Performance Visualization Tool for Algorithmic Time and Memory Analysis

(Minor Project Presentation)

Dr B R Ambedkar National Institute of Technology Jalandhar

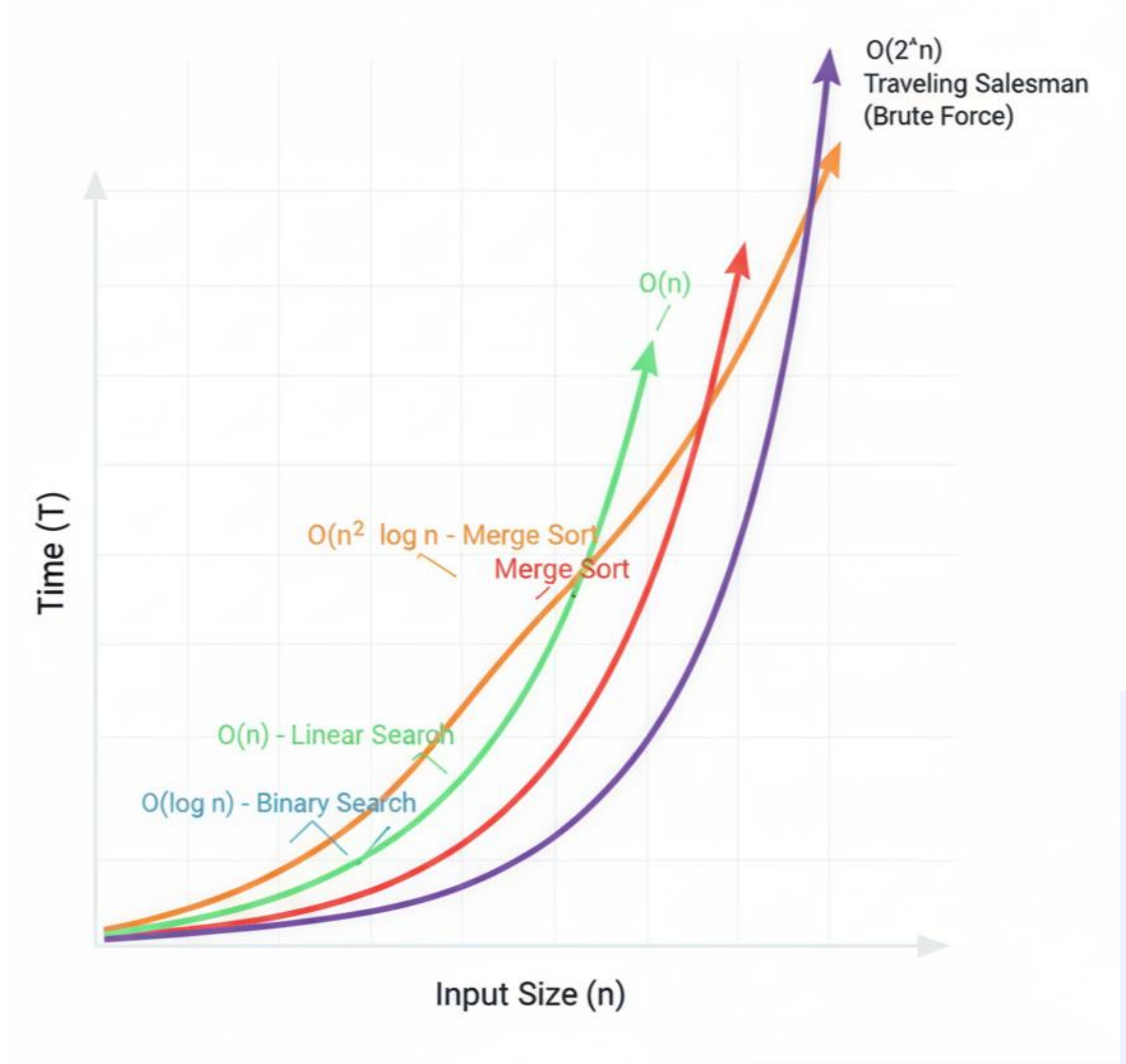
Presented By:
Beerkanwar Singh (23103030)
CH Jithender Reddy (23103035)
Inderjeet Singh (23103068)

