	School of Engineering & Technology	
	Department: SOET	**Session: 2025-26**
	Program: BCA (AI&DS)	**Semester: V**
	Course Code: ENCA351	**Number of students: 188**
	Course Name: Design and Analysis of Algorithms Lab	**Faculty: Dr. Aarti**

Lab Assignment -1: Algorithm Foundations

Instructions:

- Assignment must be submitted within the deadline communicated by the instructor at the time of release.
- Assignment must be submitted on https://lms.krmangalam.edu.in/
- You must provide a link to your GitHub repository with your submission on LMS.
- Use of ChatGPT and similar tools is strictly prohibited.
- The assignment needs to be submitted by each individual.
- This assignment carries a total of 10 marks.
- Assignment will be assessed based on the evaluation rubrics.
- Estimated Duration: 5-6 hours

Assignment Title: Analyzing and Visualizing Recursive Algorithm Efficiency

---

Real-World Problem Context

Software engineers and computer scientists often face the challenge of choosing the most efficient algorithm for a given problem. Recursive and iterative algorithms, though conceptually simple, vary significantly in performance based on input size and implementation strategy. The goal of this project is to analyze and compare common sorting and searching algorithms, visualize their time and space behavior, and understand the practical impact of their design.

Learning Objectives

By completing this project, you will:

1. Apply key algorithmic characteristics (finiteness, input/output, effectiveness, etc.).
2. Visualize time and space trade-offs using plots and profiling tools.

3. Communicate algorithm behavior and trade-offs effectively through documentation.

Assignment Tasks

Task 1: Workspace Bootstrap

- Create a private GitHub repository named: algo-efficiency-mini-project-<yourname>
- Add README.md and .gitignore.
- Create a Python virtual environment.
- Install necessary packages: matplotlib, numpy, memory_profiler, time, jupyter.
- Create and validate a Jupyter notebook to run your code.

Task 2: Algorithm Selection and Design

- Implement the following algorithms in Python:
    - **Fibonacci** (naïve recursive and dynamic programming version)
    - **Merge Sort**
    - **Quick Sort**
    - **Insertion Sort**
    - **Bubble Sort**
    - **Selection Sort**
    - **Binary Search**
- For each algorithm, document:
    - Input/output
    - Time complexity (best, average, worst)
    - Space usage
    - Suitability and trade-offs

Task 3: Experimental Profiling & Visualization

- For each algorithm:
    - Use time and memory_profiler to measure actual performance.
    - Plot execution time vs. input size using matplotlib.
    - Discuss trade-offs in time vs. space complexity.
    - Comment on recursive depth and stack overflow risks, if any.

Task 4: Final Summary and Documentation

- Include:
    - A summary table comparing the algorithms and their performance characteristics
    - Visual insights in markdown or exported as PDF
    - Reflections on observed vs. expected behavior
    - README.md with proper citations and usage instructions

Evaluation Rubric (Total: 10 Marks)

Criteria	Marks	What is Expected
1. GitHub Setup & Organization	1	Proper repo setup with README, .gitignore, and organized files
2. Algorithm Implementation	3	Correct and clear Python code for all seven algorithms
3. Performance Profiling & Plots	2.5	Execution time and memory usage plots with meaningful input sizes
4. Analysis & Insights	2.5	Clear explanation of results, trade-offs, and challenges (e.g., recursion depth)
5. Code Quality & Documentation	1	Clean code, inline comments, markdown explanations, proper citations in README

Submission Instructions

- Push the following to your GitHub repository:
    - README.md: project overview and instructions
    - algo_analysis_notebook.ipynb: with all code, plots, and analysis
    - images/: folder with exported plots (optional)
    - requirements.txt: all required packages
    - .gitignore and virtual environment files (no need to upload full env folder)
- Submit GitHub repo link via LMS

Academic Integrity & Plagiarism Policy

- This is an individual project.
- Code, plots, or writeups copied from peers or external sources will result in zero marks.

- Cite external references properly in the README.

Support Resources

- CLRS – “Introduction to Algorithms”
- Python Time and Memory Profiling Docs
- matplotlib and memory_profiler tutorials

Contact for Queries: Dr. Aarti, aarti.sangwan@krmangalam.edu.in