SOFE 4820U Modelling & Simulation

Winter 2024

Due Date: By 11:59 pm on Feb 25, 2024

No late assignments will be accepted!

Assignment 1: Random Walk for Graph Connectivity Testing

- **Introduction**

Graphs serve as powerful tools for modelling relationships between entities, and understanding the connectivity of a graph is crucial in various applications. This assignment explores the design and implementation of a random walk algorithm to test the connectivity of a given graph.

- **Objective**

The primary objective of this assignment is to design and implement a robust random walk algorithm to assess the connectivity of a graph. The design component emphasizes creating a solution that is modular, scalable, and flexible for different types of graphs.

- **Requirements**

Understanding Graph Connectivity:

Review the concepts of graph connectivity, including connected components and the characteristics of undirected and directed graphs.

- **Python and NetworkX:**

Ensure Python is installed on your machine.

Install the NetworkX library using pip install networkx.

- **Design Component:**

Design a modular random walk algorithm that can be easily adapted for different types of graphs.

Consider the use of functions or classes to encapsulate different aspects of the algorithm.

Think about how your design can accommodate future modifications or extensions.

- **Implementation:**

Implement the designed random walk algorithm in Python.

Test the algorithm on various graphs, ensuring it can handle both undirected and directed graphs.

- **Parameterization:**

Allow for flexible parameterization of the algorithm, such as adjusting the number of walks (num_walks) and maximum steps per walk (max_steps).

Justify the chosen parameters based on the characteristics of the graphs being tested.

- **Checking Connectivity:**

Develop a mechanism to check the connectivity of the graph after completing random walks.

Ensure the solution provides meaningful insights into whether the graph is connected.

- **Documentation:**

Document the design decisions made during the development process.

Provide clear and concise comments in your code, explaining the purpose of each module or function.

- **Testing and Analysis:**

Conduct thorough testing on different types of graphs.

Analyze the results, highlighting the performance and adaptability of your designed solution.

- **Reflection:**

Reflect on the design choices made and how they contribute to the overall scalability and modularity of the solution.

Discuss potential improvements or modifications that could be made in the future.

- **Submission Guidelines**

Submit your Python code file containing the implementation of the random walk algorithm.

Include a design document (PDF) explaining the modular design and justifying the parameter choices.

Submit your testing and analysis document discussing the results of the tests conducted.

Submission Deadline:

Submit your assignment by Feb. 25, 11:59 pm.

- **Evaluation Criteria**

Your assignment will be evaluated based on the following criteria:

  - Design Component:

  Clarity and effectiveness of the modular design.

  Consideration of future modifications or extensions.

  - Code Quality:

  Proper implementation of the designed algorithm.

  Modular and well-structured code.

  - Documentation:

Comprehensive design document explaining the modular design.

Clear comments in the code.

- Testing and Analysis:

Thorough testing on different graphs.

Insightful analysis of the results.

- Reflection:

Thoughtful reflection on the design choices and their impact on the solution.

- **Conclusion**

This assignment challenges you to not only implement a random walk algorithm but also to think critically about the design decisions involved. By focusing on modularity and flexibility, you will develop a solution that can be easily adapted for various graph connectivity testing scenarios.