Under Supervision of:
Dr. Urvashi Bansal
Assistant Professor
Department of Computer Science

Table of Contents

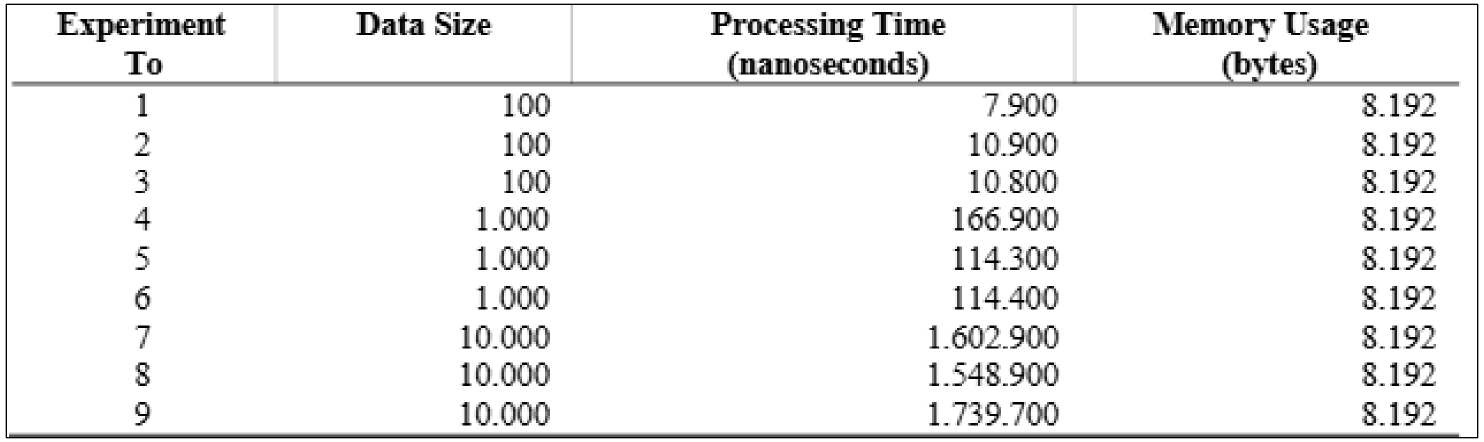
Introduction

- Algorithms form the **core of computer science**, determining efficiency and scalability of computational processes.
- Analyzing **time and memory performance** helps understand how algorithms behave with varying input sizes.
- Traditional **Big-O notation** gives theoretical insight but can mis real-world performance variations.
- The project provides a **practical and visual approach** to study and compare algorithm efficiency.
- Our goal is to make learning algorithms **interactive, dynamic, and easier to understand**.



Problem Statement

- Existing platforms focus mainly on theoretical analysis and lack interactive ways to observe real-time performance changes.
- Users cannot easily visualize how different algorithm parameters impact execution time and memory consumption.
- This limits understanding of the practical trade-offs and efficiency of algorithms under varying conditions.
- There is a need for an interactive tool that empirically demonstrates these relationships to enhance learning and analysis.



