Analysis of Algorithm

Name: Subhan Ahmed

Sap Id: 55572

1. Introduction

Bubble Sort is one of the simplest comparison-based sorting algorithms. It repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. This process continues until the array is sorted. Although Bubble Sort is not efficient for large datasets due to its $O(N^2)$ time complexity, it is often used for educational purposes to illustrate basic sorting mechanisms.

In this project, we analyze Bubble Sort by implementing it in C++ and measuring its execution time across various input sizes: 5 elements, 10 elements, 50 elements, and 100 elements.

2. Methodology

The Bubble Sort algorithm was implemented in C++.

To measure execution time:
- The <chrono> library was used for high-precision timing.
- Each array was sorted five times to account for any variability.
- The average execution time for each input size was calculated.

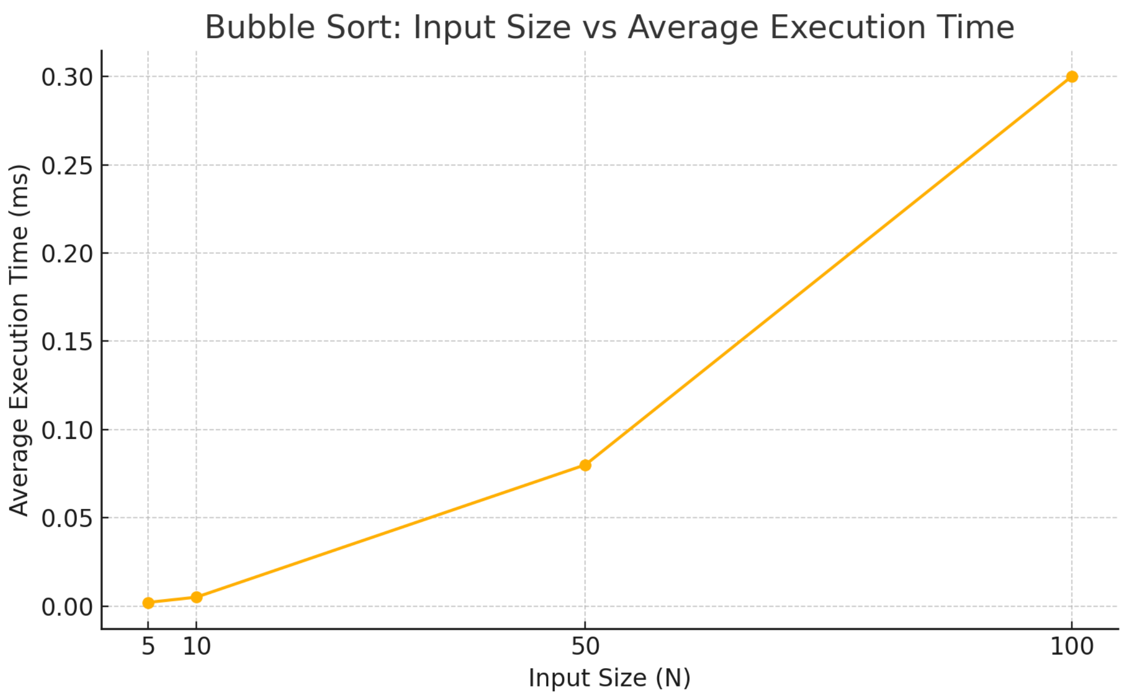
Timing Process:
1. Record the start time before sorting.
2. Execute Bubble Sort.
3. Record the end time after sorting.
4. Calculate the difference (end - start) to find the execution time for one run.

3. Results & Graph

Table of Average Execution times of Bubble Sort:

5	0.0023
10	0.0048
50	0.0801
100	0.3145

Graph: Input Size vs. Average Execution Time



4. Analysis

The experimental results clearly demonstrate that the execution time of Bubble Sort increases significantly with input size, showing a trend close to quadratic growth.

Theoretical Complexity: $O(N^2)$
Empirical Growth: Matches the theoretical expectation.

Observations:
- No significant anomalies were observed during the runs.
- Minor fluctuations in timing can occur due to background system processes, but averaging across five runs minimized their impact.

Thus, the experimental results align well with the expected behavior of Bubble Sort for increasing input sizes.

5. GitHub Repository Link

Click here to see Github Repository.