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14]
After pass 7 : [2, 1, 0, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14]
After pass 8 : [1, 0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14]
After pass 9 : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14]
After pass 10 : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14]
After pass 11 : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14]
After pass 12 : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14]
After pass 13 : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14]
After pass 14 : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14]
After sorting: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14]
```

4. **Produce a graph showing performance (time complexity) for different input sizes:**

Even though I made significant progress in improving my programming skills over the past few weeks, I am yet to be fully proficient in programming, which is why I did not manage to develop code for a graph even after multiple tries. However, Therefore, I plan to submit my work without a code of a graph this time.

Report

One of the earliest notations of the bubble sort algorithm is considered to have been invented by Edward Harry Friend in 1956. This is a simple algorithm that is used to sort lists. of n numbers in ascending order. The algorithm functions by examining each pair of neighboring elements in the series, from left to right, and switching their positions if they are out of order.

Bubble sort algorithms are used for educational purposes, for tiny datasets, or for nearly sorted lists. It is used primarily for teaching sorting concepts or for small, nearly sorted lists. A computer programmer will need to comprehend the concept in which she or she would like to learn how to develop software code. The sorting process can be done manually by a human, which is not feasible with large data sets; algorithms will do that much more quickly, effectively, and automatically using code within a program. The use of algorithms and sorting algorithms in the real world encompasses a wide range of applications, including data analysis, research, space exploration, and many more.

(Rai, M., Parmar, H. and Sharma, S., 2025)

In addition to automating these activities, some algorithms also enhance their efficiency through advanced mathematical calculations.

bubble sort algorithms are less effective when used with large datasets and should be used with smaller ones. Algorithms such as quicksort are more effective when dealing with large numbers, in real-world scenarios. The quicksort algorithm will be used for general-purpose sorting, for example in databases, libraries and large datasets. Quick sort is often used in practice to sort large sets of data because it is fast and efficient. It is widely used in practice for sorting large datasets as it has speed and efficiency. At the same time, the bobble sort algorithm will be used in the real world for educational purposes, tiny datasets, or nearly sorted lists. It is mainly used for teaching sorting concepts or small, nearly sorted lists.

(Muhammad, Malik, Akhtar, Baloch, and Rajwana, 2025)

References

Bakare, K.A., Okewu, A.A., Abiola, Z.A., Jaji, A. and Muhammed, A., (2024). A COMPARATIVE STUDY OF SORTING ALGORITHMS: EFFICIENCY AND PERFORMANCE IN NIGERIAN DATA SYSTEMS. FUDMA JOURNAL OF SCIENCES, 8(5), pp.1-5.

Muhammad, M.H.G., Malik, J.A., Akhtar, M., Baloch, M.A. and Rajwana, M.A., (2025). Sort Data Faster: Comparing Algorithms. Southern Journal of Computer Science,

1(02), pp.28-37. Available from: https://sjcs.isp.edu.pk/index.php/SJCS/article/view/11 [Accessed: 21 August 2025]

Rai, M., Parmar, H. and Sharma, S., (2025), February. Comparative Analysis of Sorting Algorithms: Time Complexity and Practical Applications. In 2025 3rd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT) (pp. 26-32). IEEE.

This document has been written solely for educational purposes. All references, names, and trademarks mentioned here remain the property of their respective owners and are used here strictly for the educational context. Grammarly was used exclusively for proofreading and enhancing the clarity and language of the text. All academic writing, analysis, argumentation, and conclusions are entirely the original work of the author.