Experiment To	Data Size	Processing Time (nanoseconds)	Memory Usage (bytes)
1	100	7.900	8.192
2	100	10.900	8.192
3	100	10.800	8.192
4	1.000	166.900	8.192
5	1.000	114.300	8.192
6	1.000	114.400	8.192
7	10.000	1.602.900	8.192
8	10.000	1.548.900	8.192
9	10.000	1.739.700	8.192

Source: Comparative Analysis of Memory and Time of Sorting Algorithm Using C++

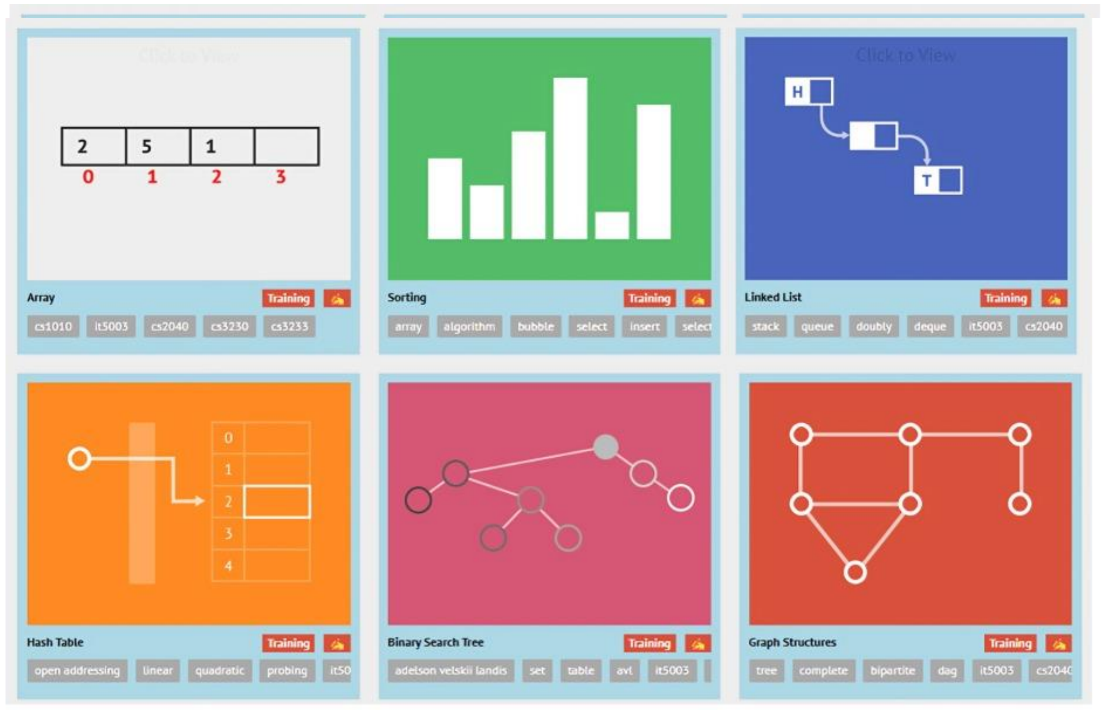
Literature Review:

Research Paper	Key Findings	Relevance to Project
Comparative Analysis of Memory Performance and Processing Time of Five Sorting Algorithms Using C++ (M. Nazril Ilham et al., 2025)	Benchmarks five sorting algorithms (Bubble, Insertion, Selection, Merge, Quick) based on execution time and memory consumption using real test data.	Demonstrates how algorithm choice and input size affect time and memory — provides baseline for comparative visualization.
Using Visualization to Evaluate the Performance of Algorithms for Multivariate Time Series Classification (E. Acuña & R. Aparicio, 2025)	Utilizes visual analytics to assess and compare the performance of multiple algorithms on time-series data.	Supports the idea of using *visualization* to make algorithm performance analysis more intuitive and interpretable.
Key Concepts, Weakness and Benchmark on Hash Table Performance (Tapia-Fernández et al., 2022)	Analyzes different hash table implementations, highlighting performance trade-offs in speed, memory usage, and collision handling.	Illustrates how data-structure design impacts time and memory — aligns with the project’s focus on parameter-based performance evaluation.

