Assignment: Empirical Analysis of Sorting Algorithms in Worst-Case Scenarios

Objective

The goal of this assignment is to analyze the **time complexity when input size increases** of a sorting algorithm (e.g., Bubble Sort, Selection Sort, Insertion Sort, and Merge Sort) by implementing it on different input sizes (given), measuring execution time, plotting the results, and providing a detailed report.

Tasks

1. Implementation

- Implement sorting algorithms (e.g., Bubble Sort, Selection Sort, Insertion Sort, and Merge Sort) in a programming language of your choice (C, C++, Java, Python, etc.).
- Consider following arrays:
  - Arr1= {1,2,3,4,5}
  - Arr2= {1,2,3,4,5,6,7,8,9,10}
  - Arr3={1,2,3,4,5,6,7,8,9,0,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50}
  - Arr4={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,66,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100}
- For each size, measure the **execution time** using a high-precision timer (e.g., `clock()` in C, `System.nanoTime()` in Java, `time.time()` in Python).

2. Data Collection & Analysis

- Run the sorting algorithm **multiple times** (at least 5 runs per size) to ensure consistency.
- Record the **average execution time** for each input size.
- Plot a **graph** (using Excel, Python's `matplotlib`, or any other tool) with:
  - **X-axis**: Input size (N)

- **Y-axis**: Average execution time (in milliseconds or microseconds)
- Compare the empirical results.

3. Report Submission

Prepare a **PDF report** containing:

1. **Introduction**: Briefly explain the sorting algorithm and its analysis for the given input array(s).
2. **Methodology**:
   - How execution time was measured.
3. **Results & Graph**:
   - Table of average execution times for each N.
   - Graph plotting time vs. input size.
4. **Analysis**:
   - Does the empirical growth match the theoretical complexity?
   - Any anomalies or deviations observed?
5. **GitHub Repository Link**:
   - A public Git repository (GitHub/GitLab) containing:
   - Source code implementation.

Submission Guidelines

- Submit the **PDF report** (clearly named as `SAP_Name_SortingAnalysis.pdf`).
- Ensure the GitHub repository is **publicly accessible** and includes a `README.md` explaining how to run the code.

Evaluation Criteria

- Correct implementation of the sorting algorithm.
- Proper worst-case input generation.
- Accurate time measurement and statistical averaging.
- Correct graph plotting and analysis.
- Clarity and completeness of the report.