Project Title: SENG 313 Midterm Project

Objectives:

1. Apply time and space complexity concepts to algorithm analysis.
2. Implement and compare fundamental algorithms covered in class.
3. Explore the effect of input size on algorithm performance.

Requirements:

1. **Algorithm Implementation**: Write code to implement each algorithm.
2. **Complexity Analysis**: Analyze the time and space complexity of each algorithm and explain how they behave with different input sizes.
3. **Empirical Analysis**: Test the algorithms on datasets of varying sizes and provide runtime measurements (small, medium, large).
4. **Report**: A 2-3 page(without cover) report that includes:
    - Problem description
    - Algorithms used
    - Complexity analysis
    - Empirical results with graphs to visualize performance.

Deliverables:

1. **Source Code**: Code for each algorithm, with comments explaining key sections.
2. **Report**: A structured report as outlined above.

Evaluation Criteria:

1. Correctness of the implementations (25%)
2. Depth of complexity analysis (25%)
3. Quality of empirical performance analysis (25%)
4. Clarity of report (25%)

QUESTIONS

Problem 1: Sorting Customer Orders for a Sales Dashboard

You work on the back-end system for a retail platform that processes daily customer orders. Sales managers need to see the highest-value orders at the top of their dashboard to prioritize shipment, and during major sales events like Black Friday, the number of orders increases significantly.

Task:

- Implement two sorting algorithms of your choice.
- Test and compare their performance on datasets representing a typical day's orders (small dataset) and during peak sales periods (large dataset).
- Decide which sorting algorithm is better suited for each situation based on time and space complexity.

Problem 2: Product Search in an Online Store

The product search function on an e-commerce website must find products quickly, whether the customer provides the exact product name or a partial match (e.g., "sneaker" might match "running sneaker" or "sneaker shoe").

Task:

- Implement two search algorithms: one for exact match and one for partial match
- Compare their performance on product catalogs of varying sizes (small, medium, and large).
- Decide which algorithm performs best for each situation and explain how search times vary depending on dataset size and search type.

Problem 3: Sorting Medical Test Results

You are developing software for a hospital that handles large sets of patient test results. Medical professionals need to see these results sorted by urgency (critical results first) to prioritize patient care. During times of high patient volume, the system must still function efficiently without delays.

Task:

- Choose and implement two sorting algorithms to rank patient test results by urgency.
- Compare how the algorithms perform on small and large datasets, and analyze their behavior when there are ties (i.e., multiple tests with the same urgency).

- Explain which sorting algorithm is better suited for a healthcare environment with real-time data processing needs.

Problem 4: Search for Missing Documents in a Government Archive

A government archive contains millions of documents, but due to human error, some documents are misplaced. You need to develop a system that searches for these missing documents by either exact document ID or partial metadata (like document title or author).

Task:

- Implement two search algorithms: one for finding exact document IDs and one for searching through partial metadata.
- Test and compare the efficiency of the algorithms on different archive sizes (e.g., 10,000, 100,000, and 1 million documents).
- Recommend the best approach for each situation and discuss the trade-offs between the algorithms in terms of search speed and accuracy.

Instructions

- **Use the Provided Datasets:** Start by using the sample datasets provided above to implement and test your algorithms.
- **Dataset Modifications:** I have created datasets with 10,100,1000 and 10000 inputs, but feel free to add more records or modify the data to create different scenarios for testing. To do this, use "Dataset_Production.ipynb" file.
- **Algorithm Selection:** Based on the problem description, decide which algorithms are most appropriate to implement.
    - For sorting problems, consider algorithms like Bubble Sort, Insertion Sort, Merge Sort, or Quick Sort.
    - For searching problems, consider Linear Search, Binary Search, or other suitable algorithms.
- **Complexity Analysis:** Analyze the time and space complexity of your chosen algorithms and explain why they are suitable for the given problem.
- **Empirical Testing:** Measure the performance of your algorithms using the datasets and discuss how the results align with your theoretical analysis.
- **Report Writing:** Summarize your findings in a clear and concise report, including any graphs or tables that support your analysis.