Existing Tools

- **VisuAlgo (NUS):** A popular tool visualizing many algorithms and data structures with animations (e.g., swaps, tree rotations).
- **AlgoAR:** A recent mobile AR tool that includes performance metrics like execution time, CPU, and memory usage.
- **Algorithm Visualizer:** An interactive platform for visualizing sorting, searching, and graph algorithms step-by-step.

Primary Focus: Most tools emphasize algorithmic steps and logic over quantitative performance trade-offs.
Memory Visualizers: Some tools focus on memory usage, but this area is less mature (Blanco et al.).



Source: VisuAlgo Tool

Gaps & Our Focus

Parameter Tuning: Few tools allow real-time tuning of algorithm-specific parameters (e.g., pivot choice, buffer sizes) to see live performance impact.

Quantitative Metrics: A lack of tools that simultaneously measure and chart actual time and memory usage vs. parameters.

Memory Footprint: Dynamic memory allocation, peak usage, and fragmentation are often not visualized.

Our Goal: To build a tool focused on these gaps: interactive parameter/input sweeping with live, quantitative performance graphing.

Objectives

- To design and develop a **performance visualization tool** that analyzes algorithms based on **time and memory usage**.
- To allow users to **select and run different algorithms** with varying input sizes or parameters.
- To **measure real execution metrics** (not theoretical) and store them for comparison.
- To **visualize results graphically** through plots showing time vs. input size and memory vs. input size.
- To provide an **interactive and educational platform** that helps users understand algorithm efficiency and scalability.

Tech Stack

Python

Core language for its simplicity and rich library support for the entire implementation.

Time & psutil

Used to measure real execution time and system memory usage (peak memory) respectively.

Pandas

Used for efficient data storage, manipulation, and analysis of performance results.

Matplotlib

Used to visualize the final time and memory performance data through graphs and charts.

System Design

Algorithm Selection

- Choose commonly used algorithms such as sorting (Bubble, Merge, Quick), searching, or others.
- Define the parameters that will vary - e.g., data type, data distribution, data storage.

Input Generation

- Automatically generate input datasets of different sizes.
- Ensure consistent input conditions for fair comparison between algorithms.

Performance Measurement

- Record **execution time** using Python’s time module.
- Measure **memory usage** using libraries like **psutil** .

System Design

Data Collection and Processing

- Store all results (time, memory, input size) in a structured format using **PandasDataFrames**.
- Process and prepare the data for visualization.

