# Project Assignment 1

## DAA SEC-R

## Comparative Study and Complexity Analysis of Sorting Algorithms

Total Marks : 20

## 🎯 Objective

To design, implement, and analyze various sorting algorithms, comparing their **theoretical** and **experimental** time complexities under different input sizes and data characteristics.

## 🧠 List of Sorting Algorithms to Implement

1. **Bubble Sort**
2. **Selection Sort**
3. **Insertion Sort**
4. **Merge Sort**
5. **Quick Sort** (both deterministic and randomized versions)
6. **Heap Sort**
7. **Counting Sort**
8. **Radix Sort**
9. **Bucket Sort**

(Optional for high performers: Shell Sort, Tim Sort, or Intro Sort)

# 📘 Tasks

1. **Algorithm Design & Implementation**
    - Implement each sorting algorithm in C/C++/Python.
    - Include proper comments, pseudocode, and modular design.
    - Ensure your implementation supports multiple array sizes and randomized data.
2. **Theoretical Complexity Analysis**
    - For each algorithm, provide **Time and Space Complexity** in:
        - Best Case
        - Average Case
        - Worst Case
    - Present results in tabular form.
3. **Experimental Analysis**
    - Measure *actual execution time* for varying input sizes (100, 500, 1000, 5000, 10000, 50000, 100000, 500000, 1000000, 5000000, 10000000).
    - Perform experiments under different input conditions:
        - Random data
        - Sorted data (best case)
        - Reverse sorted data (worst case)
    - Use system time or high-resolution clock to record runtimes.
4. **Graphs**
    - For **each algorithm**, draw a separate graph:
        - **X-axis:** Input size (n)
        - **Y-axis:** Time (ms or sec)
        - Include both **Theoretical Curve** (expected trend) and **Experimental Data Points**.
    - Use tools like Matplotlib (Python), Excel, or MATLAB.

5. **Robustness**

    - Repeat each experiment multiple times (e.g., 5 trials per input size).
    - Take the **average runtime** to minimize noise.
    - Clearly mention hardware/software configuration used.

6. **Conclusion**

    - Compare and interpret results:
        - Which algorithm performs best for small vs large data?
        - How do divide-and-conquer algorithms scale?
        - When do theoretical expectations diverge from experimental results?
        - And more what you see and observe.

| Criteria | Description | Marks |
|---|---|---|
| **Algorithm Design & Implementation** | Correct algorithmic logic, efficiency, correctness of code | **6** |
| **Complexity Analysis** | Theoretical + experimental (plots or tables comparing runtime) | **4** |
| **Documentation & Report** | Clear explanation, pseudocode, results discussion, references | **4** |
| **Viva / Presentation** | Understanding of algorithm design, performance, and trade-offs | **6** |
| **Total** | | **20** |

## Submission Requirements

- Create a folder include
- Source code with comments
- Short report (4–6 pages) (PDF):
    - Problem description
    - Algorithm and pseudocode
    - Time and space complexity analysis
    - Experimental setup & results
    - Conclusion and future scope

- Graphs or charts comparing performance
- Viva questions based on algorithmic design principles.