Final Exam Part B

Start Assignment

Due Friday by 4:59pm **Points** 80 **Submitting** a file upload
Available Dec 14 at 1:05pm - Dec 15 at 5:59pm

Part B comprises two questions (80 points).

The due date is Dec 15th, 4:59 pm. Late submission is accepted until 5:59 pm, but it will be penalized with 15 points.

If you need to consider reasonable assumptions, please include them in your report.

Please upload a separate PDF per question in addition to the Jupyter Notebook file.

Question 1 (50 points)

You are tasked with designing a system to manage a large dataset of social media connections. The dataset comprises billions of users and their connections (friendship links) in a social network. Each user has an ID and maintains a list of their friends' IDs. Every user has defined hobbies, music, and movie preferences.

Design an efficient system that:

1. Store and manage this dataset considering the massive scale.
2. Allows for quick retrieval of a user's friends or the friends of friends up to a certain level.
3. Implements algorithms to suggest new friends to a user based on mutual friends or other relevant criteria (preferences of Hobbies, Music/Movies).
4. Optimizes for space and time complexities in operations like adding new users, adding connections, and suggesting new connections.
5. Define a name and a description for the classes and functions/methods you will implement. Add this list to your report.

Your solution should include:(must be included in your report)

1. A mind map with images and text for your approach encourages creativity and imagination in presentation. (take some ideas about mind maps here: https://mindmapsunleashed.com/10-really-cool-mind-mapping-examples-you-will-learn-from (https://mindmapsunleashed.com/10-really-cool-mind-mapping-examples-you-will-learn-from)).
2. A general approach to your solution (4-5 paragraphs)

3. A detailed explanation of the data structures you would use to store the user data and their connections, considering the scale of the dataset.
4. Algorithms for efficiently retrieving friends or friends of friends within a limited degree of separation.
5. Strategies for suggesting new connections to users.
6. Analysis of the time and space complexities of your proposed solution, highlighting its efficiency and scalability.

Code/Jupyter Notebook:

A Python code with a small dataset including users (with hobbies/music/movie preferences), their friends, and a recommendation working algorithm.

Additional Notes:

- Consider scenarios where the dataset continues to increase.
- Discuss any trade-offs, limitations, or challenges your proposed system might encounter.
- Please ensure that you include any rational assumptions in your report.
- Create one comprehensive Jupyter Notebook with the report and your working code.
- If you need to upload any additional files with code, such as CSV, txt, jpg, or any other important files, please submit them to Canvas. Please make sure that you compile your solution with the uploaded files.

Grading:

- 5 points Diagram a Mind Map of your solution.
- 25 points for the details of explanations/descriptions, algorithms used, and their complexity (Space and Time), classes, and methods (names and tasks).
- 20 for the running Code in Python. (Jupyter Notebook)

This question encompasses multiple aspects of data structures (like graphs, trees, hash maps), algorithms (graph traversal, recommendation algorithms), and problem-solving strategies (optimization, scalability). It challenges you to apply your understanding of these concepts and think critically about designing a system that addresses real-world issues at a large scale.

Question 2 (30 points)

Your task is to create a comprehensive guide demonstrating basic CRUD operations (Create, Read, Update, Delete) for a selection of fundamental data structures in Python: sets, tuples, dictionaries, lists, linked lists, stacks, and queues.

Your guide should cover:

1. **Sets:** Demonstrate how to create sets, add/remove elements, check membership, and set operations (union, intersection, difference).
2. **Tuples:** Explain tuple immutability and showcase creating tuples, accessing elements, concatenating tuples, and unpacking values.
3. **Dictionaries:** Illustrate creating dictionaries, adding key-value pairs, accessing/modifying values, deleting entries, and iterating through keys/values.
4. **Lists:** Show creating lists, appending/removing elements, accessing elements by index, slicing, and list operations (concatenation, extending, sorting).
5. **Linked Lists:** Implement a linked list with basic operations: adding elements at the beginning/end, removing elements, and traversing the list.
6. **Stacks:** Create a stack using a linked list or list and demonstrate stack operations (push, pop, peek) and its LIFO (Last In, First Out) behavior.
7. **Queues:** Implement a queue using a linked list or list and demonstrate queue operations (enqueue, dequeue) and its FIFO (First In, First Out) behavior.

In your guide, provide well-commented code examples for each data structure, explaining the purpose of each operation and discussing scenarios where one data structure might be more suitable than another. In addition, explain the appropriate use of each data structure in a real-world case without implementing the case in code.

Discuss the time complexities of the operations for each data structure and any significant differences or optimizations to consider.

Delivery: Your guide must be contained in one Jupyter Notebook file with at least 7 sections. Each section is a data structure; additional sections can be added if needed. Use text and images to support your guide with exercises. Upload two files, one Jupyter Notebook and another PDF (of your Jupyter Notebook tutorial).

Grading: Every Data structure guide is worth 4 points, and the overall flow/design/learning process is worth 2 points. Total 30 points