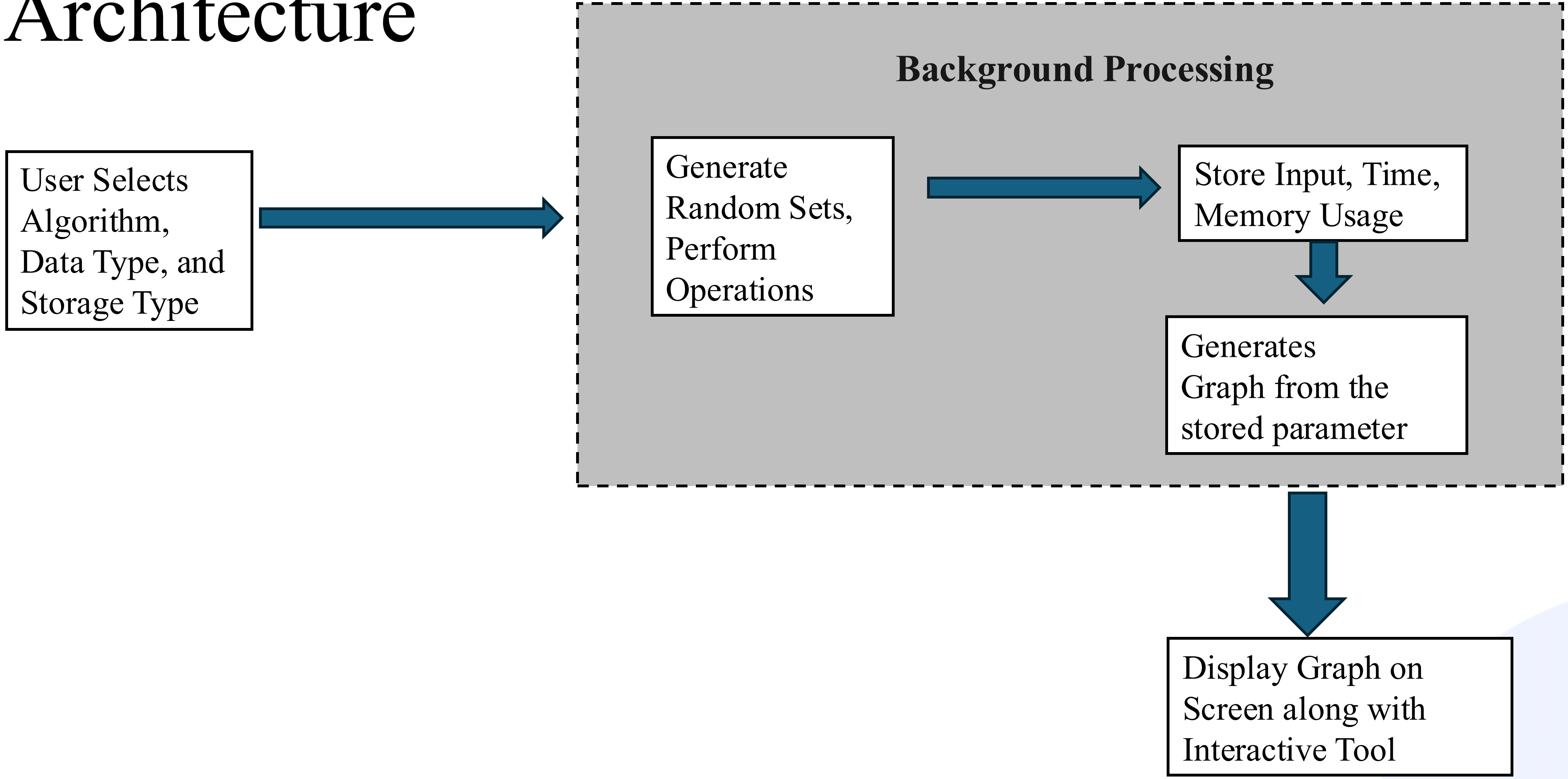
Visualization

- Use **Matplotlib** to plot graphs showing performance trends.
- Display **Time vs. Input Size** and **Memory vs. Input Size** comparisons.

User Interface / Interaction

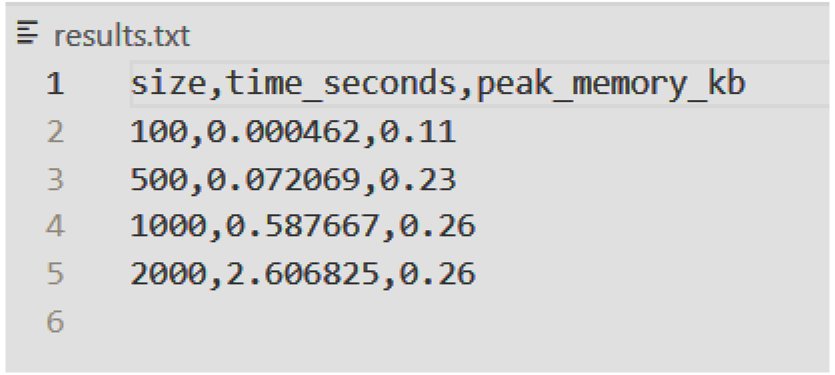
- Provide a simple GUI or command-based interface.
- Allow users to choose algorithms, input sizes, and view real-time visual results.

