Algorithm Project Work summary

Overview

This project is divided into three tasks, each focused on applying different algorithmic techniques and analyzing their performance. You are required to submit a Jupyter Notebook with all your code and results, along with up to 10 slides summarizing your findings. A final discussion will be held with me and Professor Finocchi.

You have 3 datasets, you must use all of them

Task 1: Data Exploration and Sorting Algorithms

Objectives:

1. Explore and understand the dataset:
    - Perform basic statistical analysis using Python built-in methods (e.g., mean, min, max, median or others).
    - Focus on identifying the most informative columns.
    - Visualize data using plots and tables
2. Apply and compare sorting algorithms:
    - Implement various sorting algorithms (e.g., Bubble Sort, Merge Sort, Quick Sort).
    - Measure and compare their performance (execution time and efficiency) across different datasets.
    - Use appropriate plots to show performance comparisons (e.g., line graphs, bar charts).

Task 2: Search Algorithms and Binary Search Trees (BST)

Objectives:

1. Perform search operations on your data:
    - Implement and compare Linear Search and Binary Search to find any info of your choice
    - You can find examples in the slides

- Evaluate performance across different datasets in terms of time and efficiency.

2. Work with Binary Search Trees (BST):
    - Build a BST from your dataset.
    - Find information of your choice through Binary Search Tree
    - Compare with Binary Search and Linear Search

    Examples of what you could search for:

    - **Find the Most Frequent Airline**
    - **Find All Flights From a Given Origin**
    - **Find Flights with the Longest Delays**
    - **Find flights numbers in a range**

3. Compare search techniques:
    - Perform dictionary lookups and compare performance with BST, Linear Search and Binary Search.

Task 3: Graphs and Connected Components

Objectives:

1. Model a flight network:
    - Each node represents an airport.
    - An edge between two nodes exists if there is at least one flight between them.
    - Edge weights should be proportional to the number of flights between the airports.
    - Use the NetworkX library to build and visualize the graph or also python and matrices if you wish.
2. Find connected components:
    - Treat the graph as **UNDIRECTED**.
    - Use different methods to identify connected components:
        - Level 1: Use networkx (basic evaluation).
        - Level 2: Implement either DFS or BFS and Network X to find connected components (intermediate evaluation).

        - Level 3: Implement both DFS and BFS and NetworkX (advanced evaluation).
3. Compare performance:
    - Measure execution times of DFS and BFS
    - Visualize performance differences with graphs or tables.

Deliverables:

1. Python Notebook with:
    - Complete code for all tasks
    - Outputs and visualizations
    - Performance comparisons
2. Presentation Slides (max 10) summarizing:
    - Key findings from each task
    - Visual highlights (charts, diagrams)
    - Observations and conclusions
3. Project Discussion with me and Professor Finocchi.