Architecture


Background Processing
User Selects Algorithm, Data Type, and Storage Type
Generate Random Sets, Perform Operations
Store Input, Time, Memory Usage
Generates Graph from the stored parameter
Display Graph on Screen along with Interactive Tool

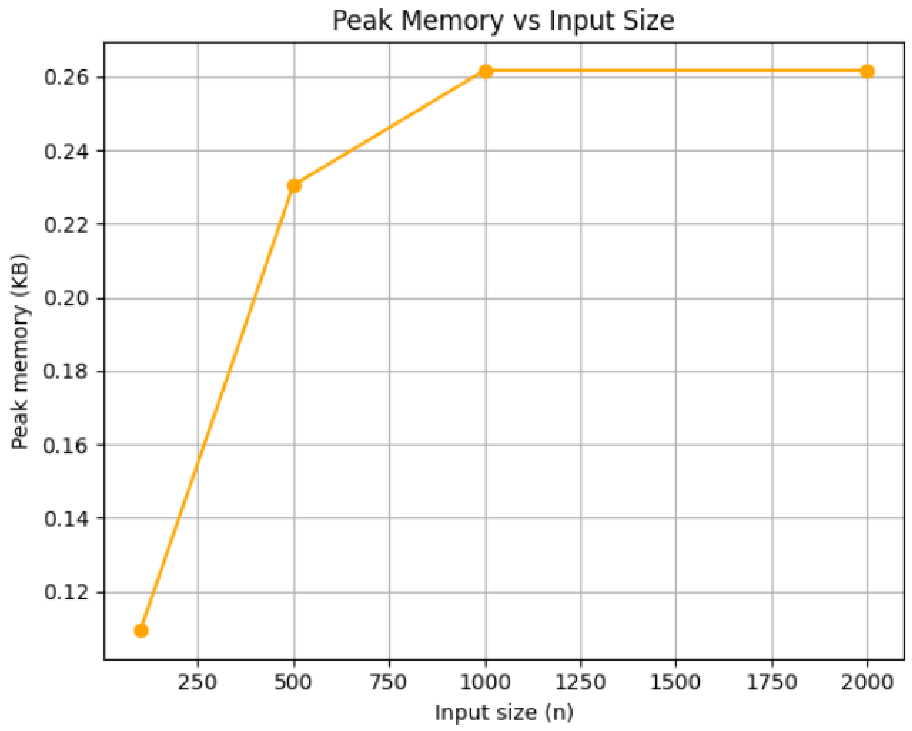
Sample Visualization

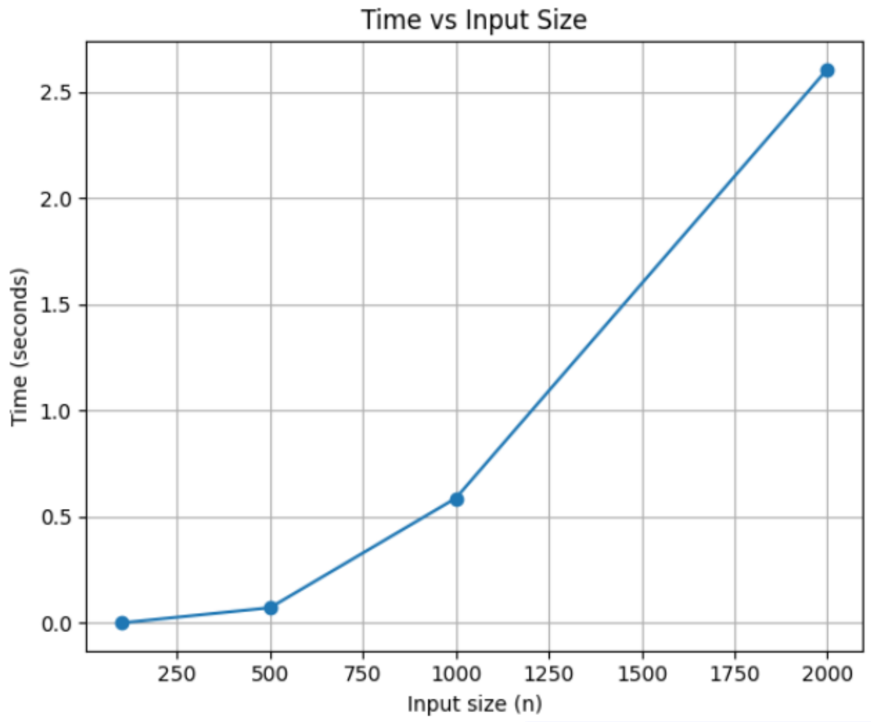
- **Algorithm Type:** Sorting
- **Algorithm:** Bubble Sort
- **Input Type:** Integer
- **Storage Type:** array



results.txt

```
size,time_seconds,peak_memory_kb
100,0.000462,0.11
500,0.072069,0.23
1000,0.587667,0.26
2000,2.606825,0.26
```





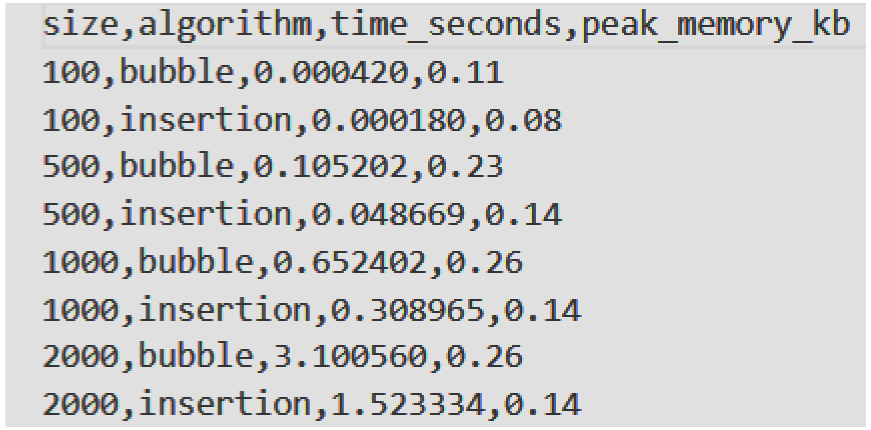
Comparative Visualization

Algorithm Type: Sorting

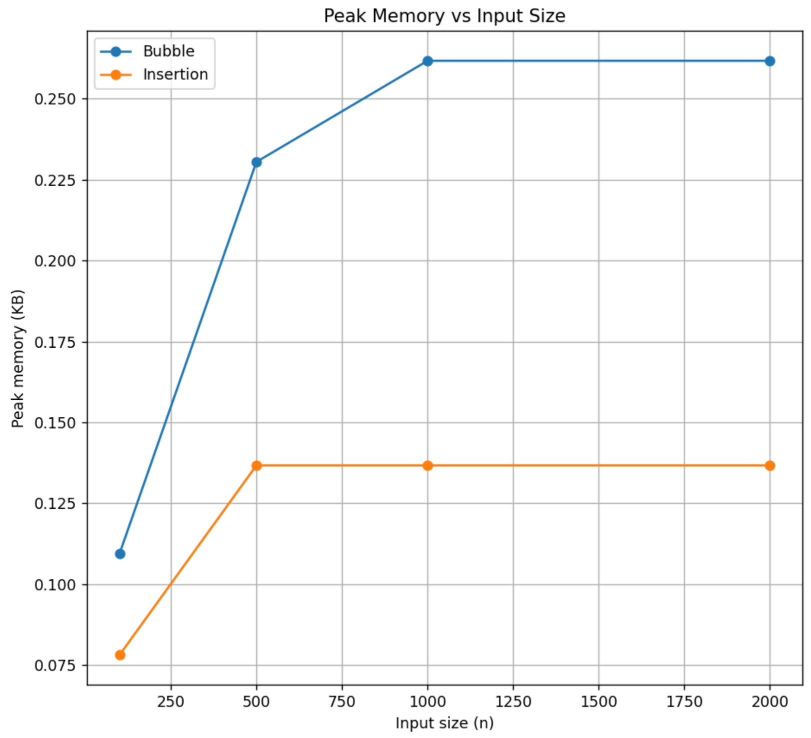
Algorithms: Bubble v/s Insertion Sort

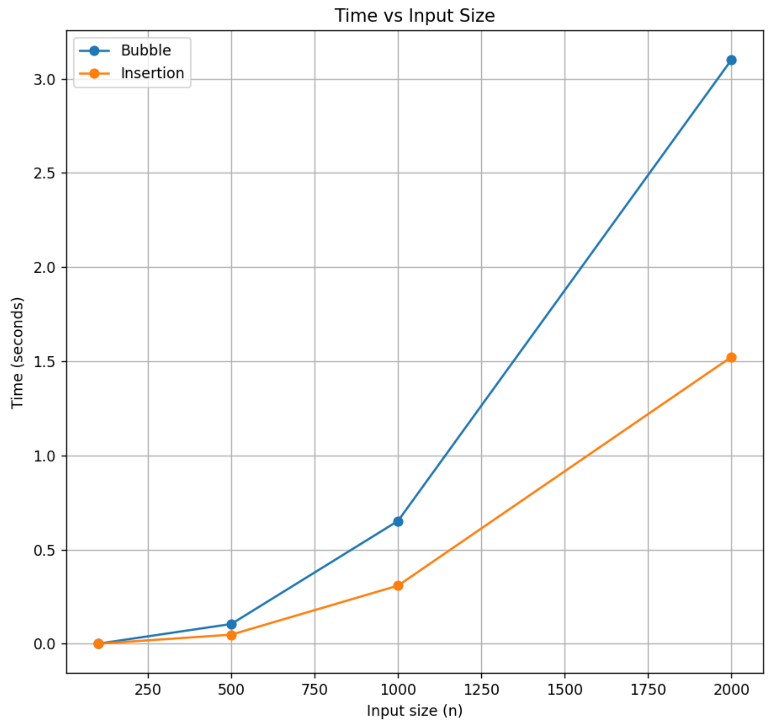
Data Type: Integer

Data Storage: Array



```
size,algorithm,time_seconds,peak_memory_kb
100,bubble,0.000420,0.11
100,insertion,0.000180,0.08
500,bubble,0.105202,0.23
500,insertion,0.048669,0.14
1000,bubble,0.652402,0.26
1000,insertion,0.308965,0.14
2000,bubble,3.100560,0.26
2000,insertion,1.523334,0.14
```





Advantages and Applications

- Provides a **visual and interactive way** to understand algorithm performance in terms of time and memory.
- Bridges the gap between **theoretical complexity** and **real-world execution behavior**.
- Enables **real-time comparison** of multiple algorithms under varying input sizes.
- Acts as a **learning aid** for students and an effective **teaching tool** for educators.
- Assists **developers and researchers** in analyzing and choosing efficient algorithms.
- Lays the foundation for **future expansion** into advanced performance profiling or web-based tools.

References

- Comparative Analysis of Memory Performance and Processing Time of Five Sorting Algorithms Using C++ Programming Language — M. Nazril Ilham et al., *Journal of Artificial Intelligence and Engineering Applications*, 2025. link
- Using Visualization to Evaluate the Performance of Algorithms for Multivariate Time Series Classification” — E. Acuña & R. Aparicio, *Data*, 2025. link
- Key Concepts, Weakness and Benchmark on Hash Table Performance (Tapia-Fernández et al., 2022) link
- An Interactive Visualization Framework for Performance Analysis of Input Sensitive Profiles (Aprof-plot paper) link
- Interactive Visualization for Memory Reference Traces (Choudhury et al., 2008) link

Thank you